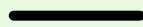


The logo for ISMIR2021 features the text "ISMIR2021" in a bold, dark grey sans-serif font. The text is centered and overlaid on a stylized graphic of radiating lines in shades of yellow, orange, and green, resembling a sunburst or a musical instrument's sound waves. The background of the entire cover is a light green color with abstract geometric shapes: a large light green circle in the top left, a smaller circle with black diagonal stripes in the top left, a dark grey diagonal line in the top left, a large yellow circle in the bottom right, a smaller yellow circle with black diagonal stripes in the bottom right, a dark grey diagonal line in the bottom right, and wavy black lines in the bottom left and top right corners.

ISMIR2021

THE 22ND INTERNATIONAL SOCIETY FOR
MUSIC INFORMATION RETRIEVAL CONFERENCE

PROCEEDINGS



NOVEMBER 7-12, 2021 (ONLINE)

ISMIR 2021 was organized virtually online by the International Society for Music Information Retrieval, Women in Music Information Retrieval (WiMIR) and a diverse international committee of organizers.

Website: <https://ismir2021.ismir.net>

ISMIR 2021 logo design: ZETA@DesignCrowd

Edited by:

Jin Ha Lee (*University of Washington, Seattle, USA*)

Alexander Lerch (*Georgia Institute of Technology, Atlanta, USA*)

Zhiyao Duan (*University of Rochester, Rochester, USA*)

Juhan Nam (*Korea Advanced Institute of Science and Technology, Daejeon, South Korea*)

Preeti Rao (*Indian Institute of Technology Bombay, Mumbai, India*)

Peter Van Kranenburg (*Utrecht University, Utrecht, The Netherlands*)

Ajay Srinivasamurthy (*Amazon Alexa, Bengaluru, India*)

ISBN: 978-1-7327299-0-2

Title: Proceedings of the 22nd International Society for Music Information Retrieval Conference, Online, Nov 7-12, 2021.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee, provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page.

© 2021 International Society for Music Information Retrieval



ISMIR Organizers



ISMIR

ISMIR Sponsors

Platinum Sponsors

SONY



Gold Sponsors



pandora[®]



Silver Sponsors



ACRCloud



utopia[™]



WIMIR Sponsors

Patrons



Contributors



Supporters



Organizing Team

Conference Chairs

General Chairs

Jin Ha Lee (*University of Washington, Seattle, USA*)

Alexander Lerch (*Georgia Institute of Technology, Atlanta, USA*)

Scientific Program Chairs

Zhiyao Duan (*University of Rochester, Rochester, USA*)

Juhan Nam (*Korea Advanced Institute of Science and Technology, Daejeon, South Korea*)

Preeti Rao (*Indian Institute of Technology Bombay, Mumbai, India*)

Peter Van Kranenburg (*Utrecht University, Utrecht, The Netherlands*)

Tutorial Chairs

Yi-Hsuan Yang (*Academia Sinica, Taipei, Taiwan*)

Ashis Pati (*Apple, Portland, USA*)

Late-Breaking / Demo Chairs

Li Su (*Academia Sinica, Taipei, Taiwan*)

Chih-Wei Wu (*Netflix, Los Gatos, USA*)

Siddharth Kumar Gururani (*NVIDIA, Santa Clara, USA*)

Music Program Chairs

Carlos Guedes (*New York University, Abu Dhabi, UAE*)

Bochen Li (*ByteDance, Mountain View, USA*)

Diversity and Inclusion Chairs

Blair Kaneshiro (*Stanford University, USA*)

Jordan Smith (*TikTok, London, UK*)

Technology Chairs

Sharath Adavanne (*Zapr Media Labs, Bengaluru, India*)

Stefan Balke (*pmOne Group, Paderborn, Germany*)

Andrea Cogliati (*LighTopTech Corp, West Henrietta, USA*)

Hendrik Schreiber (*tagtraum industries, Cologne, Germany*)

Fabian-Robert Stöter (*Audioshake.ai, Montpellier, France*)

Publications Chair

Ajay Srinivasamurthy (*Amazon Alexa, Bengaluru, India*)

Sponsorship Chairs

Sertan Şentürk (*Kobalt Music, London, UK*)

Alia Morsi (*Universitat Pompeu Fabra, Barcelona, Spain*)

Lamtham (Hanoi) Hantrakul (*TikTok, San Francisco, USA*)

Social Program

Oriol Nieto (*Adobe, San Francisco, USA*)

Social Media Chair

Qhansa Bayu (*Telkom University, Indonesia*)

Volunteers Chair

Kahyun Choi (*Indiana University, Bloomington, USA*)

Lab Showcase Chair

Lele Liu (*Queen Mary University of London, London, UK*)

Newcomer Initiatives Chairs

Nick Gang (*Apple, San Francisco, USA*)

Elona Shatri (*Queen Mary University of London, London, UK*)

Website Chair

Ashvala Vinay (*Georgia Institute of Technology, Atlanta, USA*)

Volunteers

Jeremiah Abimbola

Oğuz Araz

Rafael Arias

Sultan Eylem Atabay

Hasan Sercan Atlı

Hemantha Bharadwaj

Sanga Chaki

Qinyi Chen

Robertus Chris

Annie Chu

Stewart Engart

Maurice Frank

Lingan Gu

Dilip H

Nora Hernandez

Carlos Hernández Oliván

Mohit Jain

Dias Khalniasov

Warsha Kiringoda

Ingrid Knochenhauer de Souza Mendes

Krishna Kumar

Keon J. Lee

Ruilun Liu

Giorgio de Luca

Gurunath Reddy Madhumani

Brian McCorkle

Baibhav Nag

Jyoti Narang

Ke Nie

Soroush Omranpour

Amir Abbas Orouji

Varvara Papazoglou

Lorenzo Porcaro

Ying Que

Gowriprasad R

Ameaza Rodrigues

Anastasia Sagalovitch

Partha Pratim Saha

Shreshth Saxena

Joshua Schlichting

Harshita Seth

Yi Shan

Dinuka de Silva

Ajeet Kumar Singh

Jaya Srivastava

Vinod Subramanian

Lela Tvaliashvili

Carrus Wan

Zuo Wang

Karn Watcharasupat

Gareth Wong

Yiting Xia

Yixiao Zhang

Huan Zhang

Liu Zhaorui

Program Committee

Meta-Reviewers

Kat Agres, National University of Singapore
Vipul Arora, Indian Institute of Technology, Kanpur
Christine Bauer, Utrecht University
Juan Pablo Bello, New York University
Emmanouil Benetos, Queen Mary University of London
Rachel Bittner, Spotify
Nicholas J. Bryan, Adobe Research
John Ashley Burgoyne, University of Amsterdam
Carlos Eduardo Cancino-Chacón, Johannes Kepler University Linz
Estefania Cano, Fraunhofer IDMT
Mark Cartwright, New Jersey Institute of Technology
Kahyun Choi, Indian University Bloomington
Ching-Hua Chuan, University of Miami
Tom Collins, University of York; MAIA, Inc.
Julie Cumming, McGill University
Roger Dannenberg, School of Computer Science, Carnegie Mellon University
Matthew Davies, Centre for Informatics and Systems of the University of Coimbra (CISUC)
Johanna Devaney, Brooklyn College
Christian Dittmar, Fraunhofer IIS
Simon Dixon, Queen Mary University of London
Chris Donahue, Stanford University
Stephen Downie, University of Illinois at Urbana-Champaign
Philippe Esling, IRCAM
Sebastian Ewert, Spotify
Gyorgy Fazekas, Queen Mary University of London
Arthur Flexer, Johannes Kepler University Linz
Ichiro Fujinaga, McGill University
Emilia Gomez, Universitat Pompeu Fabra
Masataka Goto, National Institute of Advanced Industrial Science and Technology (AIST)
Fabien Gouyon, Pandora/SiriusXM
Romain Hennequin, Deezer Research
Dorien Herremans, Singapore University of Technology and Design
Andre Holzapfel, KTH Royal Institute of Technology in Stockholm
Xiao Hu, The University of Hong Kong
Ozgur Izmirlı, Connecticut College
Dietmar Jannach, University of Klagenfurt
Katherine M. Kinnaird, Smith College
Peter Knees, Technische Universität Wien
Audrey Laplante, Université de Montréal
Olivier Lartillot, RITMO, University of Oslo
Kyogu Lee, Seoul National University
Florence Levé, Université de Picardie Jules Verne - Lab. MIS - Algomus
Cynthia C. S. Liem, Delft University of Technology
Michael Mandel, Brooklyn College, CUNY
Brian McFee, New York University
Cory McKay, Marianopolis College
Andrew McLeod, École polytechnique fédérale de Lausanne (EPFL)
Matt McVicar, Apple
Meinard Müller, International Audio Laboratories Erlangen
Hema A. Murthy, Indian Institute of Technology Madras
Tomoyasu Nakano, National Institute of Advanced Industrial Science and Technology (AIST)
Oriol Nieto, Adobe
Mitsunori Ogihara, University of Miami
Bryan Pardo, Northwestern University
Johan Pauwels, Imperial College London

Geoffroy Peeters, LTCI - Télécom Paris, IP Paris
Jordi Pons, Dolby Laboratories
Marcelo Queiroz, University of São Paulo
David Rizo, Universidad de Alicante
Axel Roebel, IRCAM
Justin Salamon, Adobe Research
Markus Schedl, Johannes Kepler University Linz
Alexander Schindler, Austrian Institute of Technology
David Sears, Texas Tech University
Xavier Serra, Universitat Pompeu Fabra
Mohamed Sordo, Pandora
Sebastian Stober, Otto von Guericke University
Bob L. T. Sturm, KTH Royal Institute of Technology
Li Su, Academia Sinica
Douglas Turnbull, Ithaca College
George Tzanetakis, University of Victoria
Julián Urbano, Delft University of Technology
Gabriel Vigliensoni, Goldsmiths University of London
Cheng-i Wang, Smule, Inc.
Ye Wang, National University of Singapore
Christof Weiss, International Audio Laboratories Erlangen
Gerhard Widmer, Johannes Kepler University
Gus Xia, New York University Shanghai
Yi-Hsuan Yang, Academia Sinica
Kazuyoshi Yoshii, Kyoto University
Eva Zangerle, University of Innsbruck

Reviewers

Jakob Abeßer, Fraunhofer IDMT
Stefanie Acevedo, University of Dayton
Byron Alejandro Acuña Acurio, Universidade Estadual de Campinas
Sharath Adavanne, Tampere University of Technology
Islah Ali-Maclachlan, Birmingham City University
Anna Aljanaki, University of Tartu
Vinoob Alluri, IIIT - Hyderabad
Pablo Alonso-Jiménez, Universitat Pompeu Fabra
Ishwarya Ananthabhotla, MIT Media Lab
Nazareno Andrade, Universidade Federal de Campina Grande
Stefan Balke, Johannes Kepler University Linz
Doğaç Başaran, Audible Magic
Oded Ben-Tal, Kingston University
Gilberto Bernardes, INESC TEC & University of Porto
Louis Bigo, Université de Lille
Dmitry Bogdanov, Universitat Pompeu Fabra
Geoffray Bonnin, LORIA
Nicolas Bouillot, Société des arts technologiques, Montreal, Canada
Paul Brossier, aubio
Dan Brown, University of Waterloo
Michele Buccoli, BdSound S.r.l.
Bryony Buck, University of Nottingham
Marcelo Caetano, CNRS-PRISM
Jorge Calvo-Zaragoza, University of Alicante
Pavel Campr, Pex.com
Giorgia Cantisani, Telecom ParisTech
Julio Carabias, University of Jaen
Rafael Caro Repetto, Universitat Pompeu Fabra
Benjamin Carterette, Spotify

Santosh Chapaneri, St. Francis Institute of Technology, Mumbai
Bo-Yu Chen, Academia Sinica
Tianyao Chen, NYU Shanghai
Tsung-Ping Chen, Academia Sinica
Yu-Hua Chen, NTU
Tian Cheng, National Institute of Advanced Industrial Science and Technology (AIST)
Diana Estefania Chérrez Barragán, Universidade Estadual de Campinas
Paulo Chiliguano, Audio Engineering Society
Hyeong-Seok Choi, Seoul National University
Keunwoo Choi, ByteDance
Soonbeom Choi, KAIST
Shreyan Chowdhury, Johannes Kepler University Linz
Ching-Hua Chuan, University of Miami
Chia-Hao Chung, GICE, NTU
Ondřej Cífka, Télécom Paris
Ana Clemente, University of the Balearic Islands
Nathaniel Condit-Schultz, Georgia Institute of Technology
Bas Cornelissen, University of Amsterdam
Helene-Camille Crayencour, CNRS
Laura Cros Vila, KTH
Helena Cuesta, Universitat Pompeu Fabra
Shuqi Dai, Carnegie Mellon University
Antonio Deusany de Carvalho Junior, USP
Timothy R. de Reuse, McGill University
Reinier de Valk, Moodagent
Alessio Degani, Università degli Studi di Brescia
Andrew M. Demetriou, Delft University of Technology
Emir Demirel, Queen Mary University of London

Junqi Deng, Alibaba
Hao-Wen Dong, UC San Diego
Guillaume Doras, IRCAM
Konstantinos Drossos, Tampere University
Jake Drysdale, Birmingham City University
Simon Durand, Spotify
Jeffrey Ens, Simon Fraser University
Behnam Faghhi, Maynooth University
Felipe V Falcão, Universidade Federal de Campina Grande
Jianyu Fan, Simon Fraser University
Xavier Favory, Universitat Pompeu Fabra
Andres Ferraro, Universitat Pompeu Fabra
Christoph Finkensiep, École polytechnique fédérale de Lausanne
Frederic Font, Universitat Pompeu Fabra
Mathieu Fontaine, RIKEN AIP
Dominique Fourer, IBISC
Klaus Frieler, University of Music "Franz Liszt" Weimar
Magdalena Fuentes, New York University
Satoru Fukayama, National Institute of Advanced Industrial Science and Technology (AIST)
Diego Furtado Silva, Universidade Federal de São Carlos
Nick Gang, Apple, Inc.
Kaustuv Kanti Ganguli, New York University Abu Dhabi
Roman B. Gebhardt, Cyanite / Audio Communication Group, TU Berlin
Elena Georgieva, Stanford University
François Germain, iZotope, Inc.
Jon Gillick, UC Berkeley
Mathieu Giraud, CNRS, Université de Lille
Aggelos Gkiokas, Universitat Pompeu Fabra
Juan S. Gómez-Cañón, Universitat Pompeu Fabra
Samuel P Goree, Indiana University Bloomington
Mark R. H. Gotham, Universität des Saarlandes
Richard Groult, Université de Rouen Normandie
Joris A. J. Grouwels, KTH
Yupeng Gu, Indiana University
Carlos Guedes, NYU Abu Dhabi
Chitralekha Gupta, National University of Singapore
Siddharth Gururani, Georgia Institute of Technology
Ranjani H. G., Ericsson India
Gaëtan Hadjeres, Sony Computer Science Laboratories
Yoonchang Han, Cochlear.ai
Yun Hao, University of Illinois at Urbana-Champaign
Daniel Harasim, École Polytechnique Fédérale de Lausanne
Curtis Hawthorne, Google Brain
Ben Hayes, Queen Mary University of London
Florian Henkel, Johannes Kepler University Linz
Jorge Herrera, Shazam
Peyman Heydarian, University of Waikato
Ulf A. S. Holbrook, University of Oslo
Jiawen Huang, Queen Mary University of London
Yu-Fen Huang, Academia Sinica
Chris Hubbles, NEDCC
Karim M. Ibrahim, Telecom-Paristech
Charles Inskip, University College London
Florent Jacquemard, Inria, CNAM-Cédric lab, Paris
Berit Janssen, Utrecht University
Dasaem Jeong, SK T-Brain
Il-Young Jeong, Cochlear.ai
Junyan Jiang, Carnegie Mellon University
Yucong Jiang, University of Richmond
Cong Jin, Communication University of China
Zeyu Jin, Adobe Research
David Johnson, Fraunhofer IDMT
Nicolas Jonason, KTH Royal Institute of Technology
Anna Jordanous, School of Computing, University of Kent
Yaolong Ju, McGill University
Maximos Kaliakatsos-Papakostas, Aristotle University of Thessaloniki
Stefano Kalonaris, ΣΚ
Tejaswinee Kelkar, University of Oslo
Jong Wook Kim, OpenAI
Jaehun Kim, Delft University of Technology
Minje Kim, Indiana University
Phillip B. Kirlin, Rhodes College
Tetsuro Kitahara, Nihon University
Rainer Kleinertz, Saarland University
Eunjeong Stella Koh, UC San Diego
Qiuqiang Kong, ByteDance
Radha Manisha Kopparti, City, University of London
Amanda Krause, University of Melbourne
Michael Krause, International Audio Laboratories Erlangen
Kosmas Kritsis, Athena Research Center
Sangeun Kum, Neutune
Frank Kurth, Fraunhofer FKIE
Taegyun Kwon, KAIST
Pierre Laffitte, Moodagent
Stefan Lattner, Sony Computer Science Laboratories, Paris
Juheon Lee, Seoul National University
Kyungyun Lee, KAIST
Jongpil Lee, Neutune
Simon Leglaive, CentraleSupélec
Mark Levy, Apple
Elisabeth Lex, TU Graz
Fanjie Li, The University of Hong Kong
Shengchen Li, Xi'an Jiaotong - Liverpool University
Wei Li, Fudan University
Bochen Li, University of Rochester
Rongfeng Li, Beijing University of Posts and Telecommunications
Beici Liang, Tencent Music Entertainment Group
Wei-Hsiang Liao, Sony Corporation
Elad Liebman, SparkCognition Research
Robert Lieck, EPFL
Liwei Lin, New York University Shanghai
Baihan Lin, Columbia University
Yuan-Pin Lin, National Sun Yat-sen University
Lele Liu, Queen Mary University of London
Yi-Wen Liu, National Tsing Hua University
Antoine Liutkus, Inria
Patricio López-Serrano, Zplane Development
Carlos Lordelo, Queen Mary University of London / DoReMIR Music Research AB
Vincent Lostanlen, Cornell Lab of Ornithology

Gurunath Reddy Madhumani, Indian Institute of Technology Kharagpur
Akira Maezawa, Yamaha Corporation
Lucas S Maia, Federal University of Rio de Janeiro / Télécom Paris
Ilaria Manco, Queen Mary University of London
Ethan Manilow, Interactive Audio Lab, Northwestern University
Sandy Manolios, TU Delft
Leandro Balby Marinho, Federal University of Campina Grande
Matija Marolt, University of Ljubljana
Marco A. Martínez Ramírez, Queen Mary University of London
David Martins de Matos, INESC-ID Lisboa
Matthias Mauch, Queen Mary University of London
Maximilian Mayerl, University of Innsbruck
Blai Meléndez-Catalán, Universitat Pompeu Fabra / BMAT Licensing S.L.
Angelo Cesar Mendes da Silva, University of São Paulo
Gianluca Micchi, TikTok / ByteDance
Remi Mignot, IRCAM
Marius Miron, Universitat Pompeu Fabra
Yuki Mitsufuji, Sony Group Corporation
Nicola Montecchio, Spotify
Fabio Morreale, University of Auckland
Mina Mounir, KU Leuven
Eita Nakamura, Kyoto University
Maria Navarro, University of Salamanca
Jeremy Tzi Dong Ng, The University of Hong Kong
Eric Nichols, Zillow Group
Mitsunori Ogihara, University of Miami
Sergio Oramas, Pandora
Patricio Ovalle, Independent
Piyush Papreja, Arizona State University
Emilia Parada-Cabaleiro, Johannes Kepler University Linz
Sanjeel Parekh, Télécom Paris
Saebyul Park, KAIST
So Yeon Park, Stanford University
Ashis Pati, Georgia Institute of Technology
Miguel Perez Fernandez, MTG, Huawei
Antonio Pertusa, University of Alicante
Matevž Pesek, University of Ljubljana
Pedro D. Pestana, CITAR - UCP
Aggelos Pikrakis, University of Piraeus
António Pinto, Universidade do Porto
Pedro J. Ponce de León, University of Alicante
Lorenzo Porcaro, Universitat Pompeu Fabra
Alastair Porter, Universitat Pompeu Fabra
Verena Praher, Johannes Kepler University Linz
Thomas Prätzlich, Learnfield GmbH
Katharina Prinz, Johannes Kepler University Linz
Ying Que, University of Hong Kong
Marcelo Queiroz, University of São Paulo
Zafar Rafii, Gracenote
Antonio Ramires, Universitat Pompeu Fabra
Gang Ren, University of Miami
Jemily I. Rime, University of York
Bruno Rocha, University of Coimbra, Portugal
Gerard Roma, University of Huddersfield
Iran R. Roman, NYU
Sebastian Rosenzweig, International Audio Laboratories Erlangen
Joe Cheri Ross, LinkedIn Bangalore
Robert Rowe, New York University
Germán Ruiz-Marcos, Open University
Charalampos Saitis, Queen Mary University of London
Andy Sarroff, iZotope, Inc.
Patrick E. Savage, Keio University
Frank Scherbaum, University of Potsdam
Bertrand Scherrer, LANDR
Maximilian Schmitt, University of Augsburg
Rodrigo Schramm, Universidade Federal do Rio Grande do Sul
Hendrik Schreiber, tagtraum industries
Eleanor Selfridge Field, Stanford University
Sertan Şentürk, Kobalt Music Group / Independent Researcher
Zhengshan Shi, Stanford University
Siddharth Sigtia, Apple
Federico Simonetta, Università di Milano
George Sioros, University of Oslo
Christian J. Steinmetz, Queen Mary University of London
Eli Stine, Oberlin Conservatory
Daniel Stoller, Queen Mary University of London
Fabian-Robert Stöter, Inria, France
Vinod Subramanian, Queen Mary University of London
Serkan Sulun, INESC TEC
Hao Hao Tan, Singapore University of Technology and Design
Tiago Tavares, UNICAMP
Florian Thalmann, Kyoto University
Marko Tkalcic, Free University of Bozen Bolzano
Petri Toiviainen, University of Jyväskylä
George Tourtellot, Kapitus
Philip Tovstogan, Universitat Pompeu Fabra
Caitlyn Trevor, University of Zurich
Timothy Tsai, Harvey Mudd College
Kosetsu Tsukuda, National Institute of Advanced Industrial Science and Technology (AIST)
Alexandra Uittenbogerd, RMIT
Finn Upham, McGill University
J V, UPF Barcelona
Jose J. Valero-Mas, University of Alicante
Aneesh Vartakavi, Gracenote
Igor Vatolkin, TU Dortmund
Olga Vechtomova, University of Waterloo
Makarand Velankar, Cummins COE
Gissel Velarde, Consultant
Prateek Verma, Stanford University
Amruta Vidwans, Georgia Institute of Technology
Venkata S Viraraghavan, Tata Consultancy Services
Michael Vötter, Universität Innsbruck
Sanna Wager, Indiana University Bloomington
Changhong Wang, Queen Mary University of London
Chung-Che Wang, FinTech Center, NTU
Ju-Chiang Wang, ByteDance
Ziyu Wang, NYU Shanghai
Yu Wang, NYU
Kento Watanabe, National Institute of Advanced Industrial Science and Technology (AIST)

David M. Weigl, University of Music and Performing
Arts Vienna
Gordon Wichern, Mitsubishi Electric Research
Laboratories (MERL)
Elizabeth J Wilson, Queen Mary University of London
Scott Wisdom, Google
Daniel Wolff, Institut de Recherche et Coordination
Acoustique/Musique (IRCAM)
Minz Won, Universitat Pompeu Fabra
Kyle J. Worrall, University of York
Chih-Wei Wu, Netflix, Inc.
Shih-Lun Wu, National Taiwan University
Bruna Wundervald, Maynooth University
Anna Xambó, De Montfort University
Yujia Yan, University of Rochester
Luwei Yang, Alibaba Group
Furkan Yesiler, Universitat Pompeu Fabra
Sangeon Yong, KAIST
Frank Zalkow, International Audio Laboratories
Erlangen
Mickael Zehren, Umeå University
Shuo Zhang, Bose
Kejun Zhang, Zhejiang University
Weiwei Zhang, Dalian Maritime University
Yichi Zhang, University of Rochester
Yudong Zhao, Queen Mary University of London
Pablo Zinemanas, Universitat Pompeu Fabra

Preface

Welcome to ISMIR 2021, the 22nd International Society for Music Information Retrieval Conference. ISMIR is the world’s leading research forum on processing, searching, organizing, and accessing music-related data. Our community reflects a diversity of scientific disciplines, seniority levels, professional affiliations, and cultural backgrounds. Our goal is in fostering and stimulating this diversity, leading to better science and better music services. Due to the COVID-19 pandemic, the 22nd ISMIR conference became the second ISMIR to be fully virtual. While this posed unique challenges, we also took this as an opportunity to expand the reach of ISMIR and improve the inclusivity of the conference. The organizing team, who came together from all over the world to ensure the success of this event, welcomes you to ISMIR 2021.

I. Scientific Program

ISMIR 2021 Scientific Program comprised two Keynote talks, one WiMIR Keynote talk, 6 tutorials and 104 papers. A total of 278 abstracts were registered of which 258 were submitted as full papers eligible for review. In keeping with the practices of the previous years, a two-tier double-blind review process was conducted involving a total of 298 reviewers and 81 meta-reviewers. Each paper was assigned to a single meta-reviewer and 4 reviewers, to ensure that each would eventually receive at least 3 completed reviews, accounting for the foreseen limited availability of some reviewers. Meta-reviewers were also instructed to complete a full review of each of their assigned papers, in addition to the final meta-review summarizing the individual reviews. Each meta-reviewer was responsible for between 2 and 5 papers, and each reviewer was responsible for no more than 4 papers. The initial reviewing phase was followed by a discussion period, in which reviewers and meta-reviewers could discuss and revise their assessments of each paper. Meta-reviewers were then instructed to summarize the discussion and reviews in the final report. The Scientific Program Chairs (SPC) finally rendered decisions on each paper. The SPC would like to express their thanks to the ISMIR community of reviewers for their wholehearted support to this critical aspect of a successful ISMIR technical program.

Table 1 summarizes the submitted papers by subject area together with the corresponding accepted proportion. Figure 1 illustrates the number of papers accepted with at least one contributing author from each region. Geographic affiliations were inferred from self-reported author affiliations and email addresses. Finally, Table 2 summarizes the publication statistics over the 22-year-history of the conference.

Table 1: Papers submitted and accepted by subject area

Subject Area	Submitted	Accepted	Accept %
Applications	25	7	28
Domain knowledge	53	19	36
Evaluation, datasets and reproducibility	18	10	55
Human-centered MIR	13	7	54
MIR fundamentals and methodology	14	4	28
MIR tasks	88	29	33
Musical features and properties	43	26	60
Philosophical and ethical discussions	4	2	50
Total	258	104	40.3

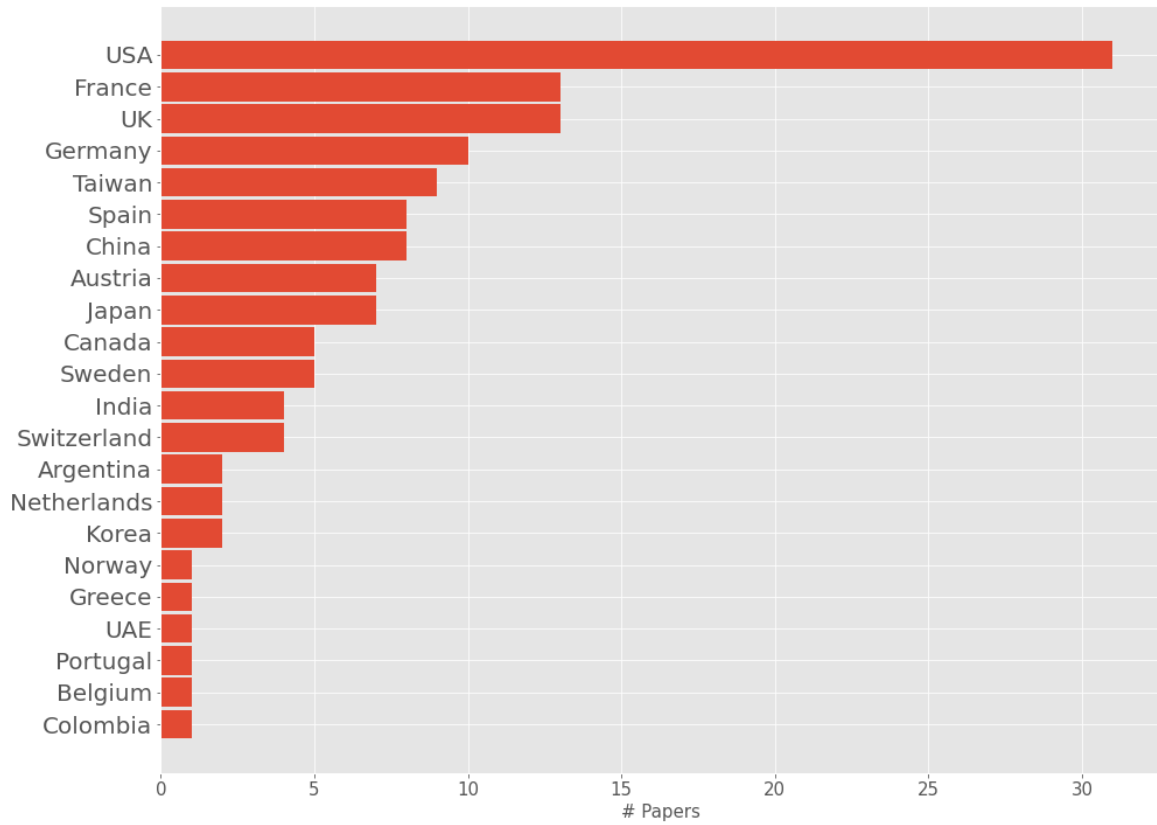


Figure 1: Number of papers accepted with at least one contributing author from each region

Table 2: Summary of publication statistics over the 22-year-history of the ISMIR conference

Year	Location	Oral	Poster	Total	Authors	Unique Authors	Authors / Paper	Unique Authors / Paper
2000	Plymouth	19	16	35	68	63	1.9	1.8
2001	Indiana	25	16	41	100	86	2.4	2.1
2002	Paris	35	22	57	129	117	2.3	2.1
2003	Baltimore	26	24	50	132	111	2.6	2.2
2004	Barcelona	61	44	105	252	214	2.4	2.0
2005	London	57	57	114	316	233	2.8	2.0
2006	Victoria	59	36	95	246	198	2.6	2.1
2007	Vienna	62	65	127	361	267	2.8	2.1
2008	Philadelphia	24	105	105	296	253	2.8	2.4
2009	Kobe	38	85	123	375	292	3.0	2.4
2010	Utrecht	24	86	110	314	263	2.0	2.4
2011	Miami	36	97	133	395	322	3.0	2.4

2012	Porto	36	65	101	324	264	3.2	2.6
2013	Curitiba	31	67	98	395	236	3.0	2.4
2014	Taipei	33	73	106	343	271	3.2	2.6
2015	Málaga	24	90	114	370	296	3.2	2.6
2016	New York	25	88	113	341	270	3.0	2.4
2017	Suzhou	24	73	97	324	248	3.3	2.6
2018	Paris			104	337	265	3.2	2.5
2019	Delft			114	390	315	3.4	2.8
2020	Montréal / Virtual			115	426	343	3.7	3.0
2021	Virtual			104	334	269	3.2	2.6

Special Call for Papers: Cultural Diversity in MIR

Music is often considered a universal language, yet different cultures have created diverse music traditions and colorful artifacts. This year, the conference organizers wanted to promote the cultural diversity of the ISMIR community and its research. To this end, the ISMIR2021 Call for Papers included a special call for papers on “Cultural Diversity in MIR”. This year's Scientific Program Chairs – Zhiyao Duan (University of Rochester, USA), Juhun Nam (KAIST, South Korea), Preeti Rao (IIT Bombay, India), and Peter van Kranenburg (Meertens Institute, The Netherlands) – organized the call with a focus on the twin topics of non-Western music and cross-cultural studies. To submit to this call, authors had expressed their intent during submission.

The submissions to the Special Call underwent the same review process as the papers in the main track, with the same number of reviews and a similar number of bids per submission, and with meta-reviewers who were carefully chosen to oversee the review process. In all, 44 papers were submitted to this call, of which 11 were accepted and verified by the SPC to match the topics of the call. An additional 3 papers were also accepted to the main track as they were verified by the SPC not to match the special call. Table 3 shows the distribution of submitted papers across subject areas together with the proportion of accepted papers in each for this special call.

Table 3: Special Call for Papers on “Cultural Diversity in MIR”: Papers submitted and accepted by subject area

Subject Area	Submitted	Accepted	Accept %
Applications	3	0	0
Domain Knowledge	12	3	25
Evaluation, datasets and reproducibility	2	1	50
Human-centered MIR	1	0	0
MIR tasks	14	2	14
Musical features and properties	7	3	43
Philosophical and ethical discussions	2	2	100
Total	41	11	26.8

Figure 2 depicts the number of Special Call papers accepted with at least one contributing author from each of the specified regions of the world. The geographic affiliations were inferred from self-reported author affiliations and email addresses. We note a wide representation of countries with at least a few international collaborations.

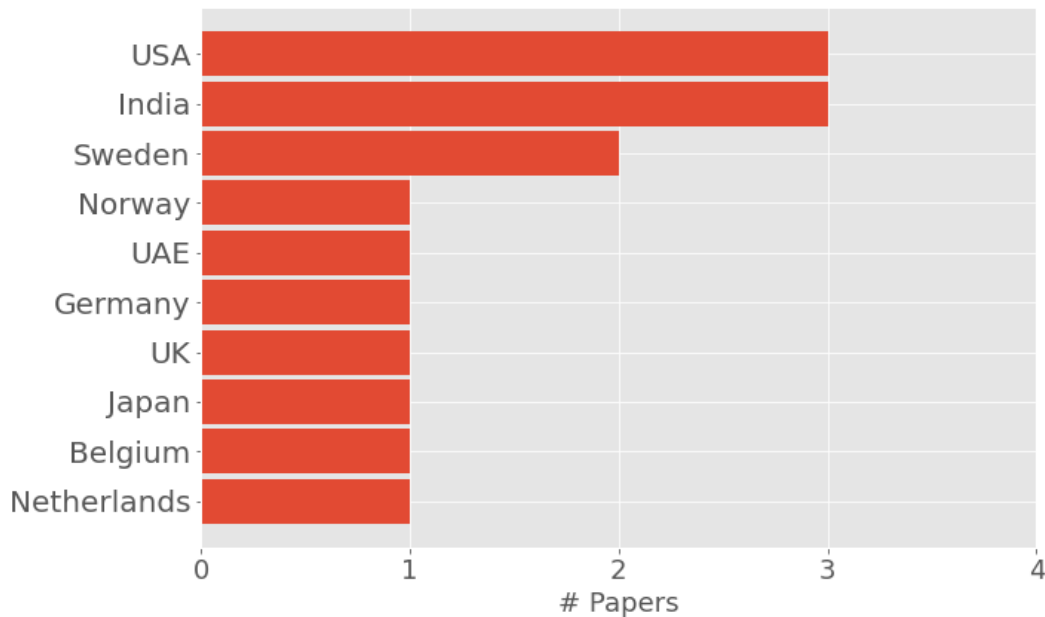


Figure 2: Number of papers in the special call accepted with at least one contributing author from each region

Best Paper Awards

Best paper candidates were selected from the 104 accepted papers. The SPC first short-listed 15 papers based on reviewers’ and meta-reviewers’ nominations as well as the paper review scores. Of these, the SPC nominated 8 paper candidates under three categories: the Best Paper (3 candidates), the Best Student Paper (3 candidates) and the Best Special Call Paper (2 candidates), based on their own judgement of the paper attributes as well as the detailed reviewer comments.

Best Paper Award Nominations

Filip Korzeniowski, Sergio Oramas, and Fabien Gouyon, Artist Similarity Using Graph Neural Networks

Hugo F. Flores Garcia, Aldo Aguilar, Ethan Manilow, and Bryan Pardo, Leveraging Hierarchical Structures for Few-Shot Musical Instrument Recognition

Rodrigo Castellon, Chris Donahue, and Percy Liang, Codified Audio Language Modeling Learns Useful Representations for Music Information Retrieval

Best Student Paper Award Nominations

Minz Won, Justin Salamon, Nicholas J. Bryan, Gautham Mysore, and Xavier Serra, Emotion Embedding Spaces for Matching Music to Stories

Daniel Yang and Timothy Tsai, Composer Classification with Cross-Modal Transfer Learning and Musically-Informed Augmentation

Harin Lee, Frank Höger, Marc Schönwiesner, Minsu Park, and Nori Jacoby, Cross-cultural Mood Perception in Pop Songs and its Alignment with Mood Detection Algorithms

Best Special Call Paper Award Nominations

Rujing Huang, Bob L. T. Sturm, Andre Holzapfel, De-centering the West: East Asian Philosophies and the Ethics of Applying Artificial Intelligence to Music

Rohit M A, Amitrajit Bhattacharjee, Preeti Rao, Four-way Classification of Tabla Strokes with Models Adapted from Automatic Drum Transcription

The final selections were made by specially appointed judges drawn from experienced researchers for each category. The following papers received awards:

Best Paper Award

Hugo F. Flores Garcia, Aldo Aguilar, Ethan Manilow, and Bryan Pardo, Leveraging Hierarchical Structures for Few-Shot Musical Instrument Recognition

Best Student Paper Award

Minz Won, Justin Salamon, Nicholas J. Bryan, Gautham Mysore, and Xavier Serra, Emotion Embedding Spaces for Matching Music to Stories

Best Special Call Paper Award

Rujing Huang, Bob L. T. Sturm, and Andre Holzapfel, De-centering the West: East Asian Philosophies and the Ethics of Applying Artificial Intelligence to Music.

During the conference a poll was set up to vote for the best poster and best video presentation and the paper that received the most votes from participants also received popular choice awards.

Best Poster Presentation (by popular vote)

Filip Korzeniowski, Sergio Oramas, and Fabien Gouyon, Artist Similarity Using Graph Neural Networks.

Best Video Presentation (by popular vote)

Laure Pr  tet, Ga  l Richard, and Geoffroy Peeters, Is There a “Language of Music-Video Clips” ? A Qualitative and Quantitative Study

Best Reviewer Awards

Based on the scores provided by meta-reviewers on the quality of individual reviews, in relation to the number of papers reviewed by each reviewer, the SPC selected a total of 16 awardees listed below:

Marcelo Caetano
Andrew Demetriou
Christoph Finkensiep
Fran  ois Germain
Daniel Harasim
Ben Hayes
Chris Hubbles
Yaolong Ju

Ilaria Manco
Sandy Manolios
Eric Nichols
Emilia Parada-Cabaleiro
Verena Praher
Katharina Prinz
Rafael Caro Repetto
Maximilian Schmitt

II. Diversity & Inclusion (D&I)

The ISMIR 2021 conference took a broad view of Diversity and Inclusion (D&I). Under the leadership of the conference D&I Chairs, in collaboration with the organizing team at large, ISMIR 2021 offered a variety of initiatives intended to make the conference a positive, welcoming, and supportive environment for a diverse range of presenters and attendees. Notably, this year’s virtual conference format, combined with generous sponsor support, enabled an unprecedented level of financial support to cover registration and childcare costs. Registration waivers were made available to students, women and other underrepresented minorities in MIR, attendees from low-income countries, presenters in the “New-to-ISMIR” late-breaking/demo track, and unaffiliated attendees. All attendees were additionally eligible to apply for childcare grants. The ISMIR 2021 organizers also worked together to write a number of blog posts aimed to decrease

barriers for participation in the MIR research community, for example, by offering insights into preparing and reviewing scientific submissions. Finally, the ISMIR conference Code of Conduct remained in place for this year's virtual format.

Newcomer Initiatives

Coming to a new conference for the first time can be intimidating and overwhelming. At ISMIR this year, the Newcomer Initiatives Chairs have drawn on the past experiences of the MIR community to provide increased support to newcomers. The initiatives planned for the conference included a pair of special sessions on "Getting the most out of ISMIR 2021", which follow up on a community survey and blog post on the subject published before the conference. The sessions were hosted by Newcomer Initiatives Chairs Nick Gang (Apple) and Elona Shatri (Queen Mary University of London). Another new initiative was the creation of Newcomer Squads, which connected ISMIR veterans with groups of newcomers to answer questions, give advice, and offer support over the course of the conference week.

Women in Music Information Retrieval (WiMIR)

Women in Music Information Retrieval (WiMIR) is a group of people dedicated to promoting the role of, and increasing opportunities for, women in the MIR field. WiMIR's initiatives started as informal gatherings around breakfast or lunch during ISMIR conferences (2011–2014), and moved to formal WiMIR events included in the conference program (2015–today) garnering a high turnout of both women and allies. These events provide occasions for people to network and to discuss several important issues ranging from mentorship and conference support, to improving the representation of women and, more broadly, diversity in the community. In 2018, WiMIR started hosting its own workshop as a satellite event, in which attendees of all genders participated. These workshops aim to offer participants an opportunity for networking, put the spotlight on technical work done by women in the field, and foster collaboration between women and allies by proposing group work led by project guides to try to solve small research problems or to undertake new research projects that could lead to longer-term collaborations. The ISMIR 2021 D&I Chairs gratefully acknowledge the support of this year's WiMIR sponsors, whose contributions support women in the field as well as the broader D&I efforts of this year's conference.

WiMIR Plenary and Special Sessions

The ISMIR 2021 conference continued the tradition of including a WiMIR plenary session in the main conference program. This year's WiMIR plenary session featured a keynote talk by Laurel Smith Pardue. The ISMIR 2021 conference also included a number of special WiMIR meetup sessions throughout the week, where attendees could engage in discussion with invited notable women in the field.

WiMIR Special Session 1: Cheng-Zhi Anna Huang (Magenta / Google Brain / Université de Montréal / Mila)

WiMIR Special Session 2: Emma Azelborn (iZotope, Inc.)

WiMIR Special Session 3: Xiao Hu (University of Hong Kong)

WiMIR Special Session 4: Katerina Kosta (ByteDance / TikTok)

Broadening Diversity & Inclusion in MIR

On July 7 2021, the D&I Chairs organized an "ISMIR Diversity & Inclusion Summit" at which invited participants discussed four topics: Clarifying the goal of D&I efforts at ISMIR; charting the future of WiMIR; maintaining ISMIR conference accessibility post-COVID; and ethical considerations for ISMIR and WiMIR sponsors. Many possible actions were considered at the summit, including using resources to nurture leadership within and provide support to marginalized and underrepresented groups at ISMIR; altering the organizational structure of WiMIR; continuing hybrid (i.e., physical/virtual) conference formats; and formalising guidelines and benefits of corporate sponsorship. At ISMIR 2021, these conversations continued at two special sessions on the themes of Broadening D&I and Local Initiatives. Each featured a Q&A with a panel of guests and discussion with the audience.

Diversity & Inclusion Special Session: Broadening D&I

Moderator: Blair Kaneshiro (Stanford University, USA)

Panelists: Johanna Devaney (Brooklyn College and the Graduate Center, CUNY, USA), Zhiyao Duan (University of Rochester, USA), Katherine M. Kinnaird (Smith College, USA), Douglas Turnbull (Ithaca College, USA)

Diversity & Inclusion Special Session: Local Initiatives

Moderator: Jordan B. L. Smith (TikTok / ByteDance, UK)

Panelists: Sakinat Folorunso (Olabisi Onabanjo University, Nigeria), Meinard Müller (International Audio Laboratories Erlangen, Germany), Elio Quinton (Universal Music Group, UK), Anja Volk (Utrecht University, The Netherlands)

III. Special Sessions

Similar to last year, the Scientific Program Chairs organized six special sessions on trending topics at ISMIR 2021. Brief introductions and session information is provided below:

Special Session 1: MIR for Human Health and Potential Panel

The MIR for Human Health and Potential Panel brings together researchers from music, cognitive psychology, neuroscience, mathematics, and computer science to continue discussion of a topic which began at ISMIR2014. This topic has become increasingly important, especially during an unprecedented global pandemic which has significant impacts on people's health, wellbeing and learning, as their interactions with others are restricted due to social distancing measures.

The COVID-19 pandemic has made music more important as a medium to connect people, to enhance both physical and mental health, and to advance human potential. Social distancing measures have greatly enhanced interest in eHealth and eLearning. Integrating music into digital health/learning technologies helps to make music interventions accessible, scalable, and personalized. Finally, these technological innovations offer effective interventions for health, wellbeing and learning that are safe and non-invasive.

Research at the intersection of neuroscience, medicine, the science of learning, and MIR has the potential to reveal new insights into individual variability and personalized interventions; while mobile systems for collecting physiological data point to promising avenues for ecologically valid studies, and even socially distanced data collection in the home.

Moderator: Ye Wang (National University of Singapore, Singapore)

Panelists: Frank Russo (Ryerson University, Canada), Elaine Chew (CNRS – STMS (IRCAM), France), Gus Xia (NYU Shanghai, China), Blair Kaneshiro (Stanford University, USA)

Special Session 2: IMS Digital Musicology Study Group

This meeting of the International Musicological Society's Digital Musicology Study Group is open to anyone interested in applying computational methods to musicological questions. The meeting will consist of a series of short presentations about ongoing research projects as well as a general discussion about the future activities of the group.

Moderators: Johanna Devaney (Brooklyn College and the Graduate Center, CUNY, USA), Frans Wiering (Utrecht University, The Netherlands)

Special Session 3: Promoting Cultural Diversity in MIR Research

Promoting cultural diversity in MIR research has been recognized by the MIR community as an important task. ISMIR 2021 is proud of having organized a special call-for-papers on cultural diversity, with an outcome of eleven papers accepted. The goal of this session is to reflect on this effort and to discuss how to promote cultural diversity in MIR research in a sustainable manner.

Moderator: Scientific Program Co-Chairs

Panelists: Magdalena Fuentes (New York University, USA), Xiao Hu (The University of Hong Kong, Hong Kong), Patrick Savage (Keio University SFC, Japan), Li Su (Academia Sinica, Taiwan)

Special Session 4: Computational Creativity for Music

This panel on Computational Creativity for Music will begin by discussing definitions of AI, CC, and what panelists perceive as important differences between the two. Portrait XO will reflect on an artist's experience of incorporating algorithms into the creative process – what works, what doesn't. We will address evaluation, especially work the academics have done to address how we could/should evaluate computational systems and their outputs. Is it sufficient for a machine learning paper to rely solely on metrics such as loss/cross-entropy for evaluation? We will get the panelists' points of view on this, and provide a useful reading list for ISMIR authors aiming to excel at evaluating their creative MIR research.

Moderator: Tom Collins (University of York, UK, Music Artificial Intelligence Algorithms, Inc., USA)

Panelists: Anna Jordanous (University of Kent, UK), Dan Ventura (Brigham Young University, USA), Geraint Wiggins (VUB, Belgium / Queen Mary University of London, UK), Portrait XO (Artist)

Special Session 5: MIR for Music Education

Computer technology, and MIR in particular, has inspired many innovations in music education, including pitch recognition for singers, score alignment for performance evaluation and AI assistants for student composers. We will discuss the potential of MIR for the future of music education as well as how the needs of music educators can guide MIR research.

Moderator: Roger B. Dannenberg (Carnegie Mellon University, USA)

Panelists: Zhiyao Duan (University of Rochester, USA), Anssi Klapuri (Yousician, Finland), Alexander Lerch (Georgia Institute of Technology, USA), Daniel Ray (Muse Group, UK)

Special Session 6: Ethical Issues in Music Ai – Perspectives on authorship with creative artificial intelligence in music

In the wake of all the exciting opportunities that creative Ai technology provides artists, a more contentious issue is emerging: that of authorship. Who can or should be denominated as the author of a work created or co-created alongside Ai systems? What is the role of the cultural commons from which ML systems draw their creative energies, especially in situations involving precarious minorities or cultural heritage groups vulnerable to exploitation? How do copyright or related neighbouring rights approach authorship and ownership in these contexts? How can revenue streams generated by the creative Ai works be allocated in a fair and sustainable manner? And finally, what consequences will these choices have for the creative artists, for the industry and for the society at large?

In this session, we will not provide easy answers to these questions, but start exploring them by mapping out authors, owners and stakeholders in various contexts of music production, performance and consumption. Through the identification of the networks of influence, power and exclusion, we can start exposing the ethical ecosystems and the shifting contours of authorship in the Ai music industry. We aim to conduct parts of this session in a workshop format, facilitating an open discussion with the audience.

Moderators: Andre Holzapfel (KTH Royal Institute of Technology, Sweden), Petra Pauliina Jääskeläinen (KTH Royal Institute of Technology, Sweden), Anna-Kaisa Kaila (KTH Royal Institute of Technology, Sweden), Sertan Şentürk (Kobalt Music Group)

IV. Late Breaking/Demo Session

This year's Late-breaking/Demo (LBD) session introduced two changes to previous years' iterations of the event. First, taking advantage of the remote conference format where space is no longer a constraint, there was no upper limit to the number of submissions accepted, providing more opportunities for people to showcase their late-breaking work. Second, a New-to-ISMIR special track was introduced wherein student authors who are looking to join the community for the first time were encouraged to submit their work and receive extra mentoring. The mentoring process included a light review process where 'mentors' assigned to each paper gave constructive feedback, thus giving participants a taste of peer-review and enabling them to refine their camera-ready submissions and facilitate effective presentations. In total, we received 38 regular track and 20 special track submissions, respectively. Sixteen ISMIR reviewers served as mentors in the special track, allowing each mentee to receive up to two reviews with detailed written feedback. With these changes, we hope to invite more people to our community and inspire future innovations in the LBD format.

V. Music

ISMIR 2021 Music Program could be entitled "A festival of visual music." We received an overwhelming amount of works in the category of creative visuals. The unexpected high number of submissions (46) made it very hard to select the pieces to be presented. However, we think we had a great selection of works — not all of them related directly to MIR — that constituted an excellent musical counterpart to the scientific program. Split into four short online concerts — two of less than 30 minutes each, another two around 40 minutes each — the program only featured musical works with a visual counterpart, be it creative visuals, networked ensembles, or (recordings of) live performances. We realized that works that had no video (e.g. acousmatic music) should not be presented in this (online) format for reasons that have

nothing to do with the (sometimes) great quality of the music — it simply would make it very hard to fully appreciate these works. Here is a list of music pieces that were presented in ISMIR 2021:

Concert 1

Symphony in Blue 2.0, Istanbul Coding Ensemble with Jerfi Aji, piano

Xeno, Enrico Dorigatti

Three Tunes from the Ai Frontiers, Bob L. T. Sturm

toy_5, Eric Lemmon

History Has Stopped at 2021, Vanissa Law

Concert 2

Golden Cuttlefish, Timothy Moyers

Topos, Giuseppe Desiato

Things I Have Seen In My Dreams, João Pedro Oliveira

coalesce;, Tamara E Ray

Concert 3

Lullaby for Stepanakert, Joseph Bohigian & Ensemble Decipher

Butterfly Garden, Donya Quick

Forme Cangianti, Fabio Morreale

Quartet, Ted Moore

Inkblot, Serge Bulat

Music for Virtual Togetherness, Poli Étnico Choir

Concert 4

Rooftops, Modality

String Quartet, Hendrik Vincent Koops

Apocalypse – Future, Oregon Electronic Device Orchestra

The Seals: Networked ensemble of pre-recorded live performance with creative AI assisted visuals, The Seals

Horizon, Mojtaba Heydari, Frank Cwitkowitz

Music was alive and kicking at virtual ISMIR 2021!

VI. Industry Sessions

The Sponsorship Program Chairs organized eight industry sessions over three themes. Brief introductions and session information are provided below:

Industry Presentations

Over four sessions, the Sponsors of ISMIR 2021 showcased the research, engineering and product work conducted in their companies with talks, demos, and (for platinum sponsors) Q&As.

Participating Sponsors: Spotify, Sony, Adobe, Bytedance/Tiktok, Pandora, Yamaha, Deezer, Dolby, Izotope, Orfium, Utopia Music

Industry Panel: MIR Technologies Across Cultures

Users of MIR technologies come from all around the world. Technologies intended for one sonic landscape sometimes generalize to others. Sometimes, they require additional patience and research to adapt to a new user base. In this forum,

panelists from audio industries and academia discussed the exciting opportunities, challenges and lessons learned that come from implementing audio technologies across users with very different needs and sonic cultures.

Moderator: Gus Xia (NYU Shanghai (CST))

Panelists: Ajay Kapur (California Institute of the Arts), Rujing Huang (University of Hong Kong), Lamtham “Hanoi” Hantrakul (ByteDance/TikTok), Oriol “Uri” Nieto (Adobe), Akira Maezawa (Yamaha)

Industry Panel: MIR Technologies and Education

Technology plays a key role in education. This panel will dive into education in its broadest sense, from audio technologies that directly improve the learning process in music or speech related activities (e.g. learning an instrument, learning to speak a language), to technologies that aim to guide and educate users in a sound related endeavour (e.g. music production, video editing, cultivating interest in a new musical style) as well as health and well-being.

Moderator: Xavier Serra (Universitat Pompeu Fabra)

Panelists: Cynthia Liem (Delft University of Technology), Anja Volk (Utrecht University), François Pachet (Spotify), Fabien Gouyon (Pandora)

Masterclass: CV Review

ISMIR Masterclasses are intended to familiarize current students and recent graduates with interviews and processes they may encounter when looking for MIR-style roles in industry. This session focused on CV and resumes for audio, MIR and machine learning related roles. We go through tips and tricks for making your resume stand out with Chris Bakes, a Campus Recruiter at TikTok and previously University Recruiter at Facebook AI.

Panelists: Chris Bakes (Campus Recruiter, TikTok/ByteDance), Lamtham “Hanoi” Hantrakul (Research Scientist, TikTok/ByteDance)

Masterclass: Systems Design Interview

ISMIR Masterclasses are intended to familiarize current students and recent graduates with interviews and processes they may encounter when looking for MIR-style roles in industry. This session focused on a Systems Design Interview based on MIR technologies. This type of interview is distinct from the Technical Coding Interview or ML Interview many attendees may be familiar with. Research code is not Production code. Systems Design tests awareness of the scalability and maintainability of a solution and the technologies used to robustly achieve the demands of a product. This session featured a deconstructed mock interview and industry best practices.

Panelists: Peter Sobot (Staff Software Engineer, Spotify), Lamtham “Hanoi” Hantrakul (Research Scientist, TikTok/ByteDance), Sertan Şentürk (Lead Data Scientist, Kobalt Music Group)

VII. Lab Showcase

This year, we introduced a new event called the Lab Showcase to the ISMIR conference. Academic labs focusing on MIR research were invited to showcase their lab at virtual booths. There were 33 lab participants across 17 different countries, showcasing their labs through live sessions and lab introduction materials at the virtual booths. Participating labs were also invited to post vacancies together with sponsors in a Job Board. By introducing the Lab Showcase, we hope to give a perspective on MIR research world-wide, enable connections between institutions and prospective students or collaborators, and simply make it easier to reach out to and learn more about various labs across the globe.

VIII. Satellite Events

In addition to the main conference, two satellite events were offered to participants:

1. The Music Demixing (MDX) Challenge, November 12, 2021 (online)
2. 2nd Workshop on NLP for Music and Spoken Audio (NLP4MuSA 2021), November 12, 2021 (online)

IX. Acknowledgements

We are happy to present to you the proceedings of ISMIR 2021. The conference program was made possible thanks to the hard work of many people, including the ISMIR 2021 conference chairs, the administrative staff at Georgia Tech School of Music, ISMIR Board members, volunteers, and the many reviewers and meta-reviewers from the program committee.

We would also like to thank our sponsors, whose contributions made this conference possible:

Platinum sponsors

- Sony CSL
- Spotify

Gold sponsors

- Adobe
- ByteDance
- Pandora
- Yamaha

Silver sponsors

- Ableton
- ACRCcloud
- Avid
- Apple
- Cochlear
- Deezer
- Dolby
- iZotope
- Orfium
- Steinberg
- Utopia
- Audible Magic

We would like to thank the sponsors that explicitly chose to sponsor WiMIR, its grants, and its initiatives:

Patrons

- Spotify

Contributors

- Deezer
- iZotope
- Ableton
- Pandora
- Adobe

Supporters

- Steinberg

ISMIR 2021 would not have been possible without the exceptional contributions of our community in response to our call for participation. The biggest acknowledgment goes to you, the presenters and the participants.

Zhiyao Duan

Juhan Nam

Preeti Rao

Peter Van Kranenburg

Scientific Program Chairs

Jin Ha Lee

Alexander Lerch

General Chairs

Contents

Keynote Talks	1
Jukebox and Musenet: Generating Raw Audio and MIDI Music <i>Christine McLeavey</i>	3
Acoustic Singularities of Some Songs of Oral Tradition <i>Michèle Castellengo</i>	4
Improving Diversity and Inclusivity Through Action <i>Laurel Smith Pardue</i>	5
Tutorials	7
Tempo, Beat, and Downbeat Estimation <i>Matthew Davies, Sebastian Bock and Magdalena Fuentes</i>	9
Designing Generative Models for Interactive Co-creation <i>Anna Huang, Jon Gillick and Chris Donahue</i>	10
Scales, Chords, and Cadences: Practical Music Theory for MIR Researchers <i>Johanna Devaney, David Sears and Daniel Shanahan</i>	11
Music Classification: Beyond Supervised Learning, Towards Real-world Applications <i>Keunwoo Choi, Minz Won and Janne Spijkervet</i>	13
Programming MIR Baselines from Scratch: Three Case Studies <i>Rachel Bittner, Mark Cartwright and Ethan Manilow</i>	14
Teaching Music Information Retrieval <i>George Tzanetakis</i>	15
Papers	17
Four-way Classification of Tabla Strokes with Models Adapted from Automatic Drum Transcription <i>Rohit M A, Amitrajit Bhattacharjee, Preeti Rao</i>	19
A Contextual Latent Space Model: Subsequence Modulation in Melodic Sequence <i>Taketo Akama</i>	27
OMR-assisted Transcription: A Case Study with Early prints <i>María Alfaro-Contreras, David Rizo, Jose M. Inesta, Jorge Calvo-Zaragoza</i>	35
Deeper Convolutional Neural Networks and Broad Augmentation Policies Improve Performance in Musical Key Estimation <i>Stefan A Baumann</i>	42
The Music Performance Markup Format and Ecosystem <i>Axel Berndt</i>	50
Identification of Rhythm Guitar Sections in Symbolic Tablatures <i>Louis Bigo, David Regnier, Nicolas Martin</i>	58
On-Line Audio-to-Lyrics Alignment Based on a Reference Performance <i>Charles Brazier, Gerhard Widmer</i>	66
Visualizing Intertextual Form with Arc Diagrams: Contour and Schema-based Methods <i>Aaron Carter-Enyi, Gilad Rabinovitch, Nathaniel Condit-Schultz</i>	74
Unsupervised Domain Adaptation for Document Analysis of Music Score Images <i>Francisco J. Castellanos, Antonio-Javier Gallego, Jorge Calvo-Zaragoza</i>	81
Codified Audio Language Modeling Learns Useful Representations for Music Information Retrieval <i>Rodrigo Castellon, Chris Donahue, Percy Liang</i>	88
Variable-Length Music Score Infilling via XLNet and Musically Specialized Positional Encoding <i>Chin-Jui Chang, Chun-Yi Lee, Yi-Hsuan Yang</i>	97

SurpriseNet: Melody Harmonization Conditioning on User-controlled Surprise Contours <i>Yi-Wei Chen, Hung-Shin Lee, Yen-Hsing Chen, Hsin-Min Wang</i>	105
Semi-supervised Violin Fingering Generation Using Variational Autoencoders <i>Vincent K.M. Cheung, Hsuan-Kai Kao, Li Su</i>	113
Listen, Read, and Identify: Multimodal Singing Language Identification of Music <i>Keunwoo Choi, Yuxuan Wang</i>	121
On Perceived Emotion in Expressive Piano Performance: Further Experimental Evidence for the Relevance of Mid-level Perceptual Features <i>Shreyan Chowdhury, Gerhard Widmer</i>	128
Cosine Contours: a Multipurpose Representation for Melodies <i>Bas Cornelissen, Willem Zuidema, John Ashley Burgoyne</i>	135
Controllable Deep Melody Generation via Hierarchical Music Structure Representation <i>Shuqi Dai, Zeyu Jin, Celso Gomes, Roger Dannenberg</i>	143
MSTRE-Net: Multistreaming Acoustic Modeling for Automatic Lyrics Transcription <i>Emir Demirel, Sven Ahlbäck, Simon Dixon</i>	151
Towards Automatic Instrumentation by Learning to Separate Parts in Symbolic Multitrack Music <i>Hao-Wen Dong, Chris Donahue, Taylor Berg-Kirkpatrick, Julian Mcauley</i>	159
An Empirical Evaluation of End-to-End Polyphonic Optical Music Recognition <i>Sachinda Edirisooriya, Hao-Wen Dong, Julian Mcauley, Taylor Berg-Kirkpatrick</i>	167
A Hardanger Fiddle Dataset with Performances Spanning Emotional Expressions and Annotations Aligned using Image Registration <i>Anders Elowsson, Olivier Lartillot</i>	174
Building the MetaMIDI Dataset: Linking Symbolic and Audio Musical Data <i>Jeffrey Ens, Philippe Pasquier</i>	182
Modeling and Inferring Proto-Voice Structure in Free Polyphony <i>Christoph Finkensiep, Martin A Rohrmeier</i>	189
PKSpell: Data-Driven Pitch Spelling and Key Signature Estimation <i>Francesco Foscarin, Nicolas Audebert, Raphael Fournier-S’Niehotta</i>	197
Filosax: A Dataset of Annotated Jazz Saxophone Recordings <i>Dave Foster, Simon Dixon</i>	205
An Interpretable Music Similarity Measure Based on Path Interestingness <i>Giovanni Gabbolini, Derek Bridge</i>	213
Leveraging Hierarchical Structures for Few-Shot Musical Instrument Recognition <i>Hugo F Flores Garcia, Aldo Aguilar, Ethan Manilow, Bryan Pardo</i>	220
What if the ‘When’ Implies the ‘What’?: Human harmonic analysis datasets clarify the relative role of the separate steps in automatic tonal analysis <i>Mark R H Gotham, Rainer Kleinertz, Christof Weiss, Meinard Müller, Stephanie Klauk</i>	229
Let’s agree to disagree: Consensus Entropy Active Learning for Personalized Music Emotion Recognition <i>Juan S. Gómez-Cañón, Estefania Cano, Yi-Hsuan Yang, Perfecto Herrera, Emilia Gomez</i>	237
Sequence-to-Sequence Piano Transcription with Transformers <i>Curtis Hawthorne, Ian Simon, Rigel Swavely, Ethan Manilow, Jesse Engel</i>	246
Neural Waveshaping Synthesis <i>Ben Hayes, Charalampos Saitis, Gyorgy Fazekas</i>	254
A Semi-automated Workflow Paradigm for the Distributed Creation and Curation of Expert Annotations <i>Johannes Hentschel, Fabian C. Moss, Markus Neuwirth, Martin A Rohrmeier</i>	262
BeatNet: CRNN and Particle Filtering for Online Joint Beat, Downbeat and Meter Tracking <i>Mojtaba Heydari, Frank Cwitkowitz, Zhiyao Duan</i>	270

Joint Estimation of Note Values and Voices for Audio-to-Score Piano Transcription <i>Yuki Hiramatsu, Eita Nakamura, Kazuyoshi Yoshii</i>	278
Learning Note-to-note Affinity for Voice Segregation and Melody Line Identification of Symbolic Music Data <i>Yo-Wei Hsiao, Li Su</i>	285
VOCANO: A Note Transcription Framework for Singing Voice in Polyphonic Music <i>Jui-Yang Hsu, Li Su</i>	293
De-centering the West: East Asian Philosophies and the Ethics of Applying Artificial Intelligence to Music <i>Rujing Huang, Bob L. T. Sturm, Andre Holzapfel</i>	301
A Benchmarking Initiative for Audio-domain Music Generation using the FreeSound Loop Dataset <i>Tun Min Hung, Bo-Yu Chen, Yen Tung Yeh, Yi-Hsuan Yang</i>	310
EMOPIA: A Multi-Modal Pop Piano Dataset For Emotion Recognition and Emotion-based Music Generation <i>Hsiao-Tzu Hung, Joann Ching, Seungheon Doh, Nabin Kim, Juhan Nam, Yi-Hsuan Yang</i>	318
Piano Sheet Music Identification Using Marketplace Fingerprinting <i>Kevin Ji, Daniel Yang, Timothy Tsai</i>	326
Learning a Cross-domain Embedding Space of Vocal and Mixed Audio with a Structure-preserving Triplet Loss <i>Keunhyoung Kim, Jongpil Lee, Sangeun Kum, Juhan Nam</i>	334
Decoupling Magnitude and Phase Estimation with Deep ResUNet for Music Source Separation <i>Qiuqiang Kong, Yin Cao, Haohe Liu, Keunwoo Choi, Yuxuan Wang</i>	342
Artist Similarity Using Graph Neural Networks <i>Filip Korzeniowski, Sergio Oramas, Fabien Gouyon</i>	350
“Finding Home”: Understanding How Music Supports Listeners’ Mental Health through a Case Study of BTS <i>Jin Ha Lee, Arpita Bhattacharya, Ria Antony, Nicole Santero, Anh Le</i>	358
Cross-cultural Mood Perception in Pop Songs and its Alignment with Mood Detection Algorithms <i>Harin Lee, Frank Höger, Marc Schönwiesner, Minsu Park, Nori Jacoby</i>	366
Reconsidering Quantization in MIR <i>Jordan Lenchitz</i>	374
A Unified Model for Zero-shot Music Source Separation, Transcription and Synthesis <i>Liwei Lin, Gus Xia, Qiuqiang Kong, Junyan Jiang</i>	381
Pitch-Informed Instrument Assignment using a Deep Convolutional Network with Multiple Kernel Shapes <i>Carlos Lordelo, Emmanouil Benetos, Simon Dixon, Sven Ahlbäck</i>	389
SpecTNT: a Time-Frequency Transformer for Music Audio <i>Wei-Tsung Lu, Ju-Chiang Wang, Minz Won, Keunwoo Choi, Xuchen Song</i>	396
AugmentedNet: A Roman Numeral Analysis Network with Synthetic Training Examples and Additional Tonal Tasks <i>Néstor Nápoles López, Mark R H Gotham, Ichiro Fujinaga</i>	404
MINGUS: Melodic Improvisation Neural Generator Using Seq2Seq <i>Vincenzo Madaghiele, Pasquale Lisena, Raphael Troncy</i>	412
User-centered Evaluation of Lyrics-to-audio Alignment <i>Ninon Lizé Masclef, Andrea Vaglio, Manuel Moussallam</i>	420
Synthesizer Sound Matching with Differentiable DSP <i>Naotake Masuda, Daisuke Saito</i>	428
A Modular System for the Harmonic Analysis of Musical Scores using a Large Vocabulary <i>Andrew Mcleod, Martin A Rohrmeier</i>	435
A Deep Learning Method for Enforcing Coherence in Automatic Chord Recognition <i>Gianluca Micchi, Katerina Kosta, Gabriele Medeot, Pierre Chanquion</i>	443
Modeling Beat Uncertainty as a 2D Distribution of Period and Phase: a MIR Task Proposal <i>Martin A Miguel, Diego Fernandez Slezak</i>	452

A Case Study of Deep Enculturation and Sensorimotor Synchronization to Real Music <i>Olof Misgeld, Torbjörn L Gulz, Jūra Miniotaitė, Andre Holzapfel</i>	460
Symbolic Music Generation with Diffusion Models <i>Gautam Mittal, Jesse Engel, Curtis Hawthorne, Ian Simon</i>	468
Learning from Musical Feedback with Sonic the Hedgehog <i>Faraaz Nadeem</i>	476
DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis With GANs <i>Javier Nistal, Stefan Lattner, Gaël Richard</i>	484
Phase-Aware Joint Beat and Downbeat Estimation Based on Periodicity of Metrical Structure <i>Takehisa Oyama, Ryoto Ishizuka, Kazuyoshi Yoshii</i>	493
Agreement Among Human and Automated Transcriptions of Global Songs <i>Yuto Ozaki, John M McBride, Emmanouil Benetos, Peter Pfordresher, Joren Six, Adam Tierney, Polina Proutskova, Emi Sakai, Haruka Kondo, Haruno Fukatsu, Shinya Fujii, Patrick E. Savage</i>	500
Automatic Recognition of Texture in Renaissance Music <i>Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Bjorn W. Schuller, Markus Schedl</i>	509
Is Disentanglement enough? On Latent Representations for Controllable Music Generation <i>Ashis Pati, Alexander Lerch</i>	517
Pulse Clarity Metrics Developed from a Deep Learning Beat Tracking Model <i>Nicolás Pironio, Diego Fernandez Slezak, Martín A Miguel</i>	525
On the Veracity of Local, Model-agnostic Explanations in Audio Classification: Targeted Investigations with Adversarial Examples <i>Verena Praher, Katharina Prinz, Arthur Flexer, Gerhard Widmer</i>	531
Is there a “language of music-video clips” ? A qualitative and quantitative study <i>Laure Prétet, Gaël Richard, Geoffroy Peeters</i>	539
Tabla Gharana Recognition from Audio music recordings of Tabla Solo performances <i>Gowriprasad R, Venkatesh V, Hema A Murthy, R Aravind, Sri Rama Murty K</i>	547
Navigating noise: Modeling perceptual correlates of noise-related semantic timbre categories with audio features <i>Lindsey Reymore, Emmanuelle Beauvais-Lacasse, Bennett Smith, Stephen Mcadams</i>	555
Quantitative User Perceptions of Music Recommendation List Diversity <i>Kyle Robinson, Dan Brown</i>	562
A Formal Model of Extended Tonal Harmony <i>Martin A Rohrmeier, Fabian C. Moss</i>	569
CRASH: Raw Audio Score-based Generative Modeling for Controllable High-resolution Drum Sound Synthesis <i>Simon Rouard, Gaëtan Hadjeres</i>	579
Curriculum Learning for Imbalanced Classification in Large Vocabulary Automatic Chord Recognition <i>Luke O Rowe, George Tzanetakis</i>	586
Deep Embeddings and Section Fusion Improve Music Segmentation <i>Justin Salamon, Oriol Nieto, Nicholas J. Bryan</i>	594
Multi-Task Learning of Graph-based Inductive Representations of Music Content <i>Antonia Saravanou, Federico Tomasi, Rishabh Mehrotra, Mounia Lalmas</i>	602
DadaGP: A Dataset of Tokenized GuitarPro Songs for Sequence Models <i>Pedro Pereira Sarmiento, Adarsh Kumar, Cj Carr, Zack Zukowski, Mathieu Barthelet, Yi-Hsuan Yang</i>	610
Does Track Sequence in User-generated Playlists Matter? <i>Harald Victor Schweiger, Emilia Parada-Cabaleiro, Markus Schedl</i>	618
A Differentiable Cost Measure for Intonation Processing in Polyphonic Music <i>Simon J Schwär, Sebastian Rosenzweig, Meinard Müller</i>	626

Improving Music Performance Assessment With Contrastive Learning <i>Pavan M Seshadri, Alexander Lerch</i>	634
Tracing Affordance and Item Adoption on Music Streaming Platforms <i>Dougal Shakespeare, Camille Roth</i>	642
Computational Analysis and Modeling of Expressive Timing in Chopin’s Mazurkas <i>Zhengshan Shi</i>	650
Computational Analysis of Melodic Mode Switching in Raga Performance <i>Nithya Nadig Shikarpur, Asawari Keskar, Preeti Rao</i>	657
SinTra: Learning an Inspiration Model from a Single Multi-track Music Segment <i>Qingwei Song, Qiwei Sun, Dongsheng Guo, Haiyong Zheng</i>	665
Contrastive Learning of Musical Representations <i>Janne Spijkervet, John Ashley Burgoyne</i>	673
Musical Tempo Estimation Using a Multi-scale Network <i>Xiaoheng Sun, Qiqi He, Gao Yongwei, Wei Li</i>	682
On the Integration of Language Models into Sequence to Sequence Architectures for Handwritten Music Recognition <i>Pau Torras, Arnau Baró, Lei Kang, Alicia Fornés</i>	690
Kiite Cafe: A Web Service for Getting Together Virtually to Listen to Music <i>Kosetsu Tsukuda, Keisuke Ishida, Masahiro Hamasaki, Masataka Goto</i>	697
Toward an Understanding of Lyrics-viewing Behavior While Listening to Music on a Smartphone <i>Kosetsu Tsukuda, Masahiro Hamasaki, Masataka Goto</i>	705
The Words Remain the Same: Cover Detection with Lyrics Transcription <i>Andrea Vaglio, Romain Hennequin, Manuel Moussallam, Gael Richard</i>	714
MuseBERT: Pre-training Music Representation for Music Understanding and Controllable Generation <i>Ziyu Wang, Gus Xia</i>	722
Supervised Metric Learning For Music Structure Features <i>Ju-Chiang Wang, Jordan B. L. Smith, Wei-Tsung Lu, Xuchen Song</i>	730
Learning Long-term Music Representations via Hierarchical Contextual Constraints <i>Shiqi Wei, Gus Xia</i>	738
Learning Pitch-Class Representations from Score-Audio Pairs of Classical Music <i>Christof Weiss, Johannes Zeitler, Tim Zunner, Florian Schubert, Meinard Müller</i>	746
Training Deep Pitch-Class Representations With a Multi-Label CTC Loss <i>Christof Weiss, Geoffroy Peeters</i>	754
Audio Defect Detection in Music with Deep Networks <i>Daniel Wolff, Remi Mignot, Axel Roebel</i>	762
Semi-supervised Music Tagging Transformer <i>Minz Won, Keunwoo Choi, Xavier Serra</i>	769
Emotion Embedding Spaces for Matching Music to Stories <i>Minz Won, Justin Salamon, Nicholas J. Bryan, Gautham Mysore, Xavier Serra</i>	777
CollageNet: Fusing Arbitrary Melody and Accompaniment into a Coherent Song <i>Abudukelimu Wuerkaixi, Christodoulos Benetatos, Zhiyao Duan, Changshui Zhang</i>	786
Human-in-the-Loop Adaptation for Interactive Musical Beat Tracking <i>Kazuhiko Yamamoto</i>	794
Composer Classification With Cross-Modal Transfer Learning and Musically-Informed Augmentation <i>Daniel Yang, Timothy Tsai</i>	802
Aligning Unsynchronized Part Recordings to a Full Mix Using Iterative Subtractive Alignment <i>Daniel Yang, Kevin Ji, Timothy Tsai</i>	810

ADTOF: A Large Dataset of Non-synthetic Music for Automatic Drum Transcription	
<i>Mickael Zehren, Marco Alunno, Paolo Bientinesi</i>	818
Learn by Referencing: Towards Deep Metric Learning for Singing Assessment	
<i>Huan Zhang, Yiliang Jiang, Tao Jiang, Hu Peng</i>	825
AccoMontage: Accompaniment Arrangement via Phrase Selection and Style Transfer	
<i>Jingwei Zhao, Gus Xia</i>	833

Author Index	843
---------------------	------------

Keynote Talks

Keynote Talk - 1

Jukebox and MuseNet - Generating Raw Audio and MIDI Music

Christine McLeavey

Member of Technical Staff (Researcher), OpenAI
Portola Valley, California, USA

Abstract

Music generation is exciting both as a tool for augmenting human creativity, and as a domain for pushing the current capabilities of generative neural net models. OpenAI's MuseNet is a MIDI-based model able to generate music imitating hundreds of composers and styles. Composers such as Philip Glass have experimented with the model, and it has been used as a co-composing tool for works performed by the BBC Philharmonic, among others. Jukebox is a model that generates music with singing in the raw audio domain. Provided with written lyrics and an artist and genre to imitate, the model generates complete songs. This talk discusses both MuseNet and Jukebox in more depth, as well as some recent artistic collaborations.

Biography

Christine McLeavey is a research scientist at OpenAI where she created MuseNet and collaborated to create Jukebox. Groups such as the BBC Philharmonic and SF Symphony have performed pieces co-composed using MuseNet. Also a Juilliard-trained pianist and avid chamber musician, she is particularly interested in Human/AI musical collaborations. She holds a masters in neuroscience from Stanford, and a degree in physics from Princeton.

Keynote Talk - 2

Acoustic Singularities of Some Songs of Oral Tradition

Michèle Castellengo

Emeritus Research Director
CNRS, Sorbonne Université

Abstract

After studying the acoustic properties and sound qualities of several musical instruments (flute, organ) as well as those of European vocal techniques, I have directed my research towards the study of songs of oral tradition - thus without notation - for which problems of analysis and transcription arise, as well as difficulties due to our own listening references, fundamentally different from those of native musicians.

In the course of the presentation we will travel to Central Africa with the Aka Pygmies, to Taiwan with the Bunun, to Central Asia with the Mongols and their "diphonic" singing, and back to Europe with a religious polyphonic song of the Sardinians.

At the end of this journey we will show that the classical notions of musical acoustics: pitch, intensity and timbre rarely correspond, for the human listener, to independent physical parameters, and that it would be necessary to consider the apprehension of global forms coordinating these parameters to account for the musical listening.

Biography

After studying music and musicology, Michèle Castellengo joined Emile Leipp's laboratory of musical acoustics where she defended a thesis under his direction. In 1982, she joined the CNRS and became director of the laboratory (LAM). Her research focuses on the acoustics of flutes, the organ and the sung voice, and more generally on the perception of musical sounds. Strongly involved in the dissemination of musical knowledge to musicians, she created in 1989 the class of musical acoustics at the Paris Conservatory (CNSMDP) and wrote an exhaustive book of musical acoustics for musicians. Director of the LAM and head of the Atiam master's program (University of Paris 6/Ircam/SupTélécom) until 2002, she is now an emeritus research director at the CNRS. She pursues her research on the musical perception of sounds within the framework of cognitive categorization and more particularly in the field of ethnomusicology.

WiMIR Keynote Talk

Improving Diversity and Inclusivity Through Action

Laurel Smith Pardue

Software Engineer

Ableton Live

Abstract

Lack of diversity and inclusion within a community means that important ideas and viewpoints are missing; a community's needs are missed or misrepresented, data is more likely to be biased, and human beings can end up feeling and being excluded. The music technology and machine learning fields are known for long-standing issues in diversity with members being largely white, male, able-bodied, and originating from wealthy countries. While there are systemic challenges to broadening a community, this talk will introduce Dr. Pardue's work on improving accessibility of musical performance before concentrating on potential actionable ways to make the research community more inclusive including focusing diversity and inclusivity efforts beyond the single strata of gender to incorporate ethnicity, language, disability, experience, and more.

Biography

Dr. Laurel S. Pardue has worked in music technology and instrument design for over 15 years. She focuses on real-world, real-time performance, designing and building instruments including Gamelan Elektrika (debuted in collaboration with Kronos Quartet at the NY Lincoln Center in 2010), the world's first electronic tabla, Tabla Touch, with Kuljit Bhamra (launched 2020), and most recently, the Svampolin (3rd place 2020, Guthman Musical Instrument Competition, NIME 2019 Best Presentation). She is also a founding member of Bela.io. She holds 4 degrees from MIT and completed a PhD at Queen Mary University of London with Dr. Andrew McPherson with follow-on research positions using technology to study the learning of musical instruments at Aarlborg University Copenhagen, and the Sonic Arts Research Centre, Queens University Belfast. Having previously worked as a ProTools software engineer, she is now a programmer for Ableton Live. Dr. Pardue is also an active violinist having appeared at major festivals in Western Europe, NY, SF, live on BBC Radios 3,4, & 6, and German television with various artists including Sam Lee, Mishaped Pearls, Bang on a Can, and, as Bitchlovsky, playing semi-improvised violin with live electronic music. She is currently the head of the NIME diversity and inclusion committee (<https://diversity.nime.org/>).

Tutorials

Tutorial 1

Tempo, Beat, and Downbeat Estimation

Matthew Davies, Sebastian Bock and Magdalena Fuentes

Abstract

The highly interrelated topics of tempo, beat, and downbeat estimation from musical audio signals have spanned the entire history of MIR. Along with many MIR topics, the uptake of deep learning has fundamentally altered how these rhythm-oriented tasks have been addressed and has led to a profound increase in performance. This tutorial seeks to position itself within this narrative by providing a high-level understanding of historical signal processing-oriented approaches leading to hands-on practical experience in building, training, and evaluating the most recent state-of-the-art deep learning approaches. Our goal is not only to expose participants to a complete rhythm analysis pipeline, but also to emphasize the importance of technical and musical design choices, the reliability of annotated data, and multidisciplinary. In addition, we seek to provide insight into common pitfalls and discuss future challenges.

The tutorial is targeted towards those in the ISMIR community who wish gain comprehensive insight and practical experience in tempo, beat, and downbeat estimation of musical audio signals. For those new to this area, we seek to provide a hands-on technical and pedagogical guide which can serve as the basis for fostering future research. For those with prior knowledge in the area, we hope to convey a solid understanding of recent advances and current state-of-the-art approaches. As a prerequisite for participation, we would expect some basic experience in the execution of python notebooks.

Biographies of Presenters

Matthew E. P. Davies is a researcher in the Centre for Informatics and Systems of the University of Coimbra (CISUC), Portugal. His research interests include the analysis of rhythm in musical audio signals, evaluation methodology, creative music applications, and reproducible research. His most recent research has addressed the use of compact deep neural networks for the analysis of rhythmic structure, and computational ethnomusicology.

Sebastian Böck received his diploma degree in electrical engineering from the Technical University in Munich and his Ph.D. in computer science from the Johannes Kepler University Linz. Within the MIR community he is probably best known for his machine learning-based algorithms and as the principal maintainer of open source python library, madmom. Currently he works as an AI research engineer for enliteAI in Vienna, Austria.

Magdalena Fuentes is a Provost's Postdoctoral Fellow at the Music and Audio Research Lab and the Center for Urban Science and Progress of New York University (NYU). She completed her Ph.D. at Université Paris Saclay on multi-scale computational rhythm analysis, with focus on the interaction of microtiming, beats, downbeats and music structure. Her research interests include Machine Listening, Self-Supervised Representation Learning, Computational Rhythm Analysis and Environmental Sound Analysis.

Tutorial 2

Designing Generative Models for Interactive Co-creation

Anna Huang, Jon Gillick and Chris Donahue

Abstract

Recent advances in generative modeling have enabled AI systems to create realistic musical outputs, and additionally offer exciting potential to assist a broad range of music creators. However, these models have limitations that impose a unique set of challenges when embedded in interactive environments.

How do we design generative models and interactions that enable new creative (or co-creative) possibilities, while at the same time addressing real musical needs and user goals? This tutorial covers a range of considerations that come into play when using AI as a design material for creative musical interactions: identifying user needs and interaction opportunities, translating our high-level interactive objectives into actionable ML problems, (crudely) codifying desired behavior or aesthetics into quantitative metrics that can be hill-climbed during model development, and identifying cadences for evaluating progress with users in controlled experiments and in the wild. This process might span multiple projects or papers, with each diving deeper on different aspects of the process. We will draw from our own experiences and projects, highlighting choices that we made and reflecting on what we might do differently next time as we trace the lifecycle of these projects from research to the real world and back.

This tutorial will be geared towards anyone with some experience in MIR who is not already working at the intersection of music generative modeling and human interaction but may be interested in learning more. The primary purpose of this tutorial is to demystify this daunting process: we will offer guidelines and point out pitfalls, keeping in mind that there is no one-size-fits-all protocol. We hope that attendees will leave the tutorial with a clearer understanding of the challenges associated with designing, building, and evaluating interactive music AI experiences, and strategies which may help them overcome these obstacles.

Biographies of Presenters

Anna Huang is a Research Scientist at Google Brain, working on the Magenta project. Her research focuses on designing generative models and tools to make music more interactive and approachable. She is the creator of Music Transformer, and the ML model Coconet that powered Google's first AI Doodle, Bach Doodle. She holds a PhD from Harvard University, masters from the MIT Media Lab, and a dual bachelor's degree in computer science and music composition from University of Southern California. She is currently co-advising students at Mila, the Quebec AI Institute. She is also a guest editor for TISMIR's Special Issue on AI and Musical Creativity, and a judge and organizer for the international AI Song Contest.

Jon Gillick is a PhD Candidate at the School of Information at UC Berkeley, where he is also affiliated with the Center for New Music and Audio Technologies (CNMAT). His research centers around exploring new ways of creating and interacting with music and sound using machine learning. Before coming to Berkeley, he studied Computer Science at Wesleyan University in Connecticut and Music Composition and Production at the Pyramid Music Production Institute in San Francisco, and he spent time working in the Bay Area as both a freelance composer/audio engineer and as a software developer.

Chris Donahue is a postdoc at Stanford University in the computer science department. The primary goal of his research is to build AI systems which help humans be more creative. In practice, this often involves both improving generative models and designing new interactive environments which make these models more useful to humans. In a music context, he is particularly interested in how generative models may allow non-musicians to unlock their dormant musical creativity. Before Stanford, Chris completed his Ph.D. at UC San Diego, where he was co-advised by Miller Puckette (music) and Julian McAuley (CS).

Tutorial 3

Scales, Chords, and Cadences: Practical Music Theory for MIR Researchers

Johanna Devaney, David Sears and Daniel Shanahan

Abstract

Much pitch-related MIR research builds either implicitly or explicitly on music-theoretic domain knowledge. Unfortunately, music theory is an esoteric discipline, with many of its canonical organizational principles presented in textbooks with dozens of classical musical examples and little indication of how these principles can be applied to other musical traditions. This tutorial will introduce fundamental pitch-related concepts in music theory for the ISMIR community and relate them to tasks associated with melodic, chord, and structural audio analysis for a range of musical styles. It will include sections on the scales, chords, and cadences routinely associated with Western art music of the common-practice tradition (~1650-1900), as well as non-Western folk musics and the popular music traditions of the twentieth and twenty-first centuries. The three sections will be broken down as follows, with both lecture and hands-on coding demonstration components:

- Scales
 - Scale formation (octave equivalence, mathematical properties)
 - Scale and mode types (western and non-Western)
 - Implications for scale and key identification, automatic melody extraction
- Chords
 - Types (triads, seventh chords, extensions)
 - Representation schemes (e.g., chord labeling)
 - Syntactic principles (e.g., functional harmony, grammars)
 - Implications for automatic chord recognition, pattern discovery
- Cadences
 - Types
 - Linear/voice-leading patterns
 - Relationship to large-scale formal types (phrases, themes, sonata, etc.)
 - Implications for cadence discovery/classification, automatic segmentation

This tutorial will be of interest to a broad range of the ISMIR community, but will be of specific interest to MIR researchers with limited formal training in music theory. This workshop assumes a basic understanding of musical notation, but does not assume prior knowledge of Western music theory. It will be accessible to researchers new to the field, but will also be of interest to experienced researchers hoping to incorporate more music-theoretically based models into their research.

Biographies of Presenters

Johanna Devaney is an Assistant Professor at Brooklyn College and the CUNY Graduate Center, where she teaches courses in data analysis, music technology, music theory, and sonic arts. Her research focuses on interdisciplinary approaches to the study of musical performance, with a particular focus on the relationship between pitch structure and intonation in the singing voice. More broadly, she examines the ways in which recorded performances can be used to model performance and develops computational tools to facilitate this, primarily the Automatic Music Performance Analysis and Comparison Toolkit (AMPACT). Johanna has been active in the ISMIR community since 2008, giving the WiMIR keynote in 2020 and currently serving on the TISMIR editorial board. She holds a PhD in Music Technology from McGill University.

David R. W. Sears is an Assistant Professor of Interdisciplinary Arts and Co-Director of the Performing Arts Research Lab at Texas Tech University, where he teaches courses in arts psychology, arts informatics, and music theory. His current research examines the structural parallels between music and language using both behavioral and computational methods, with a particular emphasis on the many topics associated with pitch structure, including scale theory, tonality, harmony, cadence, and musical form. Recent publications appear in his [Google Scholar profile](#). He holds a PhD in music theory from McGill University.

Daniel Shanahan is an Associate Professor of Music Theory and Cognition at Ohio State University. He is interested in studying musical transmission, musical communication, and jazz improvisation, and likes to explore these topics with both experimental and computational tools. Daniel's work has been published in *Music Perception*, *The Journal of New Music*

Research, *Musicae Scientiae*, and many other outlets. He is an editor of the forthcoming *Oxford Handbook of Corpus Studies in Music* and has been managing editor of *Empirical Musicology Review* since 2012, serving as the journal co-editor since 2016. He also serves on the editorial boards of *Music Theory Spectrum*, *Musicae Scientiae*, and *Indiana Theory Review*. He holds a PhD in music theory from the University of Dublin, Trinity College.

Tutorial 4

Music Classification: Beyond Supervised Learning, Towards Real-world Applications

Keunwoo Choi, Minz Won and Janne Spijkervet

Abstract

Music classification is a music information retrieval (MIR) task to classify music items to labels such as genre, mood, and instruments. It is also closely related to other concepts such as music similarity and musical preference. In this tutorial, we put our focus on two directions - the recent training schemes beyond supervised learning and the successful application of music classification models.

The target audience for this session is researchers and practitioners who are interested in state-of-the-art music classification research and building real-world applications. We assume the audience is familiar with the basic machine learning concepts. For those who are not, we kindly refer to [1, 2] to be prepared for this session.

We plan to present three lectures as follows:

1. Music classification overview: Task definition, applications, existing approaches, datasets
2. Beyond supervised learning: Semi- and self-supervised learning for music classification
3. Towards real-world applications: Less-discussed, yet important research issues in practice

We provide an accompanying code repository and Jupyter notebooks that can be used along with the video presentation. With the material, attendees can easily train semi- and self-supervised models with their own audio data.

[1] A Tutorial on Deep Learning for Music Information Retrieval, Keunwoo Choi et al., 2017 (Concepts in deep learning) <https://arxiv.org/abs/1709.04396>

[2] An Introduction to Statistical Learning, Daniela Witten et al., 2013 (Chapter 2-4 for ML fundamentals) <https://www.statlearning.com/>

Biographies of Presenters

Keunwoo Choi ([website](#)) is a research scientist at ByteDance, developing machine learning products for music recommendation and discovery. He received a Ph.D degree from [Queen Mary University of London \(c4dm\)](#) in 2018. As a researcher, he also has been working at Spotify (2018 - 2020) and several other music companies as well as open-source projects such as [Kapre](#), [librosa](#), and [torchaudio](#). He argues that he writes [some good music](#).

Minz Won ([website](#)) is a Ph.D candidate at the Music Technology Group (MTG) of Universitat Pompeu Fabra in Barcelona, Spain. His research focus is music representation learning. Along with his academic career, he has put his knowledge into practice with industry internships at Kakao Corp., Naver Corp., Pandora, Adobe, and he recently joined ByteDance as a research scientist. He contributed to the winning entry in the WWW 2018 Challenge: Learning to Recognize Musical Genre.

Janne Spijkervet ([website](#)) graduated from the University of Amsterdam in 2021 with her Master's thesis titled "Contrastive Learning of Musical Representations". The paper with the same title was published in 2020 on self-supervised learning on raw audio in music tagging. She has started at ByteDance as a research scientist (2020 - present), developing generative models for music creation. She is also a songwriter and music producer, and explores the design and use of machine learning technology in her music.

Tutorial 5

Programming MIR Baselines from Scratch: Three Case Studies

Rachel Bittner, Mark Cartwright and Ethan Manilow

Abstract

This tutorial will walk through the creation of MIR baselines programmed live, including pitch tracking, instrument identification, and drum transcription. Each case study will start with building a system and finish with evaluations and visualization/sonification, each using different tools and styles of programming. This tutorial is both beginner and experienced programmer-friendly and will start from the basics but will move quickly. While the tutorial is not interactive, all code will be made available afterwards.

Biographies of Presenters

Rachel Bittner is a Senior Research Scientist at Spotify in Paris. She received her Ph.D. in Music Technology in 2018 from the Music and Audio Research Lab at New York University under Dr. Juan P. Bello, with a research focus on deep learning and machine learning applied to fundamental frequency estimation. She has a Master's degree in mathematics from New York University's Courant Institute, as well as two Bachelor's degrees in Music Performance and in Mathematics from the University of California, Irvine. In 2014-15, she was a research fellow at Telecom ParisTech in France after being awarded the Chateaubriand Research Fellowship. From 2011-13, she was a member of the Human Factors division of NASA Ames Research Center, working with Dr. Durand Begault. Her research interests are at the intersection of audio signal processing and machine learning, applied to musical audio. She is an active contributor to the open-source community, including being the primary developer of the pysox and mirdata Python libraries.

Mark Cartwright is an Assistant Professor at New Jersey Institute of Technology in the Department of Informatics. He completed his PhD in computer science at Northwestern University as a member of the Interactive Audio Lab, and he holds a Master of Arts from Stanford University (CCRMA) and a Bachelor of Music from Northwestern University. Before his current position, he spent four years as a researcher in the Music and Audio Research Lab (MARL) and the Center for Urban Science and Progress (CUSP) at New York University (NYU). His research lies at the intersection of human-computer interaction, machine learning, and audio signal processing. Specifically, he researches human-centered machine listening and audio processing tools for creative expression with sound and understanding the acoustic world.

Ethan Manilow is a PhD candidate in Computer Science at Northwestern University under advisor Prof. Bryan Pardo. His research lies in the intersection of signal processing and machine learning, with a focus on source separation, automatic music transcription, and open source datasets and applications. Previously he was an intern at Mitsubishi Electric Research Labs (MERL) and at Google Magenta. He is one of the lead developers of nussl, an open source audio separation library. He lives in Chicago, where he spends his free time playing his guitar and smiling at dogs he passes on the sidewalk.

Tutorial 6

Teaching Music Information Retrieval

George Tzanetakis

Abstract

The research field of Music Information Retrieval (MIR) has a history of more than 20 years. During this time many different tasks have been defined and a variety of algorithms have been proposed. MIR topics are taught around the world in a variety of settings both in academia and industry. The teaching of MIR takes many forms ranging from teaching regular undergraduate and graduate courses to delivering specialized tutorials, seminars, and online courses. MIR is a fundamentally interdisciplinary topic and that creates unique challenges when it is taught. The goal of this tutorial is to cover various topics of interest to people involved with teaching MIR. The material covered is informed by modern pedagogical practices and how these practices can be adapted to address the unique characteristics of learning about MIR. The global Covid pandemic has resulted in increased activity and interest about online learning. Advice and guidelines for effective online teaching of MIR will also be provided. The presented concepts and ideas will be illustrated using concrete examples and use cases drawn from extensive experience of the tutorial presenter with teaching MIR in a variety of settings. Although this is not the primary focus of the tutorial, these examples can also serve as an introduction to MIR for participants that are new to the field.

Biography of the Presenter

George Tzanetakis is a Professor in the Department of Computer Science with cross-listed appointments in ECE and Music at the University of Victoria, Canada. He was Canada Research Chair (Tier II) in the Computer Analysis and Audio and Music from 2010 to 2020. In 2012, he received the Craighdaroch research award in artistic expression at the University of Victoria. In 2011 he was Visiting Faculty at Google Research. He received his PhD in Computer Science at Princeton University in 2002 and was a Postdoctoral fellow at Carnegie Mellon University in 2002- 2003. His research spans all stages of audio content analysis such as feature extraction, segmentation, classification with specific emphasis on music information retrieval.

He has designed and developed for Kadenze Inc. the first widely available online program in Music Information Retrieval consisting of 3 courses that were launched in December 2020. More than 2000 students from around the world have been involved with the program. He is also the primary designer and developer of Marsyas an open source framework for audio processing with specific emphasis on music information retrieval applications. His pioneering work on musical genre classification received a IEEE signal processing society young author award and is frequently cited. He has given several tutorials in well-known international conferences such as ICASSP, ACM Multimedia and ISMIR. More recently he has been exploring new interfaces for musical expression, music robotics, computational ethnomusicology, and computer-assisted music instrument tutoring. These interdisciplinary activities combine ideas from signal processing, perception, machine learning, sensors, actuators and human-computer interaction with the connecting theme of making computers better understand music to create more effective interactions with musicians and listeners. More details can be found <http://www.cs.uvic.ca/~gtzan>.

Papers

FOUR-WAY CLASSIFICATION OF TABLA STROKES WITH MODELS ADAPTED FROM AUTOMATIC DRUM TRANSCRIPTION

Rohit M A Amitrajit Bhattacharjee Preeti Rao

Department of Electrical Engineering

Indian Institute of Technology Bombay, India

{rohitma, amitrajit, prao}@ee.iitb.ac.in

ABSTRACT

Motivated by musicological applications of the four-way categorization of tabla strokes, we consider automatic classification methods that are potentially robust to instrument differences. We present a new, diverse tabla dataset suitably annotated for the task. The acoustic correspondence between the tabla stroke categories and the common popular Western drum types motivates us to adapt models and methods from automatic drum transcription. We start by exploring the use of transfer learning on a state-of-the-art pre-trained multiclass CNN drums model. This is compared with 1-way models trained separately for each tabla stroke class. We find that the 1-way models provide the best mean f-score while the drums pre-trained and tabla-adapted 3-way models generalize better for the most scarce target class. To improve model robustness further, we investigate both drums and tabla-specific data augmentation strategies.

1. INTRODUCTION

Tabla, a ubiquitous part of the North Indian art music ensemble, comprises two drums that can be struck singly or together with a variety of articulations to give rise to sequences of individual and compound strokes of changing timbre, termed *bols*. With a set of between 10-20 distinct tabla bols (depending on playing style) found in practice, the bols have been traditionally viewed as single entities of different timbres, and tabla transcription addressed as a monophonic timbre recognition problem [1].

The earliest work on tabla transcription was reported by Gillet et. al. [2] who modelled stroke spectra by a 4-mixture Gaussian Mixture Model for 10-category classification using Hidden Markov Models (HMM). Chordia [3] extended this work by targeting a larger, more diverse dataset, and using neural network (NN) and tree-based classifiers to categorize strokes based on spectral and temporal envelope features. Both works mention the difficulty of generalizing across instruments, and report lower scores

on tabla sets not seen in training. Later work [4] used frame-level mel-frequency cepstral coefficients (MFCC) to capture bol timbre in an HMM model in a classification task on a single tabla set. More recent works that make use of NN and tree-based bol classifiers [5–8] are restricted either in their use of small datasets, or the absence of any instrument-independent performance evaluation.

An important taxonomic level for tabla sounds is based on which of the two drums is struck and the manner of striking, giving rise to the three classes: resonant treble (right drum), resonant bass (left drum), and damped (either drum); the right & left are with respect to a right-handed player. That is, the specific manner gives rise to either a damped stroke with a sharp and short-duration sound or a pitched (resonant) stroke with ringing sound, which can further be pitch modulated in the case of the left drum. The different timbres of the tabla bols are obtained by individual or combinations of basic strokes, with the combination of resonant bass and treble (resonant both) being especially important. In the archetypal drum pattern known as the *theka*, subsections of the rhythmic cycle are chiefly discriminated by the presence or absence of right and left drum resonant strokes [9, 10]. The associated classification has been useful in the empirical analyses of tabla accompaniment in khyal vocal performances [11]. Motivated by the musicological applications of the above categorization of tabla sounds, a 4-way stroke classification task was previously defined exploiting the acoustic characteristics of the strokes [12]. A training dataset of labelled tabla solo recordings was created to train a random forest classifier

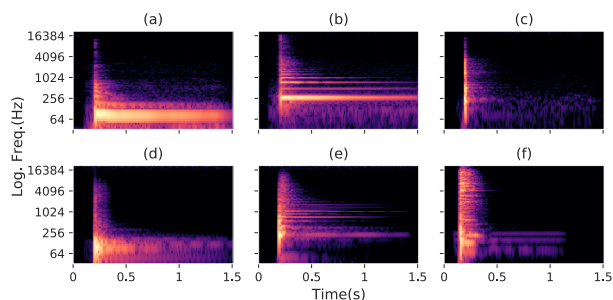


Figure 1: Spectrograms of samples (at $f_s=44.1$ kHz) of the 3 basic tabla strokes (top) and drum types (bottom). Note the similarity between (a) Resonant Bass & (d) BD, (b) Resonant Treble & (e) SD, and (c) Damped & (f) HH



© M. A. Rohit, A. Bhattacharjee, and P. Rao. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. A. Rohit, A. Bhattacharjee, and P. Rao, “Four-way Classification of Tabla Strokes with Models Adapted from Automatic Drum Transcription”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

Category	Bols	Bass	Treble	Drumkit
D	Ti-Ta, Te-Re, Tak, Ke, Tra, Kda	D/Nil	D/Nil	HH
RT	Na, Tin, Tun, Din	D/Nil	R	SD
RB	Ghe, Dhe, Dhi, Dhet	R	D/Nil	BD
B	Dha, Dhin	R	R	BD+SD

Table 1: Tabla stroke categories, corresponding bols, constituent stroke types (Resonant/Damped/none) on each tabla drum, and western drum equivalents. D - damped, RT - resonant treble, RB - resonant bass, B - resonant both.

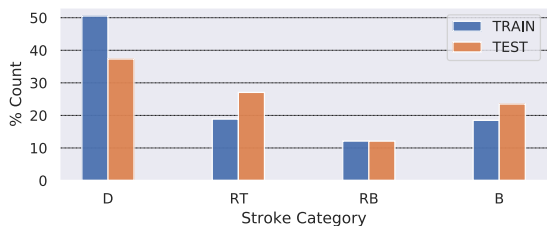


Figure 2: Distribution of stroke categories in our dataset.

with a large set of stroke specific acoustic features, which was then evaluated on a tabla accompaniment test set.

In this work, we recognise the similarity of the reduced category tabla stroke classification problem to the automatic drums transcription (ADT) task with its considerable published work focused on transcribing the 3 main percussion instruments in Western popular music – bass (BD), snare (SD), and hi-hat (HH) [13, 14]. Figure 1 illustrates the correspondence of these drums with the bass, treble, and damped tabla strokes, respectively. Starting with segment-and-classify approaches based on extracting suitable acoustic features for classification from automatically segmented drum tracks using onset detection, more recent methods for ADT adapt deep-learning based onset detection models, trained to directly predict the instrument along with its onset location [14]. We extend previous available work on 4-way tabla stroke category detection to use recently proposed convolutional neural network (CNN) models comprising the state-of-the-art in ADT.

Our chief new contributions include significantly expanding the available training dataset of tabla solo recordings with new instruments, and investigating CNN architectures from ADT literature for the tabla stroke classification task. In an attempt to alleviate training data scarcity, we explore domain adaptation or transfer learning with an available pre-trained multiclass CNN drums model [15]. To counter target class imbalance, we also investigate architecture optimizations with a bank of single-stroke (binary) CNN classifiers [16]. Finally, we explore a number of data augmentation approaches including new tabla-specific transformations inspired by drum-specific augmentation methods from ADT [17]. We present next the dataset used in this work, followed by the classification and data augmentation methods and, finally, the results.

	Source	# tablas	Duration	# strokes
Train	Train-set of [12]	3	18 min.	6,680
	Suppl. data of [18]	3	16 min.	5,178
	New	4	42 min	14,742
	Total	10	76 min.	26,600
	Test-set of [12]	3	20 min.	4,470

Table 2: The various subsets in the train and test datasets.

2. DATASET

An important application of the present work is in classifying tabla strokes played in accompaniment to lead music. We thus use an existing dataset of realistic tabla accompaniment recordings to test our methods. While we would prefer matched training data, concert audios are not readily available in bleed-free multi-track format. And creating such a dataset is challenging not only to record, but also to annotate due to the lack of a precise score. Therefore, we resort to the use of tabla solo playing and build upon previous datasets to create a diverse training set. Table 2 lists the sources of our train and test datasets with a common sampling rate of 16 kHz. The target classes used in this work, common bols that they map to, and the types of strokes played simultaneously on each drum to realise them appear in Table 1. For D, we have damped strokes on either one or both drums. RT and RB strokes produce resonant sounds on the corresponding drum and may be accompanied by a damped stroke on the other drum.

Testing: The test set consists of 10 pieces of only the tabla accompaniment recorded in perfect isolation to pre-recorded solo Hindustani vocal tracks. It contains 20 minutes of audio and nearly 4,500 strokes. These recordings, made on 3 unique tabla sets by 2 different artists, are diverse in terms of tuning, *tala* (metre), and tempo.

Training and Cross-Validation (CV): Solo compositions and common theka patterns recorded from 7 different tabla-sets are added to the training dataset from [12]. Out of these, 3 are from a previous study [18] for which written scores are available but are not time-aligned with the audios. The 4 others were newly recorded for this work by different artists. In order to achieve better diversity, we choose instruments of sufficiently different tuning, include a variety of playing styles, and cover a wide tempo range. Annotation was carried out by automatically aligning the composition score (supplied by artists) with the audios, and replacing the bols with corresponding target stroke categories (Table 1). Given the imperfect score-stroke matching [3, 4], labels were manually verified to assign the same category to similar sounding bols. The dataset spans a total audio duration of about 1.25 hours and contains 26,600 strokes. To perform hyperparameter tuning, we split our training dataset into 3 nearly equal-sized disjoint folds, with all recordings from a single tabla set assigned to a single fold, providing instrument independent validation. The folds are similar in the distribution of stroke categories, tonic, and tempo.

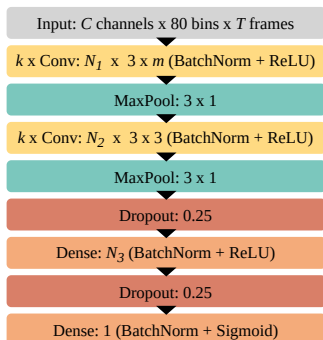


Figure 3: General CNN architecture for 1-way model tuning experiments ($N_2=2 \times N_1$). k is # repetitions of the layer.

Variant	Hyperparameter values
Baseline	$C=3, T=15$ (150 ms), $k=1, m=7, N_1=16, N_3=128$
\uparrow context	$T=21$ (210 ms)
Mid-channel	$C=1$ (middle)
\uparrow conv filters	$N_1=32$
\uparrow dense units	$N_3=256$
\uparrow conv filt. + \uparrow dense units	$N_1=32, N_3=256$
$2 \times$ conv layers	$k=2, m=3$

Table 3: The various hyperparameter settings in the tuning experiments of the 1-way CNN model (of Figure 3).

Figure 2 shows the distribution of strokes across the four target categories in the train and test sets. We observe a significant imbalance in both, with damped strokes being the most numerous and resonant bass being the least. Although the distributions are similar across datasets, differences in the playing styles (solo vs accompaniment) are likely to contribute to some train-test mismatch.

3. METHODS

We now present the CNN based classification models, their input-output representations, and the training and hyperparameter tuning experiments. Subsequently, we outline the different augmentation methods devised.

3.1 Classification Models

Two CNN-based approaches from drums transcription are compared - a 3-way model [19], and a bank of separate 1-way models for each target class [16]. In the former, we experiment with fine-tuning available pre-trained models as well as training new models with the same architecture from scratch. With the 1-way approach, a model for each stroke category is trained from scratch and their hyperparameters are optimised separately.

3.1.1 3-way Classification

We use the four 3-way CNN models from the python library *madmom* [20], each of which is trained on a different

subset of the MIREX17 drums dataset [15]. During training of the 3-way CNN, the fourth ‘resonant both’ (B) label in tabla is replaced by simultaneous onsets in RB and RT (see Table 1). Model outputs are post-processed during evaluation to obtain 4-class predictions, by replacing RB and RT onsets predicted within 10 ms with B.

Based on the common assumed roles for a CNN’s conv and dense layers of feature extractor and classifier respectively [21, 22], we explore two transfer learning strategies to fine-tune (FT) the pre-trained (PT) models on our smaller (by $\approx 3 \times$) dataset: (a) FT all dense layers while keeping all conv layers frozen at PT values, and (b) FT all layers. Under (b), we study three approaches - uniform, differential, and disjoint. In uniform and differential FT, all layers are simultaneously tuned, with the learning rate (lr) kept same for all layers in the former, and different for conv and dense layers in the latter. Disjoint FT refers to the alternating (rather than simultaneous) tuning until convergence of the dense and conv layers, in order to reasonably constrain the updatable parameters at any time. Other fine-tuning combinations with tuning only a subset of dense layers were not found to be favorable. While fine-tuning all layers uniformly has been previously used in audio event tagging [23], the differential and disjoint approaches are motivated from speech recognition [24, 25]. Finally, we consider also the PT initialisation of dense layers in all cases applicable, in addition to the usual random initialisation used for dense layers in domain adaptation. For baselines, we report results from the pre-trained models, as well as a new model with the same architecture trained from scratch (i.e. re-trained) solely on our dataset.

The input & target representations, and the optimizer used are as originally reported [15], with tabla audios up-sampled to 44.1 kHz. We expect the reduced bandwidth of our data to influence the performance minimally since the 8-15 kHz band (which is only faintly energetic in BD and SD onsets) accounts for a minor fraction of the bins in the log-scaled spectral representation. Dropout ($p = 0.5$) is added before the first dense layer, batch size is increased to 64, and early stopping with a patience of 10 epochs is included. Learning rates are not decayed across epochs and were empirically determined to be: $1e^{-5}$ in re-training, $1e^{-6}$ for dense and a lower $1e^{-7}$ for conv layers in differential FT (to better preserve the generalization capabilities of lower layers), and $1e^{-6}$ in all other experiments.

3.1.2 1-way Classification

The general model architecture used for this method appears in Figure 3. First, a common CNN model architecture (‘Baseline’ in Table 3) is obtained for all stroke classes by making modifications, targeted at achieving better convergence, to a previous architecture from ADT [16]. The input to the baseline model is a set of 3 log-scaled mel-spectrograms of dimensions 80 bands x 15 frames (150 ms) as proposed previously [26], computed from 16 kHz audios. Target activations are prepared by assigning a value of 1 to every ground truth onset frame as well as an adjacent frame on either side, and 0 to the remaining frames.

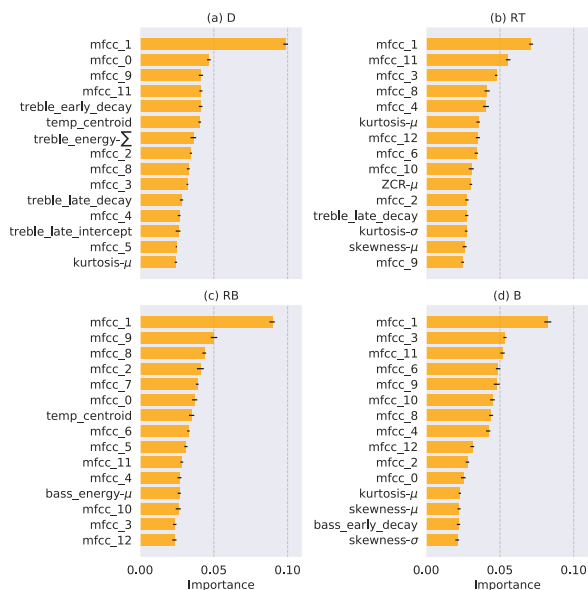


Figure 4: The 15 highest ranked features from the tabla identification task for each stroke type. Bars and whiskers show mean and standard deviation across 3 repetitions.

Models are trained to minimize BCE loss for a maximum of 150 epochs, using the Adam optimizer with lr equal of $1e^{-4}$, a batch size of 256, and early stopping with a patience of 10 epochs.

Considering the different amounts of data available for each stroke type, we experiment with the input representation and model architecture to arrive at the best hyperparameters for each class. A list of the chosen hyperparameter settings, along with the baseline, appears in Table 3. The input-related choices are based on experiments in ADT involving larger context [19] and alternate input representations [16]. Architecture variations to the conv and dense layer sizes are targeted at increasing the model capacity to benefit classes with more data.

3.1.3 Random Forest Classification

The random-forest based tabla stroke classification system presented in [12] is re-trained on our newly configured dataset to use as a baseline. This system uses the high-frequency content algorithm [27] to first segment tabla audios based on the detected onsets, and then extracts a set of 49 acoustic features for the 4-way stroke classification with a random forest. Features relating to the temporal characteristics of decay portions are found to be especially important to stroke identity across instruments.

3.2 Data Augmentation Methods

With data augmentation, we seek to simulate tabla instrument diversity from different physical structures, tuning, and playing styles. We employ pitch-shifting and time-scaling (audio-specific), attack-remixing (percussion-specific), and two methods closely tied to the acoustics and sound production characteristics of tabla - spectral filtering and NMF-based stroke-remixing (tabla-specific). All

transformations are applied to the time-domain audio signal. To ensure that the modified audio sounds realistic, we enlist the help of an expert tabla player to determine the suitable range for the control parameter in each transformation. With every augmentation method, we use 4 values of the parameter (as in Table 4), obtaining 4 versions for each audio. This is combined with the original dataset, thus increasing the size to five times the original. During training, data from two CV folds (along with any augmentations) is used, while only the original data from the third fold is used for validation.

3.2.1 Pitch-shifting (PS) and Time-scaling (TS)

We use the *hptsm* algorithm from the python library *pytsmod* [28], which first separates an audio into harmonic and percussive components and then applies appropriate time-scaling methods to each component. Pitch-shifting is performed by time-scaling followed by re-sampling. The parameters α_{ps} and α_{ts} denote the pitch-shifting (in semitones) and time-scaling factors.

3.2.2 Attack-remixing (AR)

Attack-remixing refers to modifying the relative levels of attack and decay regions of an audio, and has been used to augment drums data [16]. Our implementation involves first applying harmonic-percussive separation (HPS) [29] to the audio, which leaves all the attacks in the percussive component and resonant decay portions in the harmonic component. The percussive component is scaled by a linear factor, denoted by α_{ar} , and remixed with the unmodified harmonic component.

3.2.3 Spectral Filtering (SF)

Augmenting data by perturbing ‘nuisance attributes’ that are unimportant in the specific discrimination task, can be effective [30]. We use the feature ranking of the random forest classifier to identify acoustic features that capture instrument characteristics and are less important to stroke identity. The classifier, as in Section 3.1.3, is re-purposed to solve a 10-way tabla identification task on our training set (spanning 10 instruments), using the same features. It achieves an average f-score of over 0.9 in a random 3-fold CV performed separately with each stroke category. From the resulting feature importances presented in Figure 4, we see that various MFCCs (computed frame-wise and averaged across stroke segments), representing spectral shape, are most important to instrument differentiation, with MFCC-1, representing the tilt (balance) between high and low frequencies, consistently at the top. This motivates the use of particular filtering transformations to modify spectral shape for the data augmentation.

Due to the contrasting broadband and band-limited nature of tabla attack and decay spectra, it is more effective to use filters targeting specific bands, instead of the commonly used random filtering [31]. After first applying HPS, we filter the bass (0 – 200 Hz) and treble (200 – 2k Hz) regions in the harmonic component, and modify the spectral tilt in the percussive component by changing the

Method	Values	Method	Values
α_{ps}	-1, -0.5, 0.5, 1	$\alpha_{sf-tilt}$	0.2, 0.5, 2, 3
α_{ts}	0.8, 0.9, 1.2, 1.3	$\alpha_{sr-bass}$	0.6, 0.8, 1.5, 2
α_{ar}	0.3, 0.5, 2, 3	$\alpha_{sr-treble}$	0.5, 0.8, 1.5, 2
$\alpha_{sf-bass}$	0.2, 0.5, 2, 4	$\alpha_{sr-damp}$	0.2, 0.5, 2, 3
$\alpha_{sf-treble}$	0.2, 0.5, 2, 4		

Table 4: Parameter values for the augmentation methods.

energy balance across the two halves of the spectrum. The filter is a *Hann* window positioned over the corresponding band and scaled by a linear gain factor, and is multiplied with each short-time spectral slice. Each filtering operation (bass, treble, & tilt) is considered a separate transformation with the corresponding gain factor denoted by $\alpha_{sf-bass}$, $\alpha_{sf-treble}$, and $\alpha_{sf-tilt}$. We also consider a combination of all three transformations applied simultaneously by randomly selecting a value for each parameter, while still obtaining 4 augmented versions per audio.

3.2.4 Stroke Remixing (SR)

Given that the compound bols of the tabla are produced by the independent, though simultaneous, striking of the two drums, we simulate the expected variations of the relative strengths of the drums in this mix using non-negative matrix factorization (NMF). We use the *NMFToolbox* [32] to perform the decomposition. The activations are randomly initialised while the templates, a total of 6, computed separately from attack and decay regions for each of the 3 distinct stroke types (resonant bass, resonant treble, and damped), are kept fixed. Each template is the average spectrum of the corresponding portion of the signal from across 10 isolated instances. A separate set of templates is computed for each tabla instrument in our training set and used to decompose recordings from the corresponding tabla.

To perform augmentation, we first obtain the audio for each component from the decomposition, combine the attack and decay portions for each stroke type, and then re-synthesize audio by mixing the three stroke components at different linearly scaled levels. We experiment with restricting scaling to only one of the three components at any time (factors denoted by $\alpha_{sr-bass}$, $\alpha_{sr-treble}$, and $\alpha_{sr-damp}$), as well as a combination with all components simultaneously scaled by different randomly chosen factors (similar to the combination method in filtering).

4. EXPERIMENTAL RESULTS

We evaluate performance using the f-score metric with a tolerance of 50 ms for the detected onset locations [33]. Scores are obtained separately for each stroke class on individual tracks and averaged across the dataset. The reported CV scores are the mean across 3 folds. For the network predictions, local peaks in the output layer activations are detected and thresholded. The threshold is selected based on maximizing validation set f-score and then used on the test set. In the transfer learning experiments, all 4 pre-trained models are tuned separately and used to-

Model	Stroke category			
	D	RT	RB	B
Baseline	84.6	83.2	46.5	83.8
↑context	84.3	81.4	41.9	73.0
Mid-channel	84.7	81.7	42.1	75.6
↑conv filters	84.7	84.5	44.7	77.6
↑dense units	86.7	82.9	40.1	73.6
↑conv filters+↑dense units	83.5	83.4	43.3	82.0
2x conv layers	84.3	82.4	42.4	75.9

Table 5: CV f-scores of 1-way model tuning experiments (bold values are highest in the column).

Method	Stroke category				Mean
	D	RT	RB	B	
No aug.	86.7	84.5	46.5	83.8	75.4
Pitch-shift	<u>87.2</u>	85.5	<u>51.2</u>	<u>83.9</u>	<u>76.9</u>
Time-scale	<u>88.2</u>	<u>85.0</u>	<u>50.2</u>	<u>82.2</u>	<u>76.4</u>
Attack-remix	84.3	84.2	48.1	81.3	74.5
SF-bass	84.5	80.9	40.4	79.9	71.4
SF-treble	85.8	81.7	48.7	76.0	73.0
SF-tilt	86.3	82.7	43.8	82.0	73.7
SF-all	<u>87.6</u>	84.6	<u>50.7</u>	<u>85.6</u>	<u>77.1</u>
SR-bass	86.0	<u>84.8</u>	43.3	83.6	74.4
SR-treble	86.1	<u>84.8</u>	39.4	79.0	72.3
SR-damp.	86.2	<u>85.3</u>	<u>50.1</u>	<u>86.5</u>	<u>77.0</u>
SR-all	<u>86.8</u>	<u>85.3</u>	48.1	<u>84.4</u>	76.2
Combined	88.5	84.2	53.6	87.9	78.5

Table 6: Comparing the CV f-scores of 1-way models trained using different augmentation methods (bold values are overall highest in column, underlined are top 4 among individual methods). Combined refers to PS+TS+SF-all+SR-all.

gether (ensemble) by averaging their predicted activations during cross-validation. On the test set, an ensemble of 12 models (4 from each CV split) is utilised. With 1-way classification, single models are evaluated during CV and an ensemble of the 3 models is used on the test set.

1-way model tuning: The cross-validation results of the tuning experiments with the 1-way models (discussed in Sec. 3.1.2) appear in Table 5. The input-related modifications do not lead to improved scores in any class, indicating that a 3-channel representation with moderate context duration (150 ms) is optimum for our task. With respect to model architecture, the baseline appears to be best for classes with least data (RB and B). The use of more dense units benefits only the more abundant damped class. With increased conv layer filters, the f-score for resonant treble goes up, possibly by better learning its rich and diverse harmonic content stemming from tabla tuning variations. The other modifications do not offer any further improvements.

Data augmentation: Table 6 shows the results of training the 1-way models (with hyperparameters for each class as identified in the tuning experiments), using the various augmentation methods. The underlined values are the 4

Method	Stroke category				Mean
	D	RT	RB	B	
Random forest	86.2 / 74.2	77.7 / 75.0	39.7 / 35.3	73.6 / 41.5	69.3 / 56.5
Pre-trained (PT)	36.8 / 27.3	15.1 / 9.0	9.8 / 19.8	7.3 / 2.1	17.3 / 14.6
Re-trained	81.0 / 65.5	53.7 / 72.6	15.7 / 22.9	63.0 / 60.0	53.4 / 55.3
FT dense random init.	74.4 / 65.2	55.9 / 75.2	33.6 / 45.6	63.4 / 52.0	56.8 / 59.5
FT dense PT init.	71.7 / 62.4	54.8 / 74.9	29.4 / 34.8	60.9 / 39.1	54.2 / 52.8
Uniform FT all	76.3 / 65.1	59.7 / 79.1	29.5 / 43.3	65.3 / 58.6	57.7 / 61.5
Differential FT all	72.5 / 63.4	58.7 / 77.9	30.0 / 41.2	63.5 / 49.2	56.2 / 57.9
Disjoint FT all: dense rand. init.	77.2 / 67.2	57.4 / 73.1	33.0 / 49.1	65.9 / 60.9	58.3 / 62.6
Disjoint FT all: dense PT init.	74.8 / 65.4	66.4 / 77.4	34.7 / 47.5	66.5 / 56.8	60.6 / 61.8
1-way No aug.	86.7 / 79.5	84.5 / 84.1	46.5 / 38.0	83.8 / 69.0	75.4 / 67.6
1-way Best aug.	88.5 / 83.3	85.5 / 84.3	53.6 / 34.1	87.9 / 80.1	78.9 / 70.4

Table 7: F-scores (CV/test) from 3-way models compared with the best 1-way models and a random-forest baseline [12] (values in bold are highest in the column). ‘Best aug.’ represents pitch-shifting for RT and combined aug. for the rest.

highest scores within each class that are better than no augmentation. We note that these are most often from using one of PS, TS, SF-all, SR-damp, or SR-all. We therefore experiment with combining PS, TS, SF-all, and SR-all (which includes SR-damp), by randomly choosing only 2 out of 4 versions from each method for every audio (to limit training time), taking the dataset size to 9x original. Values in bold indicate the highest scores obtained in each stroke category across all methods (individual and combination). Overall, we see that except for the resonant treble class, the combination results in the best f-scores, demonstrating the benefit of the proposed augmentation methods.

Some notable observations about the individual methods follow. The improvements from pitch-shifting and time-scaling underscore the importance of addressing tuning diversity and capturing a wide tempo range when working with datasets of realistic playing. With tabla-specific filtering and remixing, the combinations SF-all & SR-all, which pack more diversity, outperform the corresponding individual methods in most cases, and consistently give better f-scores than no augmentation.

3-way vs 1-way: Table 7 compares the CV and test set scores of the 3-way models against the best 1-way models and the random forest baseline. In the transfer learning experiments, we note that tuning conv layers helps, possibly compensating for low level acoustic differences between tabla strokes and drums. Of the three approaches to this, disjoint FT gives higher CV and test scores when compared to the other two. With regards to random versus pre-trained initialisation for dense layers in the disjoint FT setup, better test set scores are obtained with random initialisation, indicating better generalization, while PT initialisation gives higher CV scores. Finally, these domain-adapted models outperform the pre-trained only and the re-trained (from scratch) 3-way models.

Eventually, we find 1-way models mostly surpassing the best 3-way model, with data augmentation further enhancing performance. Test scores are lower than that of CV train by a few percentage points, attesting to the persistent mismatch from playing style. Only for the test resonant

bass, the f-score is highest using disjoint FT, which shows that transfer learning has helped with generalization for the class with least data. A closer look at disjoint FT versus the 1-way models further reveals that the most difference in f-score is in resonant both, indicating that treating the combination stroke as a separate class works better than viewing it as the superposition of its component stroke classes. Finally, it is interesting to note that the 3-way models trained from scratch perform much poorer than the set of similarly trained 1-way models, demonstrating the benefit of using separate models specialised for each class in this task.

5. CONCLUSIONS

We presented a four-way tabla stroke classification task for categories defined by the salient acoustic characteristics of the basic tabla strokes. Leveraging the similarity of our target categories with popular Western drumkit classes, we investigated methods from the automatic drums transcription task. We explored the adaptation of available pre-trained drums models via transfer learning on a new tabla dataset. Systematic experiments with different transfer learning strategies reveal significant improvements when both dense (classifier) layers and conv (feature extractor) layers of a multiclass CNN model are fine-tuned from the pre-trained weights in a disjoint fashion. Next, the use of separate 1-way CNN models with hyperparameters suitably tuned for each of the 4 stroke categories was found to surpass the more complex 3-class CNN model for all class accuracies except the most data-constrained resonant bass category, which benefited from pre-training on drums. Further, several data augmentation methods, untested so far in the context of tabla, were investigated. A method based on increasing training data diversity, by varying spectral characteristics that capture instrument-dependence across strokes, contributed consistently to improved classification accuracy. Future work will target recurrent architectures and the combination of transfer learning and data augmentation for further performance gains.

Supplementary: github.com/DAP-Lab/4way-tabla-transcription

6. REFERENCES

- [1] P. Chordia, “Automatic transcription of tabla music,” Ph.D. dissertation, Stanford University, 2006.
- [2] O. Gillet and G. Richard, “Automatic labelling of tabla signals,” in *Proc. of the 4th Int. Society for Music Information Retrieval Conf.*, Baltimore, U.S.A., 2003.
- [3] P. Chordia, “Segmentation and recognition of tabla strokes,” in *Proc. of the 6th Int. Society for Music Information Retrieval Conf.*, London, U.K., 2005.
- [4] S. Gupta, A. Srinivasamurthy, M. Kumar, H. A. Murthy, and X. Serra, “Discovery of syllabic percussion patterns in tabla solo recordings,” in *Proc. of the 16th Int. Society for Music Information Retrieval Conf.*, Málaga, Spain, 2015.
- [5] S. Deolekar and S. Abraham, “Tree-based classification of tabla strokes,” *Current Science (00113891)*, vol. 115, no. 9, pp. 1724–1731, 2018.
- [6] R. Sarkar, A. Singh, A. Mondal, and S. K. Saha, “Automatic extraction and identification of bol from tabla signal,” in *Advanced Computing and Systems for Security: Volume Five*, R. Chaki, A. Cortesi, K. Saeed, and N. Chaki, Eds. Springer Singapore, 2018, pp. 139–151.
- [7] S. Shete and S. Deshmukh, “Analysis and comparison of timbral audio descriptors with traditional audio descriptors used in automatic tabla bol identification of north Indian classical music,” in *Proc. of the 20th Int. Conf. on Computational Science and Applications*, S. Bhalla, P. Kwan, M. Bedekar, R. Phalnikar, and S. Sirsikar, Eds. Springer Singapore, 2020, pp. 295–307.
- [8] —, “North Indian classical music tabla tala (rhythm) prediction system using machine learning,” in *Advances in Speech and Music Technology*, A. Biswas, E. Wennekes, T.-P. Hong, and A. Wiczorkowska, Eds. Singapore: Springer Singapore, 2021.
- [9] M. Clayton, *Time in Indian Music: Rhythm, Metre, and Form in North Indian Rag Performance*. Oxford University Press, U.K., 2001.
- [10] D. Courtney, *Fundamentals of Tabla*. Sur Sangeet Services, 2013.
- [11] M. Clayton, “Theory and practice of long-form non-isochronous metres,” *Music Theory Online*, vol. 26, no. 1, 2020.
- [12] M. A. Rohit and P. Rao, “Automatic stroke classification of tabla accompaniment in hindustani vocal concert audio,” *To appear in Journal of Acoustical Society of India*, April 2021. [Online]. Available: <https://arxiv.org/abs/2104.09064>
- [13] P. Herrera, A. Yeterian, and F. Gouyon, “Automatic classification of drum sounds: A comparison of feature selection methods and classification techniques,” in *Proc. of the 2nd Int. Conf. on Music and Artificial Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 69–80.
- [14] C. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, “A review of automatic drum transcription,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, 2018.
- [15] R. Vogl and P. Knees, “Mirex submission for drum transcription 2018,” in *Proc. of the 19th Int. Society for Music Information Retrieval Conf.*, Paris, France, 2018.
- [16] C. Jacques and A. Röbel, “Automatic drum transcription with convolutional neural networks,” in *Proc. of the 21th Int. Conf. on Digital Audio Effects*, Aveiro, Portugal, 2018.
- [17] —, “Data augmentation for drum transcription with convolutional neural networks,” in *Proc. of the 27th IEEE European Signal Processing Conf.*, A Coruña, Spain, 2019.
- [18] R. Gowriprasad and K. S. R. Murty, “Onset detection of tabla strokes using LP analysis,” in *Proc. of the 13th IEEE Int. Conf. on Signal Processing and Communications (SPCOM)*, Bangalore, India, 2020.
- [19] R. Vogl, M. Dorfer, G. Widmer, and P. Knees, “Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks,” in *Proc. of the 18th Int. Society for Music Information Retrieval Conf.*, Suzhou, China, 2017.
- [20] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: a new python audio and music signal processing library,” in *Proc. of the 24th ACM Int. Conf. on Multimedia*, 2016.
- [21] K. Choi, G. Fazekas, M. Sandler, and K. Cho, “Transfer learning for music classification and regression tasks,” in *Proc. of the 18th Int. Society for Music Information Retrieval Conf.*, Suzhou, China, 2017.
- [22] J. Pons, J. Serrà, and X. Serra, “Training neural audio classifiers with few data,” in *Proc. of the 44th IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Brighton, U.K., 2019.
- [23] A. Diment and T. Virtanen, “Transfer learning of weakly labelled audio,” in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, U.S.A., 2017.
- [24] S. Zhang, C.-T. Do, R. Doddipatla, and S. Renals, “Learning noise invariant features through transfer learning for robust end-to-end speech recognition,” in *Proc. of the 45th IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Barcelona, Spain, 2020.

- [25] P. G. Shivakumar and P. Georgiou, “Transfer learning from adult to children for speech recognition: Evaluation, analysis and recommendations,” *Computer Speech & Language*, vol. 63, 2020.
- [26] J. Schlüter and S. Böck, “Improved musical onset detection with convolutional neural networks,” in *Proc. of the 39th IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Florence, Italy, 2014.
- [27] P. Brossier, J. P. Bello, and M. D. Plumbley, “Fast labelling of notes in music signals,” in *Proc. of the 5th Int. Society for Music Information Retrieval Conf.*, Barcelona, Spain, 2004.
- [28] S. Yong, S. Choi, and J. Nam, “Pytsmod: A python implementation of time-scale modification algorithms,” in *Extended Abstracts for the Late-Breaking Demo Session of the 21st Int. Society for Music Information Retrieval Conf.*, Montréal, Canada, 2020.
- [29] D. FitzGerald, “Harmonic/percussive separation using median filtering,” in *Proc. of the 13th Int. Conf. on Digital Audio Effects*, Graz, Austria, 2010.
- [30] W.-N. Hsu, Y. Zhang, and J. Glass, “Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation,” in *IEEE Automatic Speech Recognition and Understanding Workshop*, Okinawa, Japan, 2017.
- [31] J. Schlüter and T. Grill, “Exploring data augmentation for improved singing voice detection with neural networks,” in *Proc. of the 16th Int. Society for Music Information Retrieval Conf.*, Málaga, Spain, 2015.
- [32] P. López-Serrano, C. Dittmar, Y. Özer, and M. Müller, “NMF toolbox: Music processing applications of non-negative matrix factorization,” in *Proc. of the 22nd Int. Conf. on Digital Audio Effects*, Birmingham, U.K., 2019.
- [33] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, “mir_eval: A transparent implementation of common MIR metrics,” in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, Taipei, Taiwan, 2014.

A CONTEXTUAL LATENT SPACE MODEL: SUBSEQUENCE MODULATION IN MELODIC SEQUENCE

Taketo Akama

Sony Computer Science Laboratories, Tokyo, Japan
taketo.akama@sony.com

ABSTRACT

Some generative models for sequences such as music and text allow us to edit only subsequences, given surrounding context sequences, which plays an important part in steering generation interactively. However, editing subsequences mainly involves randomly resampling subsequences from a possible generation space. We propose a contextual latent space model (CLSM) in order for users to be able to explore subsequence generation with a sense of direction in the generation space, e.g., interpolation, as well as exploring variations—semantically similar possible subsequences. A context-informed prior and decoder constitute the generative model of CLSM, and a context position-informed encoder is the inference model. In experiments, we use a monophonic symbolic music dataset, demonstrating that our contextual latent space is smoother in interpolation than baselines, and the quality of generated samples is superior to baseline models. The generation examples are available online.¹

1. INTRODUCTION

Deep generative models permit sequences of decent quality to be generated such as music, lyrics, or text, where standard models generate sequences by sampling from left to right. However, to make creative works in human-machine collaborative settings, controllability—such as modifying unsatisfactory portions with specified intentions—should be improved.

Two major model classes of controllability are (i) latent space models [1–6] and (ii) positional constraint models [7–11]. Latent space models enable us to obtain variations or morphing/interpolations between generated sequences. Positional constraint models, on the other hand, allow us to resample a subsequence without changing the rest of the sequence (*context sequences*), despite the fact that subsequences are sampled randomly and cannot be controlled with morphing/interpolation or variations. Each class of

¹<https://contextual-latent-space-model.github.io/demo/>

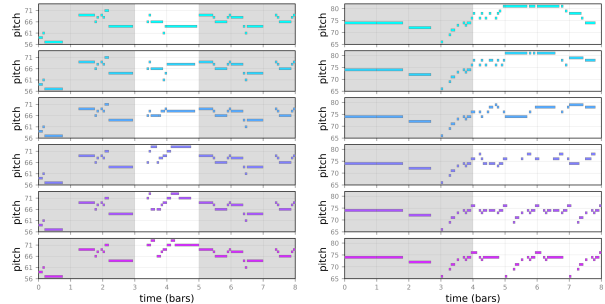


Figure 1: Contextual Interpolation Examples. Shaded regions are contexts. For both left and right figures, our CLSM first generates top and bottom melodies under constraints of contexts, and then it generates middle four interpolated points in way that is consistent with contexts. CLSM with $\beta = 0.012$ is used.

models has its own benefits for making generation systems flexible.

Can we build a hybrid model that enjoys the best of both worlds as a step towards multifaceted controllability? We propose a contextual latent space model (CLSM) that allows for positional constraints while at the same time enables latent space exploration such as interpolation or variation. An example usage of CLSM’s interpolation is narrowing down the candidates of generated subsequences given context sequences. CLSM variation can be used for obtaining minor modifications of subsequences selected among generated ones, given context sequences.

Our approach is based on the framework of variational inference, where our CLSM is composed of prior and decoder models for the generative model and an encoder model for the inference model. The prior model of CLSM outputs a latent distribution given context sequences. We refer to the support of the distribution as the *contextual latent space*. The decoder model of CLSM generates subsequences that fit in with the context, given corresponding points in the contextual latent space. Finally the encoder model infers the latent space distribution given the entire sequence.

We show the effectiveness of our approach using monophonic sequences in the Lakh MIDI dataset, a large symbolic music dataset [12]. Compared with the baseline methods, our CLSM achieves better performance in terms of the smoothness of the latent space and negative log-

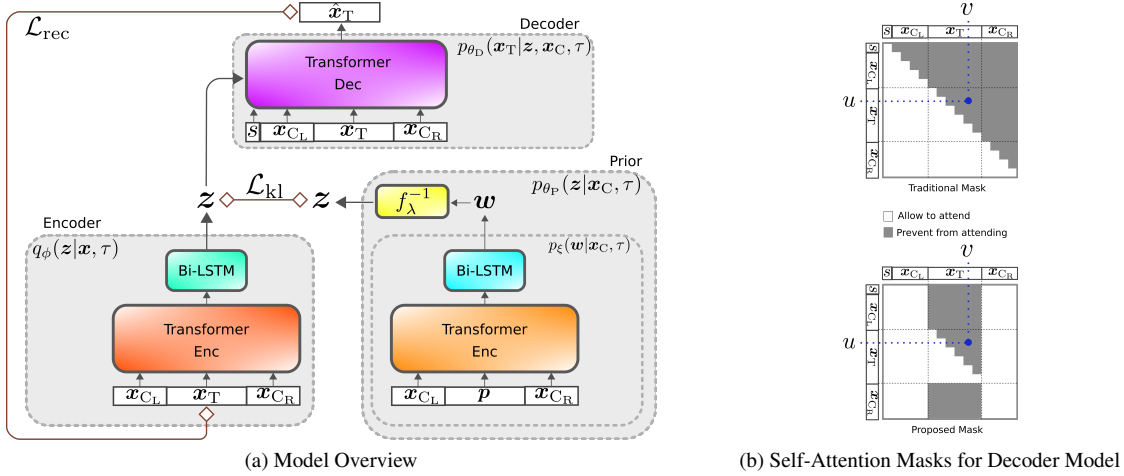


Figure 2: Schematic Diagram of Contextual Latent Space Model (CLSM). (a) Linear or MLP layers are omitted for brevity. (b) Masks indicate whether position u attends to position v .

likelihood of the generated samples. In a listening test, participants favored the generated samples of our CLSM more than those of the baselines.

Our contributions are (I) proposing a problem setting for learning a contextual latent space and providing its solution, (II) proposing novel architectures for the model, e.g., a masking strategy for the decoder and combinations of a transformer and LSTM for the encoder and prior, (III) proposing normalizing flows [13] for conditional priors in order to learn complex conditional priors, (IV) proposing an *interpolation edit distance ratio* to quantitatively assess the smoothness of latent space, and (V) demonstrating reasonable performance with our CLSM in an application for symbolic music generation.

2. METHODOLOGY

2.1 Problem Scenario

Let us consider an i.i.d dataset $\mathcal{D} = \{\mathbf{x}^{(i)} = (x_1^i, \dots, x_K^i) \in \mathcal{A}^K\}_{i=1}^N$ of a sequence of symbols $x_k \in \mathcal{A}$ of length K , where \mathcal{A} is the alphabet set of symbols. We partition each sequence \mathbf{x} into three subsequences \mathbf{x}_{C_L} , \mathbf{x}_T , and \mathbf{x}_{C_R} , such that $\mathbf{x} = \mathbf{x}_{C_L} \oplus \mathbf{x}_T \oplus \mathbf{x}_{C_R}$, where \oplus denotes the concatenation of sequences. We refer to subsequences \mathbf{x}_{C_L} and \mathbf{x}_{C_R} as *context sequences* and subsequence \mathbf{x}_T as the *target sequence*. Let τ denote a variable representing a set of indexes of the target sequence such that $\tau = \{|\mathbf{x}_{C_L}| + 1, |\mathbf{x}_{C_L}| + 2, \dots, |\mathbf{x}_{C_L}| + |\mathbf{x}_T|\} \in \mathcal{T}$, where \mathcal{T} is a set of all sets of indexes we would like to model with. For notational simplicity, we introduce the shorthand $\mathbf{x}_C = \{\mathbf{x}_{C_L}, \mathbf{x}_{C_R}\}$.

Our goal is to train a generative model with its generative process being (1) $\tilde{z} \sim p(z|\mathbf{x}_C, \tau)$ and (2) $\tilde{\mathbf{x}}_T \sim p(\mathbf{x}_T|\tilde{z}, \mathbf{x}_C, \tau)$, where $z \in \mathcal{Z} \subset \mathbb{R}^{d_z}$ captures the variability of \mathbf{x}_T given \mathbf{x}_C and τ . We would also like some distance in the latent space of the prior model to represent the similarity of \mathbf{x}_T so that the model can be used for e.g., morphing/interpolation or variation generation.

2.2 Model

Fig. 2 is a schematic illustration of our model. Our proposed approach is training a generative model by maximizing the marginal log-likelihood $\log p_\theta(\mathbf{x}_T|\mathbf{x}_C, \tau) = \log \int p_{\theta_D}(\mathbf{x}_T|z, \mathbf{x}_C, \tau)p_{\theta_P}(z|\mathbf{x}_C, \tau)dz$. Since its computation is intractable in the general case, we introduce the approximate posterior $q_\phi(z|\mathbf{x}, \tau)$ to derive the evidence lower bound (ELBO) [14]. Formally,

$$\begin{aligned} \log p_\theta(\mathbf{x}_T|\mathbf{x}_C, \tau) &= \log \int p_{\theta_D}(\mathbf{x}_T|z, \mathbf{x}_C, \tau)p_{\theta_P}(z|\mathbf{x}_C, \tau)dz \\ &= \log \int q_\phi(z|\mathbf{x}, \tau) \frac{p_{\theta_D}(\mathbf{x}_T|z, \mathbf{x}_C, \tau)p_{\theta_P}(z|\mathbf{x}_C, \tau)}{q_\phi(z|\mathbf{x}, \tau)} dz \\ &\geq \int q_\phi(z|\mathbf{x}, \tau) \log \frac{p_{\theta_D}(\mathbf{x}_T|z, \mathbf{x}_C, \tau)p_{\theta_P}(z|\mathbf{x}_C, \tau)}{q_\phi(z|\mathbf{x}, \tau)} dz \\ &= \mathcal{L}_{\text{rec}} - \mathcal{L}_{\text{kl}}, \end{aligned} \quad (1)$$

where

$$\mathcal{L}_{\text{rec}} = \mathbb{E}_{q_\phi(z|\mathbf{x}, \tau)} [\log p_{\theta_D}(\mathbf{x}_T|z, \mathbf{x}_C, \tau)], \quad (2)$$

$$\mathcal{L}_{\text{kl}} = \text{KL}(q_\phi(z|\mathbf{x}, \tau) \| p_{\theta_P}(z|\mathbf{x}_C, \tau)). \quad (3)$$

In practice, we introduce weighting factors in ELBO [15]. Then, our optimization problem is:

$$\max_{\theta, \phi} \mathbb{E}_{\mathbf{x} \in \mathcal{D}} \mathbb{E}_{\tau \in \mathcal{T}} \left[\frac{1}{|\mathbf{x}_T|} \mathcal{L}_{\text{rec}} - \beta \mathcal{L}_{\text{kl}} \right], \quad (4)$$

where $\frac{1}{|\mathbf{x}_T|}$ is a normalizing factor, and β is a balancing factor of the two terms. The specific choices of β are explained in Sec. 3.4.

Since the conditional prior is generally multimodal and complex, we propose modeling the conditional prior using normalizing flows [13]:

$$p_{\theta_P}(z|\mathbf{x}_C, \tau) = p_\xi(\mathbf{w}|\mathbf{x}_C, \tau) \left| \det \left(\frac{\partial \mathbf{w}}{\partial z} \right) \right|, \quad (5)$$

$$\mathbf{w} = f_\lambda(z) \in \mathcal{W} \subset \mathbb{R}^{d_z}, \quad (6)$$

where $f_\lambda: \mathcal{Z} \rightarrow \mathcal{W}$ is an invertible function, and $p_\xi(\cdot|\mathbf{x}_C, \tau)$ is a Gaussian distribution. We choose to use affine-coupling layers for f_λ , which was proposed for real-valued non-volume preserving (realNVP) [16].

The Gaussian distribution and the categorical distribution are used for the encoder model $q_\phi(\mathbf{z}|\mathbf{x}, \tau)$ and the decoder model $p_{\theta_D}(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_C, \tau)$, respectively.

The network architectures for parametrizing each distribution are explained in Sec. 2.4.

2.3 Applications

2.3.1 Contextual Interpolation

Given $\tilde{\mathbf{z}}^{(1)}, \tilde{\mathbf{z}}^{(2)} \sim p_{\theta_P}(\mathbf{z}|\mathbf{x}_C, \tau)$, we provide a procedure for generating contextual interpolations between $\tilde{\mathbf{x}}^{(1)}$ and $\tilde{\mathbf{x}}^{(2)}$, where $\tilde{\mathbf{x}}^{(i)} = \mathbf{x}_{C_L} \oplus \tilde{\mathbf{x}}_T^{(i)} \oplus \mathbf{x}_{C_R}$ with $\tilde{\mathbf{x}}_T^{(i)} \sim p_{\theta_D}(\mathbf{x}_T|\tilde{\mathbf{z}}^{(i)}, \mathbf{x}_C, \tau)$ for $i = 1, 2$.

First, the interpolated latent vector $\tilde{\mathbf{z}}(\alpha)$ with a blending ratio of $\alpha \in [0, 1]$ is given by

$$\tilde{\mathbf{z}}(\alpha) = f_\lambda^{-1} \left((1 - \alpha)f_\lambda(\tilde{\mathbf{z}}^{(1)}) + \alpha f_\lambda(\tilde{\mathbf{z}}^{(2)}) \right), \quad (7)$$

which means that linear interpolation is performed in the \mathcal{W} space, but not in the \mathcal{Z} space, achieving non-linear interpolation in the \mathcal{Z} space. Then, $\tilde{\mathbf{z}}(\alpha)$ is decoded and concatenated with the context sequence, yielding $\tilde{\mathbf{x}}(\alpha) = \mathbf{x}_{C_L} \oplus \tilde{\mathbf{x}}_T(\alpha) \oplus \mathbf{x}_{C_R}$ with

$$\tilde{\mathbf{x}}_T(\alpha) \sim p_{\theta_D}(\mathbf{x}_T|\tilde{\mathbf{z}}(\alpha), \mathbf{x}_C, \tau). \quad (8)$$

2.3.2 Contextual Variation

Given $\tilde{\mathbf{z}} \sim p_{\theta_P}(\mathbf{z}|\mathbf{x}_C, \tau)$, we provide a procedure for generating contextual variations of $\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \mathbf{x}_{C_L} \oplus \tilde{\mathbf{x}}_T \oplus \mathbf{x}_{C_R}$ with $\tilde{\mathbf{x}}_T \sim p_{\theta_D}(\mathbf{x}_T|\tilde{\mathbf{z}}, \mathbf{x}_C, \tau)$. Letting $\delta \in \mathbb{R}_{\geq 0}^{d_z}$ and $\epsilon \in \mathbb{R}^{d_z}$ be a scaling factor of the variation amount and sampled noise from a normal distribution $\mathcal{N}(\mathbf{0}, \Sigma)$ with Σ denoting the covariance matrix of $p_\xi(\mathbf{w}|\mathbf{x}_C, \tau)$, a variation of the latent vector $\tilde{\mathbf{z}}$ is given by $\tilde{\mathbf{z}}(\delta) = f_\lambda^{-1}(f_\lambda(\tilde{\mathbf{z}}) + \delta\epsilon)$. Then, $\tilde{\mathbf{z}}(\delta)$ is decoded and concatenated in the same manner as Sec. 2.3.1.

2.4 Network Architecture

2.4.1 Encoder Model

As main architectures, we employ a two-layer “transformer encoder” with relative attention [17, 18], followed by a two-layer bidirectional LSTM network (Bi-LSTM) [19, 20]. \mathbf{x} is first embedded and added by a positional embedding before being inputted to the transformer. The transformer uses no masks and a sequence of vectors $\mathbf{E} = (e_1, \dots, e_K)$ is outputted. Let \mathbf{E}_T be a subsequence of \mathbf{E} , consisting of \mathbf{E} ’s elements, whose indexes are in τ , i.e., $\mathbf{E}_T = (e_{|\mathbf{x}_{C_L}|+1}, \dots, e_{|\mathbf{x}_{C_L}|+|\mathbf{x}_T|})$. Only the subsequence \mathbf{E}_T is fed to the Bi-LSTM. Let \mathbf{h}_l and \mathbf{h}_r denote the last outputs of the Bi-LSTM. They are concatenated to be fed to two multi layer perceptrons (MLPs; each for the mean and covariance of the normal

distribution), yielding the encoder model $q_\phi(\mathbf{z}|\mathbf{x}, \tau) = \mathcal{N}(\mathbf{z}|\text{MLP}(\mathbf{h}_l \oplus \mathbf{h}_r), \text{diag}(\frac{1}{2} \exp(\text{MLP}(\mathbf{h}_l \oplus \mathbf{h}_r))))$. The MLPs consist of two layers with a SELU activation in between [21].

Concerning the hyper-parameters for the “transformer encoder,” the token embedding size, the hidden size, the number of heads, and the dropout rate are set to 128, 256, 8, and 0.1, respectively. The hidden size and the dropout rate for the Bi-LSTM are set to 256 and 0.1, respectively. The hidden size of the MLP and the number of dimensions of \mathbf{z} are set to 512 and 128, respectively.

2.4.2 Prior Model

As can be seen in Eq. 5 and Eq. 6, $p_\xi(\mathbf{w}|\mathbf{x}_C, \tau)$ and $f_\lambda(\mathbf{z})$ need to be defined for the prior model.

For $p_\xi(\mathbf{w}|\mathbf{x}_C, \tau)$, as with the encoder model, a two-layer “transformer encoder” is followed by a Bi-LSTM. Unlike the encoder model, we replace each of the elements in the target sequence \mathbf{x}_T with a *positional constraint symbol* p . In other words, $\mathbf{x}_{C_L} \oplus \mathbf{p} \oplus \mathbf{x}_{C_R}$ is fed to the “transformer encoder”, where $\mathbf{p} = (p, p, \dots, p)$ is a sequence of positional constraint symbols and $|\mathbf{p}| = |\mathbf{x}_T|$. The hyper-parameters for the “transformer encoder,” the Bi-LSTM, and the number of dimensions of \mathbf{z} are set to the same values as in Sec. 2.4.1.

To parameterize $f_\lambda(\mathbf{z})$, four-layer affine-coupling layers are employed. Each of the scale and bias networks of the affine-coupling layers consists of a three-layer MLP, where each hidden size is 256, and the activations are leaky ReLUs with a negative slope of 0.01. For each of the scale networks, a tanh activation is used after the last linear layer.

2.4.3 Decoder Model

As a main architecture, we employ a two-layer “transformer decoder” with relative attention. Let s be a symbol representing the start of a sequence. The concatenation $s \oplus \mathbf{x}$ is embedded and added by positional embeddings before being inputted to the transformer.

We propose using an effective encoder-decoder attention mechanism for the latent space. Each latent vector $\tilde{\mathbf{z}}$ sampled from the encoder model is first fed to a linear layer to map $\tilde{\mathbf{z}} \in \mathbb{R}^{d_z}$ to $\tilde{\mathbf{z}}' \in \mathbb{R}^{d_z l_z}$, which is reshaped to form $\tilde{\mathbf{Z}} \in \mathbb{R}^{d_z \times l_z}$, a sequence of l_z vectors, where the dimensionality of each vector is d_z . The sequence $\tilde{\mathbf{Z}}$ is then attended to by the transformer by means of encoder-decoder attention. We set $l_z = 4$ in experiments.

We propose a masking strategy for modeling the decoder, as illustrated in the bottom of Fig. 2b. The positions whose inputs are s , \mathbf{x}_{C_L} , and \mathbf{x}_{C_R} are allowed to attend to positions except those of \mathbf{x}_T . On the other hand, positions whose inputs are \mathbf{x}_T are allowed to attend to positions except the future positions of \mathbf{x}_T .

Let the output of the transformer denote $\mathbf{D} = (d_1, \dots, d_K)$. Let \mathbf{D}_T be a subsequence of \mathbf{D} , consisting of \mathbf{D} ’s elements whose indexes are in τ , i.e., $\mathbf{D}_T = (d_{|\mathbf{x}_{C_L}|+1}, \dots, d_{|\mathbf{x}_{C_L}|+|\mathbf{x}_T|})$. Let $\mathbf{x}_T[i]$ and $\mathbf{D}_T[i]$ denote

the i th elements of \mathbf{x}_T and \mathbf{D}_T , respectively. Then the decoder model is defined as follows: $p_{\theta_D}(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_C, \tau) = \prod_{i=1}^{|\mathbf{x}_T|} \text{Cat}(\mathbf{x}_T[i]|\text{Softmax}(\text{Linear}(\mathbf{D}_T[i])))$, where Cat and Linear denote a categorical distribution and a linear layer respectively. The hyper-parameters of the ‘‘transformer decoder’’ are set to be equal to those of the ‘‘transformer encoder’’ in Sec. 2.4.1.

3. EXPERIMENTAL SETUP

3.1 Dataset

We created datasets from LMD-matched of the Lakh MIDI dataset [12], comprising 45,129 files matched to the song identity entries in the Million Song Dataset [22]. Each song has one or several different versions of MIDI files. We first extracted files with a 4/4 time signature, used the accompanying tempo information to determine beat locations, and quantized each beat into 4. We then split the song identities into a proportion of 11:1:6:1:1 to create train-1, validation-1, train-2, validation-2, and test datasets, respectively. The train-1 and validation-1 datasets were for training the proposed and baseline models, whereas the train-2 and validation-2 datasets were for training evaluation models. The test dataset was for input sequences when evaluating models. We filtered out non-monophonic tracks, bass or drum tracks, and tracks outside the pitch range of [55, 84]. We conducted data augmentation by transposing tracks to all possible keys if the transposed tracks stayed within the pitch range of [55, 84]. We retrieved 8-bar sliding windows (with a stride of 1 bar) from each track followed by filtering out windows that had more than one bar of consecutive rests. For encoding musical sequences, we adopted the melodic-rhythmic encoding proposed in [9], where we used the pitch of musical notes as symbols ‘‘55’’, ‘‘...’’, ‘‘84’’ and used ‘‘R’’ to represent a rest symbol. We added an extra symbol, ‘‘_’’ representing that a note is held and not replayed.

3.2 Baseline Methods

3.2.1 VAE

We trained a VAE [14], in which a two-layered Bi-LSTM and LSTM were used for the encoder and decoder, respectively. The decoder, encoder, and prior models used the categorical, normal, and standard normal distribution, respectively. The optimization problem is

$$\max_{\psi, \omega} \mathbb{E}_{\mathbf{x} \in \mathcal{D}} \left[\frac{1}{|\mathbf{x}|} (\mathcal{L}_{\text{rec}}^{\text{VAE}} - \gamma \mathcal{L}_{\text{kl}}^{\text{VAE}}) \right], \quad (9)$$

where $\mathcal{L}_{\text{rec}}^{\text{VAE}} = \mathbb{E}_{q_{\omega}(\mathbf{z}|\mathbf{x})} [\log p_{\psi}(\mathbf{x}|\mathbf{z})]$, $\mathcal{L}_{\text{kl}}^{\text{VAE}} = \text{KL}(q_{\omega}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$, and $\frac{1}{|\mathbf{x}|}$ is a normalizing factor.

Given \mathbf{x}_C , τ , and $\tilde{\mathbf{z}}^{(1)}, \tilde{\mathbf{z}}^{(2)} \sim p(\mathbf{z})$, interpolation is conducted as follows. First, interpolated latent vector $\tilde{\mathbf{z}}(\alpha)$ is given by

$$\tilde{\mathbf{z}}(\alpha) = (1 - \alpha)\tilde{\mathbf{z}}^{(1)} + \alpha\tilde{\mathbf{z}}^{(2)}. \quad (10)$$

Then, $\tilde{\mathbf{z}}(\alpha)$ is decoded and concatenated with the context sequence: $\tilde{\mathbf{x}}(\alpha) = \mathbf{x}_{C_L} \oplus \tilde{\mathbf{x}}_T(\alpha) \oplus \mathbf{x}_{C_R}$ with

$$\tilde{\mathbf{x}}_T(\alpha) \sim p_{\psi}(\mathbf{x}_T|\tilde{\mathbf{z}}(\alpha), \mathbf{x}_{C_L}), \quad (11)$$

where $p_{\psi}(\mathbf{x}_T|\mathbf{z}, \mathbf{x}_{C_L})$ can be immediately obtained from the autoregressive decoder model of VAE. Note that the proposed CLSM has advantages over VAE in terms of probabilistic dependencies (i) the decoder model of VAE does not have dependencies on the right context \mathbf{x}_{C_R} , and (ii) the prior model of VAE does not have dependencies on either the left context \mathbf{x}_{C_L} or the right context \mathbf{x}_{C_R} . The property of (i) motivates us to separately quantify the performance of models in two cases: (1) the case where only the left context exists, and (2) otherwise. In Sec. 4.1, Sec. 4.2, and Fig. 3b, we report the performance of these two cases separately.

Random generation was conducted as follows. First, a latent vector was sampled from the prior distribution. Then, $\tilde{\mathbf{z}}$ was decoded and concatenated with context sequences in the same manner as the interpolation.

The hyper-parameters of the LSTMs in VAE were set to be equal to those of the LSTMs in CLSM.

3.2.2 ARNN

Anticipation RNN (ARNN) [9] is a sequence generation model with positional constraints. Two-layered LSTMs were used for both Token-RNN and Constraint-RNN. The hyper-parameters of the LSTMs were set to be equal to those of the LSTMs in CLSM.

3.3 2D Plane for Comparing CLSM and VAE

The balancing factors β and γ of CLSM and VAE consist in adjusting the trade-off between the reconstruction accuracy and other model performances. Which balancing factors of CLSM correspond to which ones of VAE? It is natural to compare models of similar reconstruction accuracies or compare models of similar performances (except reconstruction accuracies). Therefore, in Sec. 4.1, Sec. 4.2, and Fig. 3b, we plot reconstruction accuracies versus other performance metrics in 2D planes, where the more the plotted point is in the upper left corner, the better it is. As discussed in Sec. 3.2.1, CLSM and VAE have dependencies at least on the left context. To make the reconstruction accuracies of the CLSM and VAE mean basically identical, we used the *left-contextual reconstruction accuracy*, which is the average of the reconstruction accuracies of sequences where target sequences end with index $|\mathbf{x}|$, i.e., there is no right context and only the left context exists.

3.4 Training Settings

For all models, teacher forcing was used, the batch size was set to 64, and training was conducted for 2 epochs, when the losses converged. The Adam optimizer [23] was used for all models, with the parameters $(\alpha, \beta_1, \beta_2) = (0.0005, 0.9, 0.999)$. For CLSM or VAE, we conducted

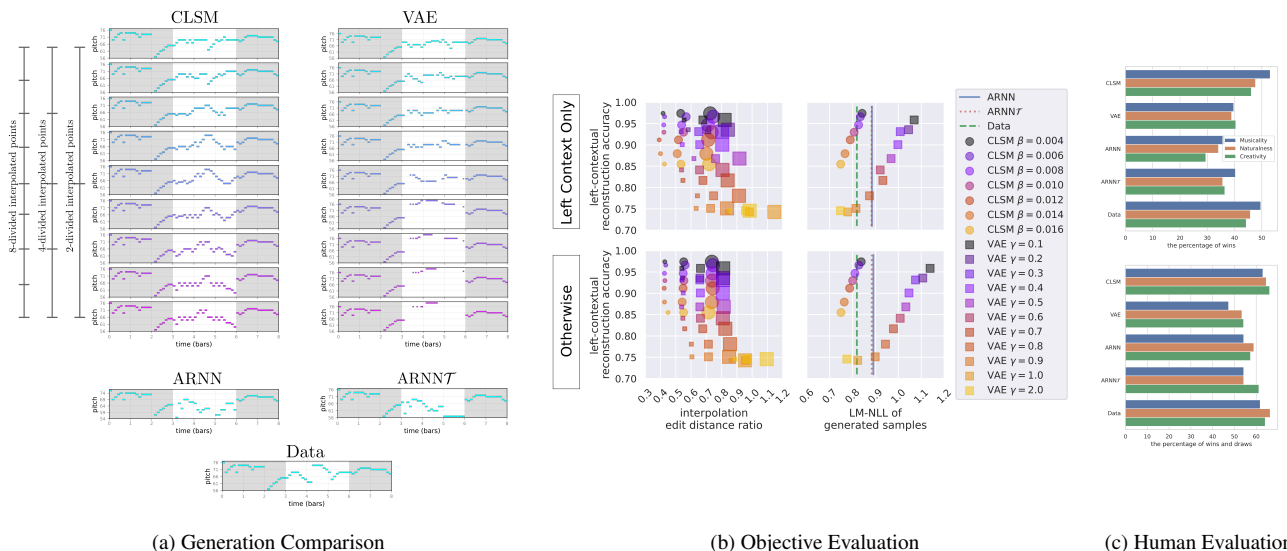


Figure 3: Generation Comparison and Experimental Results. (a) Shaded regions are contexts. For CLSM and VAE, $\beta = 0.012$ and $\gamma = 0.4$ are used. (b) For evaluating smoothness (to left), small, medium, and large marker are plots for number of divisions in interpolation $J = 8, 4, 2$, respectively. See Sec. 4.1 and 4.2. (c) See Sec. 4.3.

KL-annealing linearly from $\beta = 0$ or $\gamma = 0$ for 2 epochs [1].

3.4.1 CLSM (Ours)

At every iteration of the training, indexes of the target sequence $\tau = \{|\mathbf{x}_{C_L}| + 1, |\mathbf{x}_{C_L}| + 2, \dots, |\mathbf{x}_{C_L}| + |\mathbf{x}_T\} \in \mathcal{T}$ should be sampled. In the experiments, we chose to use either 1, 2, 3, or 4 bars as the target sequence length and used a stride of 1 bar for the starting index of the target sequence. Precisely, we sampled (i) $|\mathbf{x}_T|$ uniformly from $\{16, 32, 48, 64\}$ and then (ii) sampled $|\mathbf{x}_{C_L}|$ uniformly from $\{0, 16, 32, \dots, 128 - |\mathbf{x}_T|\}$. Note that 128 corresponds to 8 bars (the sequence length) and that 16 corresponds to 1 bar. The balancing factor β in Eq. 4 is set to $\{0.004, 0.006, \dots, 0.016\}$ in Secs. 4.1, 4.2 and 0.012 in Sec. 4.3. The expectations of the two terms in Eq. 4 were approximated with one sample from the encoder model.

3.4.2 VAE (Baselines)

The balancing factor γ in Eq. 9 is set to $\{0.1, 0.2, \dots, 1.0, 2.0\}$ in Secs. 4.1, 4.2 and 0.4 in Sec. 4.3. The expectations of the reconstruction loss were approximated with one sample from the encoder model. The KL loss term was computed analytically.

3.4.3 ARNN (Baseline)

The vanilla ARNN is capable of imposing constraints of any positions. Since it would be possible that restricting constraints to those of our \mathcal{T} would be advantageous when evaluated over \mathcal{T} , we also trained and evaluated this model, which we refer to as ARNNT.

3.5 Generation Settings

For CLSM and VAE, each element of sequences was sampled by applying the argmax operation to the categori-

cal distributions of the decoders. For ARNN, multinomial sampling with a temperature of 1.0 was used.

4. EXPERIMENTS AND RESULTS

4.1 Smoothness Analysis in Latent Space

To assess the smoothness of our latent space, we propose the *interpolation edit distance ratio* $R(J)$, which is the ratio of the distance between adjacent interpolated points (sequences) to the distance of interpolation end points (sequences). Formally, $R(J)$ is the normalized average edit distance $d_{\text{edit}}(\cdot, \cdot)$ of adjacent points in J -divided interpolated points:

$$R(J) = \mathbb{E} \left[\frac{\sum_{j=0}^{J-1} d_{\text{edit}}(\tilde{\mathbf{x}}_T(\frac{j}{J}), \tilde{\mathbf{x}}_T(\frac{j+1}{J}))}{D(J-n)} \right], \quad (12)$$

where $\tilde{\mathbf{x}}_T(\cdot)$ is defined by Eqs. 7, 8 for CLSM and Eqs. 10, 11 for VAE, while D is the edit distance between end points defined by $D = d_{\text{edit}}(\tilde{\mathbf{x}}_T(0), \tilde{\mathbf{x}}_T(J))$. Here, the expectation is approximated by sampling $\tilde{\mathbf{z}}^{(1)}, \tilde{\mathbf{z}}^{(2)} \sim p_{\theta_F}(\mathbf{z}|\mathbf{x}_C, \tau)$ for CLSM and $\tilde{\mathbf{z}}^{(1)}, \tilde{\mathbf{z}}^{(2)} \sim p(\mathbf{z})$ for VAE, where 1K samples of \mathbf{x} are uniformly sampled from the test dataset, and, for each \mathbf{x} , τ is uniformly sampled from \mathcal{T} . Since the edit distance $d_{\text{edit}}(\tilde{\mathbf{x}}_T(\frac{j}{J}), \tilde{\mathbf{x}}_T(\frac{j+1}{J}))$ sometimes becomes zero, we excluded cases through division by $J-n$ instead of J as in Eq. 12, where n is the number of edit distances that are zero among $j = 0, 1, \dots, J-1$. We also excluded cases where $D(J-n) = 0$. The left scatter plot of Fig. 3b shows a comparison of CLSM and VAE. The small, medium, and large markers are plots for the number of divisions in interpolation $J = 8, 4, 2$, respectively. For each J , CLSM performed better than VAE. The plots of VAE $J = 8$ overlap with those of CLSM $J = 4$, indicating that even 4-divided interpolation of CLSM was as smooth as 8-divided interpolation of VAE. The results

are similar in the two cases of “left context only” and “otherwise.”

4.2 NLL Evaluation of Generated Samples

To evaluate the quality of generated samples, we computed the negative log-likelihood (NLL) of the generated samples. NLL was computed by using a separately trained vanilla transformer language model (LM) that had two layers and was autoregressive. The LM was trained by using the train-2 and validation-2 datasets defined in Sec. 3.1. The samples to be evaluated were the same as the 8-divided interpolated points in Sec. 4.1. Since ARNN is not capable of interpolation, only a random sampling of two points was performed per each context for ARNN. The right scatter plot of Fig. 3b shows a comparison of the CLSM, VAE, ARNNs, and the test dataset (Data). CLSM outperformed VAE by a large margin when we compared models that were close in terms of reconstruction accuracy or NLL—a reasonable strategy of comparison as we discuss in Sec. 3.3. Compared with the ARNNs, the performance of CLSM was superior in all settings. The results of VAE were better in the case of “left context only” than in the case of “otherwise.” In contrast, for the CLSM and ARNNs, there were only slight differences between the two cases. This empirically demonstrates that the decoder model of VAE has dependencies only on the left context but not on the right context, as we discuss in Sec. 3.2.1.

4.3 Human Evaluation

To further assess the quality of sequences generated by each model, we conducted listening tests using Amazon Mechanical Turk. We sampled 32 sequences from the test dataset, for which context positions were randomly sampled, and the target sequence lengths were sampled from 1-4 bars. We conducted a pair-wise comparison of the generation results of each model using the same context sequences. We considered all possible combinations, yielding 320 pair-wise comparisons. The order within each pair was randomized. We chose to use $\beta = 0.012$ for CLSM, since the performance trade-offs are well balanced according to the results in Sec. 4.1 and Sec. 4.2. As discussed in Sec. 3.3 since it is reasonable to choose a VAE model with a similar reconstruction accuracy to that of CLSM, we chose $\gamma = 0.4$ for VAE. Participants with different levels of musical expertise were asked to rate “which music is better in terms of musicality, naturalness, and creativity” on a Likert scale. Fig. 3c shows a comparison of the CLSM, VAE, ARNNs, and the test dataset (Data). CLSM and Data performed the best in terms of the percentage of wins (Fig. 3c, top) as well as the percentage of wins and draws (Fig. 3c, bottom). Interestingly, the ARNNs tended to outperform VAE when the number of draws was included, whereas VAE tended to outperform the ARNNs when only the number of wins was considered. This might indicate that the performance of VAE tends to be extreme.

5. RELATED WORK

T-CVAE is a transformer-based conditional VAE model for story completion [24]. VAEAC [25] is a CNN- or MLP-based VAE that enables us to impose any positional constraints. Although their probabilistic frameworks are similar to ours, the models and architectures are quite different. Unlike their models, CLSM is demonstrated to perform interpolation in the latent space. Moreover, the data domain of T-CVAE is story text, and the domains of VAEAC are image and feature classification/regression datasets, while ours is for sequence datasets and experimented on music.

Although contexts are not considered for latent variables, there are several works that use transformers for learning a global latent variable for sequences using AE or VAE. For text-style transfer, a “transformer encoder” outputs are all fed to GRU to yield a latent vector, which is attended to by a decoder [26]. To learn the styles of piano performances, a “transformer encoder” outputs are summed to be attended to by a decoder [27]. In OPTIMUS for sentence modulation [28] and INSET [29] for sentence infilling, a CLS token is additionally fed to a “transformer encoder,” and the output at the position of the CLS yields a latent vector, which is fed to a decoder either by self-attention and/or by being added to word embeddings of the decoder (OPTIMUS) or by being inputted as the first token (INSET).

For language processing, the authors of UniLM propose seq-to-seq LM, where they divide a whole sequence into first and second segments [30]. The self-attention masks are bidirectional and unidirectional for the first and second segments, respectively. Our decoder mask is different in that it divides a sequence into three segments, where the length of the first and third segments can be zero during the training or inference phase. Also, the training procedure of them is BERT-like, which is different from ours [31].

RealNVP has been used for the prior in VAE in order to improve the performance of VAE [32, 33]. However, these works are not only in the domain of images but also use non-conditional priors, which differs from ours.

6. CONCLUSION

We proposed a contextual latent space model (CLSM), in which the left and/or right contexts of sequences can be constrained to generate interpolations or variations. A context-informed prior and decoder constitute the generative model of CLSM and a context position-informed encoder is the inference model.

The latent space of CLSM was quantitatively shown to be smoother than baselines. Furthermore, the generation fidelity was demonstrated to be superior to the baseline methods. It would be useful to apply our approach to other data domains such as polyphonic music, lyrics, or text. The benefits of the latent space model are not only enabling interpolations and variations but also enabling transformations of attributes or style transfer. It would be desirable to extend our approach to these kinds of applications.

7. REFERENCES

- [1] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.
- [2] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of the 35th International Conference on Machine Learning*, 2018.
- [3] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019.
- [4] A. Pati and A. Lerch, "Attribute-based Regularization of Latent Spaces for Variational Auto-Encoders," *Neural Computing and Applications*, 2020.
- [5] H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *ISMIR*, 2020.
- [6] T. Akama, "Connective fusion: Learning transformational joining of sequences with application to melody creation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference. ISMIR*, 2020.
- [7] B. Uria, I. Murray, and H. Larochelle, "A deep and tractable density estimator," in *Proceedings of the 31st International Conference on Machine Learning*, 2014.
- [8] A. Wang and K. Cho, "BERT has a mouth, and it must speak: BERT as a markov random field language model," *CoRR*, vol. abs/1902.04094, 2019.
- [9] G. Hadjeres and F. Nielsen, "Anticipation-rnn: enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, vol. 32, no. 4, pp. 995–1005, 2020.
- [10] C. A. Huang, T. Cooijmans, A. Roberts, A. C. Courville, and D. Eck, "Counterpoint by convolution," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017.
- [11] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," *ISMIR*, 2019.
- [12] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.
- [13] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.
- [14] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2013.
- [15] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *5th International Conference on Learning Representations, ICLR*, 2017.
- [16] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *5th International Conference on Learning Representations, ICLR 2017*.
- [17] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.
- [18] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *International Conference on Learning Representations*, 2019.
- [19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.
- [20] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [21] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017.
- [22] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- [23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR*, 2015.
- [24] T. Wang and X. Wan, "T-cvae: Transformer-based conditioned variational autoencoder for story completion," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019.
- [25] O. Ivanov, M. Figurnov, and D. Vetrov, "Variational autoencoder with arbitrary conditioning," in *International Conference on Learning Representations*, 2019.
- [26] K. Wang, H. Hua, and X. Wan, "Controllable unsupervised text attribute transfer via editing entangled latent representation," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

- [27] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, “Encoding musical style with transformer autoencoders,” in *Proceedings of the 37th International Conference on Machine Learning*, 2020.
- [28] C. Li, X. Gao, Y. Li, X. Li, B. Peng, Y. Zhang, and J. Gao, “Optimus: Organizing sentences via pre-trained modeling of a latent space,” in *EMNLP*, 2020.
- [29] Y. Huang, Y. Zhang, O. Elachqar, and Y. Cheng, “INSET: sentence infilling with inter-sentential transformer,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, Eds., 2020.
- [30] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, “Unified language model pre-training for natural language understanding and generation,” in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.
- [31] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*.
- [32] C.-W. Huang, A. Touati, L. Dinh, M. Drozdal, M. Havaei, L. Charlin, and A. Courville, “Learnable explicit density for continuous latent space and variational inference,” 2017.
- [33] H. Xu, W. Chen, J. Lai, Z. Li, Y. Zhao, and D. Pei, “On the necessity and effectiveness of learning the prior of variational auto-encoder,” 2019.

OMR-ASSISTED TRANSCRIPTION: A CASE STUDY WITH EARLY PRINTS

María Alfaro-Contreras, David Rizo, Jose M. Iñesta, Jorge Calvo-Zaragoza
Department of Software and Computing Systems, University of Alicante, Spain

{malfaro, drizo, inesta, jcalvo}@dlsi.ua.es

ABSTRACT

Most of the musical heritage is only available as physical documents, given that the engraving process was carried out by handwriting or typesetting until the end of the 20th century. Their mere availability as scanned images does not enable tasks such as indexing or editing unless they are transcribed into a structured digital format. Given the cost and time required for manual transcription, Optical Music Recognition (OMR) presents itself as a promising alternative. Quite often, OMR systems show acceptable but not perfect performance, which eventually leaves them out of the transcription process. On the assumption that OMR systems might always make some errors, it is essential that the user corrects the output. This paper contributes to a better understanding of how music transcription is improved by the assistance of OMR systems that include the end-user in the recognition process. For that, we have measured the transcription time of a printed early music work under two scenarios: a manual one and a state-of-the-art OMR-assisted one, with several alternatives each. Our results demonstrate that using OMR remarkably reduces users' effort, even when its performance is far optimal, compared to the fully manual option.

1. INTRODUCTION

Music is a language used and understood all over the world, hence being one of the cornerstones of cultural heritage. In order to represent this art visually, so that it can be transmitted and later interpreted as conceived by the composer, many notation systems have developed and evolved over time. Their engraving in the so-called music score was mostly done by handwriting or typesetting processes until the end of the 20th century, thus there exist millions of music documents only available as physical documents [1].

The ongoing massive digitization of those works by means of scanners is not sufficient for these sources to become truly accessible. For that, they must be transcribed in a structured digital format such as MusicXML [2] or MEI [3], among others, which enables computational tasks

such as indexing, large-scale analysis, or editing [4]. This process is often done manually, which leads to an unfeasible challenge at a large scale. Thus, the development of systems capable of automatically performing the transcription process is of substantial importance.

Optical Music Recognition (OMR) is the field of research that studies how to computationally read music notation in scanned documents and store them in a digital structured format [5]. The success of OMR would enhance the value of all the existing musical heritage in digital libraries and facilitate the retrieval of data for musicological research. However, while OMR has been an active research area for decades [6, 7] and current state-of-the-art systems have shown promising results [5], it is not always considered as a real alternative to the manual transcription process. Moreover, there is a lack of technology transfer, i.e., instead of creating a synergistic environment, the digital humanities show a certain reticence towards automatic technologies, as if both were not committed to the same ultimate goal: to study, make accessible, and preserve the existing historical sources of music worldwide [8].

We believe that the mistrust lies in the unattainable goal to which OMR systems have been subjected: a perfectly accurate transcription. Given the vast range of different situations present in real-life recognition scenarios, a perfectly accurate OMR system is a utopia. Therefore, the automatic transcription challenge must be understood as a technology-assisted one, since human-machine interaction is necessary if there is no tolerance for errors in the transcription [9].

In this paper, we study the extent to which OMR systems provide real assistance during a transcription process. For that, we study and compare the transcription process under two scenarios: a manual-working methodology and an OMR-assisted one. We chose a corpus that allows us to compare both workflows with several available open-source tools. We aim at illustrating how OMR systems facilitate the transcription process by measuring the average procedure time per page, to estimate the workdays needed to transcribe a complete music work written in mensural notation.

The rest of the paper is organized as follows: Section 2 overviews some related proposals to this topic; Section 3 thoroughly describes the experimental setup; Section 4 reports the obtained results and their main outcomes; and finally, Section 5 concludes the present work.



2. BACKGROUND

The digital preservation of musical heritage necessarily involves the encoding in a suitable, symbolic, and computer-readable format of the musical content described in the original or scanned, as appropriate, manuscripts. In the majority of cases, this transcription process follows a manual workflow; that is, the corresponding encoding is manually written directly to the computer (either by typing or by mouse-driven actions).

Music notation contains a logical structure and more information than just a series of glyphs positioned over a staff, which makes the transcription process an inherently complex task regardless of the tool used for it. All of this entails a great deal of work that is not feasible at large-scale levels. The use of automatic technologies, in particular OMR systems, would greatly facilitate the task at issue. However, despite a large amount of existing literature on OMR research [5], hardly any system has been developed that goes beyond research tools.

Some of the most popular commercial OMR systems are PhotoScore,¹ SmartScore,² and PlayScore.³ With high recognition accuracy for printed scores, they constitute a great alternative to users who want to scan and play musical content. However, they are restricted to only one type of music notation, namely Common Western Music Notation (CWMN), and are proprietary solutions, i.e., it is necessary to pay a license or a subscription to get full access.

On the side of non-commercial systems, we find the following systems:

- Aruspix, a cross-platform software for OMR of early music prints, mainly those printed during the 16th and 17th centuries [10]. It transforms the music content of each page into an editable digital music format, allowing the correction of recognition errors by the user, used as feedback to dynamically improve its performance. For a more direct correction process, Aruspix possesses superimposition and collation features.
- Audiveris, an OMR system devised to extract the musical content from printed or handwritten score sheets in order to edit them further in music edition applications [11]. It only supports CWMN.
- Rodan, a web-based customizable OMR system [12]. The user inputs an image and creates the most appropriate workflow to process it. Once the corresponding preferred adjustments are selected, the same processing can be applied to all similar images. This is a double-edged sword because it puts the manuscript at the center of the workflow at the expense of having minimum knowledge of the technologies that can constitute an OMR system, something that is not required for librarians or musicologists.

This might pose significant risks to the effective and timely achievements of the project objectives.

- “MUSIC Recognition, Encoding, and Transcription” (MuRET), a web-based application that divides the transcription process into different steps [13]. It is a technology-centered research tool, which allows the use of different transcription approaches ranging from manual to OMR-assisted ones, producing in those cases the transcribed contents in standard encodings. MuRET allows a simultaneous graphical comparison between the original and the encoded score, favoring a quick detection of errors.

Despite the various OMR systems developed, their inclusion in the transcription process of digital libraries is far from being widely common. They are relegated to a pure research application spectrum due to the mistrust caused by their imperfect behavior, therefore making human supervision necessary.

This paper attempts to provide insights into the use of OMR systems in the transcription process of a music work to see whether, despite not being perfectly accurate, the time spent on error correction compensates for the time saved, compared to a fully manual transcription paradigm.

3. METHODOLOGY

The main focus of this work is to study to what extent an OMR approach can be useful for digital libraries in the music score transcription process. It is important to emphasize that we do not intend to benchmark OMR tools. Therefore, we design a methodology aimed at estimating the effort, in terms of user time, saved by performing a transcription process assisted by OMR technology, instead of a fully manual one.

In the following sections, we thoroughly detail our methodology. First, we describe and justify the chosen music collection; then, we illustrate the different transcription pipelines considered, as well as the tools and encoding languages involved; and finally, we explain the metrics considered that allow comparison between the previously described transcription modalities.

3.1 Corpus

We must first select a suitable music work that allows meaningful comparisons. The test case considered in this work is the *Magnificat omnitonum cum quatuor vocibus* by Cristóbal de Morales,⁴ hereafter referred to as *Magnificat* corpus. It is a collection of one hundred and twenty-six typeset pages corresponding to a Spanish choir book of the 16th century written in white mensural notation. Figure 1 shows a short example of this set.

We have chosen printed mensural notation for several reasons. First, several open-source tools with varied approaches are available, which allows us to compare different transcription paradigms. Second, the graphical com-

¹ <https://www.neuratron.com/photoscore.htm>

² <https://www.musitek.com/smartscore-pro.html>

³ <https://www.playscore.co/>

⁴ RISM A/6 M 3597. <http://bdh.bne.es/bnearch/detalle/bdh0000100234>

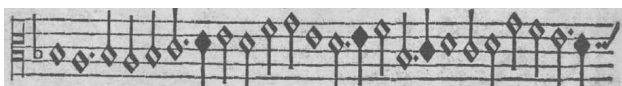


Figure 1. Staff from the *Magnificat* corpus.

plexity of the symbols is quite regular, which lets us draw conclusions using a low number of transcribed pages. Given so, we expect that the obtained conclusions can be extrapolated to other (similar) corpora. Lastly, the choice of a printed typeset work rather than a handwritten one is due to the desire of avoiding possible transcription ambiguities that may be caused by the specific writing style of the author of the work, which could affect our study.

3.2 Transcription pipelines

The transcription of a music score is understood as the process of fully transcribing the document into a structured digital format with the ultimate goal of keeping the same musical information that could be retrieved from the physical score itself [5]. In this work, we study two main transcription workflows, each one also considering different alternatives:

- (i) A manual transcription, where the musical content is directly typeset in a chosen standard format, MEI [3], PAEC [14, 15], or Humdrum [16].
- (ii) An OMR-assisted transcription, where the system performs the score transcription directly from the corresponding image, and the user corrects possible errors afterward.

Our objective is to establish a difference, quantified as the procedure time, between both paradigms. It must be noted that the transcripts will be verified at all times since it is established that, with or without the presence of OMR technology, errors might occur.

In the following sections, we will elaborate on the two transcription paradigms considered.

3.2.1 Manual transcription modality

In this modality, the manual transcription process commonly used in digital libraries is done with the computer by directly typing the encoding format or choosing graphical music symbols from a toolbar. To evaluate this paradigm, we consider the following tools:

- Verovio Humdrum Viewer (VHV)⁵, an online digital music editor and interactive notation rendering interface for Humdrum files [17]. Mensural music notation can be encoded in Humdrum using the `**mens` exclusive interpretation [18].
- Oxygen XML Editor, a paid⁶ and multi-platform editor of widespread use in the Music Encoding Initiative (MEI) community. MEI is an eXtensible

Markup Language (XML) based format that enables the transcription of a wide range of music notations. We use Verovio [19]⁷ to render the XML files edited in Oxygen XML Editor.

- The Computerized Mensural Music Editing (CMME) is a “what you see is what you get” score notation tool that allows inputting mensural notation visually [20]. For this work, it could be considered as an equivalent to MuseScore⁸, Finale⁹, or Sibelius¹⁰ but for mensural notation.
- The MuRET web application¹¹, using the graphical annotation modality for manual transcription. When considering a manual paradigm, MuRET allows two scenarios: the typesetting encoding of the music score in one of the standard formats or its manual annotation at the graphical symbol level. The former is the same process as the one performed with VHV or Oxygen XML Editor, whereas the latter is another approach to the transcription modality proposed by the CMME. The graphical annotation of MuRET consists of creating corresponding bounding boxes for each of the symbols in the score and classifying them with graphical labels, selected from a catalog. The user focuses on finding the shape and vertical position in the staff that match those of the to-be-transcribed symbol, without the need of knowing its musical meaning. Finally, the tool converts the sequence of graphical symbols into the final encoding that the user must check and correct.

3.2.2 OMR-assisted transcription modality

This modality considers an OMR system to automatically transcribe the musical content of a music score. Assuming that OMR never guarantees a perfect transcription, the objective of this paradigm is not improving accuracy but reducing the effort, measured as time, that users invest in aiding the machine to attain such perfect results. In this work, we evaluate this transcription methodology under two scenarios:

- A pre-built scenario, that is, we use an OMR system previously built for working with corpora of similar characteristics. For that, we consider Aruspix, as it has been created to recognize typeset music scores of the 16th and 17th centuries, hence being appropriate to transcribe the test case considered in this work—a typeset body of work of the 16th century.
- A from-scratch scenario, that is, we train a new OMR model for this work. To develop this approach, training pairs, consisting of problem images together with their corresponding transcript solutions, are required. This implies that we need to first manually transcribe some pages of the corpus in order

⁷ <https://www.verovio.org/index.xhtml>

⁸ <https://musescore.org>

⁹ <https://www.finalemusic.com>

¹⁰ <https://avid.com/sibelius>

¹¹ <https://muret.dlsi.ua.es/muret>

⁵ <https://verovio.humdrum.org>

⁶ For the experiment, we use a trial version of 30 days.

to train the model to be able to use it for the transcription of the remaining pages. Therefore, we will study how many pages are necessary to train a model that yields results that can be considered effective. In this scenario, we use the OMR-assisted workflow of MuRET, as it allows the retraining of its own OMR model. Thus, the training pages are transcribed with the aforementioned manual annotation from MuRET.

3.3 Evaluation procedure

To avoid biases in the results, associated both with the lack of knowledge of musical notation and the use of the tools considered, a person with a good level of computer literacy and sufficient knowledge of music to understand the test case has been chosen to carry out the proposed experimentation. The task performer is familiar with the tools and coding languages considered, and so the transcription scenarios are not challenging by themselves.

We compare the different transcription modalities by measuring the time to execute the task at hand, i.e., the time it takes to transcribe a complete score will be timed, including that needed for revision and correction of possible errors processes. This time will be referred to as *procedure time*. This will allow us to obtain the average transcription time per page for the different scenarios, which we will use to estimate the total time needed, measured in 8-hour workdays, to transcribe the entire work with each of the tools.

4. RESULTS

In this section, we present and discuss the results obtained in our experiments, in terms of the average procedure time per page, in the following order: first, those obtained for the manual paradigm; and second, those concerning the OMR-assisted one. Afterward, we compare both transcription paradigms when estimating the workdays needed to transcribe the full *Magnificat* corpus in each of them.

4.1 Manual paradigm

We first introduce the results obtained in a manual transcription process for each of the tools considered in this scenario, as described in Section 3.2.1. Table 1 shows the average procedure time per page and its standard deviation.

An inspection of the results in Table 1 reveals that the tool used in a manual transcription influences the procedure time. On the one hand, we observe that the procedure time of Oxygen XML Editor is higher than that of VHV. This is rather expected, since the encoding vocabulary size is much larger when considering an XML standard format like MEI, as in the case of Oxygen, instead of the compact Humdrum syntax, as in the case of VHV, which codifies music symbols with less than 5 characters. Furthermore, when considering graphical interfaces for the manual transcription, MuRET depicts a higher procedure time than CMME. In the latter, the graphical label corresponding to the musical symbol to be transcribed is directly dragged to

Table 1. Average time and its standard deviation for the transcription of one page of the *Magnificat* corpus for each of the tools considered in a manual transcription paradigm. Oxygen stands for Oxygen XML Editor.

<i>Manual transcription paradigm</i>				
	VHV	Oxygen	CMME	MuRET
Average time per page	27' 06"	56' 14"	33' 37"	49' 19"
Standard deviation	3' 52"	5' 49"	3' 07"	11' 27"

the desired staff position, while in MuRET a corresponding bounding box must first be created and then graphically labeled in terms of shape and vertical position in the staff, respectively, which slows down the correct labeling of the music symbol.

We consider it necessary to comment on the importance of a good design of the tool from a user experience point of view [21], and specifically the ease and speed of error correction offered by each of the tools used in the manual paradigm, as it has been also shown in other similar tasks [22]. To begin with, we find that the music sheet is rendered as it is being transcribed in VHV, which facilitates the comparison process because a division of the computer screen allows having both the original score and the transcribed version in the same viewing plane. This facilitates the detection of errors, which are easily corrected thanks to the reduced character syntax of the Humdrum encoding used in VHV. The situation changes in the case of Oxygen XML Editor because it lacks an instantaneous rendering. The result cannot be visualized until the transcription is finished and for that, an external tool, like Verovio, is required. This makes error correction a slow and tedious process.¹² On the other hand, the error correction in the CMME is similar to that of VHV, with the difference that in the former we correct graphical symbols instead of text characters. Finally, we must mention that in MuRET, the source image is present along with the transcription rendering during the whole process, facilitating the user's ability to detect and correct errors.

Note that we are not drawing conclusions from the standard deviation values because we believe that they are mainly caused by the variations in the number of symbols per page and not by the operation of the different alternatives. We nonetheless provide them for the sake of completing the results.

4.2 OMR-assisted paradigm

In order to gain insights into the OMR-assisted transcription paradigm, two scenarios are evaluated: (i) a pre-built one, where we employ an OMR system built for working with corpora of similar characteristics to the one we want to transcribe (Aruspix), and (ii) a from-scratch one, where

¹² The interactive MEI encoding tool MEISE [23] has been discarded as it failed to render mensural notation.

an OMR system is built anew by means of MuRET.

We measure the average procedure time per page in both scenarios. In the from-scratch one, as the goal is to know the number of pages that must take part in the training set to achieve a model with an acceptable recognition accuracy, we will measure the average procedure time by increasing the training set by one page each time, until reaching 10 pages. After that, the number of pages will increase by 5, as the number of pages becomes less relevant. Note that in this case, the procedure time only refers to the model recognition and the revision of the recognized manuscript times.

Table 2 and Table 3 show the average times and their standard deviations obtained in the two OMR-assisted scenarios considered. As in the previous paradigm, we will not take into account the standard deviation during the analysis because they are mainly caused by variations in the number of symbols per page.

Table 2. Average time and its standard deviation for the transcription of one page of the *Magnificat* corpus for the pre-built OMR-assisted transcription paradigm, in which Aruspix is the OMR system used. Standard deviations are not considered relevant as they are mostly linked to the variability of the number of symbols of each page and scarcely to the transcription tool.

	Average time per page	Standard deviation
<i>Pre-built</i>	8' 39''	3' 10''
<i>OMR-assisted paradigm</i>		

It should be noted that Aruspix has been considered as a static model, i.e., the dynamical improvement feature has not been used as each page has been recognized independently. We want to establish comparisons between two possible OMR-assisted transcription methodologies.

We now proceed to compare the two OMR-assisted scenarios. The pre-built scenario allows us to evaluate how OMR systems facilitate the transcription process without the need to transcribe some test pages to first train the model. Looking at Table 2, we observe that it takes less than 9 minutes to correctly transcribe a typeset musical score written in mensural notation. This indicates that we are leveraging previous efforts and existing labeled data. Moreover, the transcription process is smooth thanks to the superimposition feature that allows for straightforward comparison with the recognized results.

Oppositely, the second situation allows us to study how many test pages have to be manually transcribed to train an OMR system before it can be used in the transcription process. According to the results in Table 3, the average procedure time decreases as the number of training pages increases. This drop is very steep at the beginning, especially when considering relatively small training sets of three pages or less. A model trained with one page gives a mean transcription time of approximately 49 minutes, whereas one trained with 3 pages gives an average procedure

time of more or less 12 minutes. Moreover, training sets of 6 pages, or more, estimate lower transcription times than the pre-built OMR-assisted scenario.

4.3 Manual vs. OMR-assisted

To establish a comparison between both transcription paradigms, manual and OMR-assisted, we estimate the workdays needed for the complete transcription of the *Magnificat* corpus in both of them, by multiplying the number of pages with the average procedure time per page obtained in Section 4.1 and Section 4.2, respectively.

It is important to note that, as seen in the previous sections, in both the manual and the pre-built OMR-assisted paradigms, the average procedure times are constant as they contemplate static scenarios where both the user effort and the tool's performance can be estimated as averages. Hence, the total estimated transcription time will also be a constant value. However, in the from-scratch OMR-assisted scenario, the average transcription time of a page changes as a function of the number of training pages, as they influence the accuracy of the model used. Therefore, the time spent in the transcription of the complete corpus will vary as a function of the training pages, and to estimate it, the time spent in manually transcribing those training pages must be taken into account. We have to point out that the model's training time is not considered because it can be done in background¹³.

Figure 2 reports the workdays needed to transcribe the *Magnificat* corpus completely (126 pages) for both transcription paradigms.

By examining Figure 2, we draw several conclusions. The most important one is that the pre-built OMR-assisted transcription paradigm yields much shorter times, almost up to 5 workdays less, compared to the best case of the manual transcription paradigm. This result could be intuited when comparing the average procedure times per page obtained in the previous sections. This demonstrates that OMR systems can be embraced as a really helpful alternative, even when it does not completely automatize the process.

Additionally, the results show that by labeling only 3 pages to train an OMR model, the workdays needed for the total transcription of the corpus are less than in the manual paradigm for any of the tools considered there. This means that even if a model has to be built for a specific corpus, the OMR model-assisted transcription saves the user time, compensating for the time spent correcting errors. However, this trend is not constant; in our case, after 10 pages of training, the workdays for the from-scratch OMR-assisted scenario begin to increase. This is because the recognition accuracy of the model does not increase enough to reduce the average procedure time per page by a large amount, resulting in the time spent on the manual transcription not being compensated. In other words, a slight reduction in the OMR-assisted transcription time, of the order of a minute per page, does not make up for the manual transcription

¹³ It takes approximately 7 minutes to train a state-of-the-art OMR model with a training set of 30 pages with a low-profile GPU unit.

Table 3. Average times and their standard deviations for the transcription of one page of the *Magnificat* corpus for the from-scratch OMR-assisted transcription paradigm. This scenario uses MuRET as it allows to retrain its OMR model. Times are given as a function of the number of training pages, as they directly affect the accuracy of the model.

From-scratch OMR-assisted paradigm														
Training pages	1	2	3	4	5	6	7	8	9	10	15	20	25	30
Avg. time per page	48' 02"	32' 53"	11' 56"	9' 48"	8' 54"	6' 46"	5' 25"	5' 25"	4' 58"	4' 47"	3' 52"	3' 33"	3' 31"	3' 15"
Standard deviation	2' 49"	9' 29"	3' 40"	3' 41"	3' 18"	3' 09"	2' 51"	2' 41"	2' 45"	2' 48"	2' 44"	2' 42"	2' 49"	2' 44"

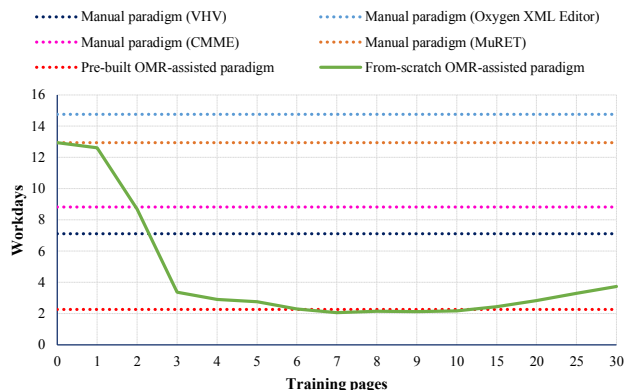


Figure 2. Workdays (assuming 8 hours per day) needed to transcribe the 126 typeset pages of the *Magnificat* corpus. In the manual and pre-built OMR-assisted paradigms, the curves are constant as they simulate static scenarios, where the user effort can be estimated as an average. However, in the from-scratch OMR-assisted case, the tendency varies depending on the number of pages used to train the OMR system, since this affects the accuracy of such a model and therefore, the subsequent error correction effort made by the user. It should be noted that in this scenario there is an inflection point after which the effort made in the manual transcription of the training pages does not compensate for the improvement of the model.

time, of the order of fifty minutes per page, needed to train such a system.

As the last point to mention, we expect that if more complex compositions are introduced, such as cross-staff notation, the difficulty of correction with the corresponding tools increase. In addition, if the OMR systems behave as expected, then we could say that the trend of the curves will maintain and the conclusions drawn can be extrapolated to other corpora. In any case, the methodology presented would still be valid.

5. CONCLUSIONS

The transcription of existing musical heritage, available only as physical documents, is a necessary activity for their preservation, access, and dissemination. Optical Music Recognition (OMR) was born with the goal of facilitating the manual transcription process by its automation. However, after decades of research and promising results,

its inclusion in the transcription workflow as a user tool has not materialized. Acceptable but not accurate results have relegated OMR systems to just representing a scientific challenge to solve.

For all the stated above, we posed the following question in this work: to what extent do OMR systems facilitate the transcription process? To answer the question, we set up an experiment that allows us to draw meaningful comparisons between two transcription methodologies: a fully manual one and an OMR-assisted one. For that, we transcribe the content of a printed early music work written in white mensural notation under both paradigms and compare the procedure time. Additionally, we evaluate the OMR-assisted paradigm from two points of view: (i) a pre-built one, where an OMR system built to recognize works of similar characteristics to those of the test case, to take advantage of previous efforts and see if the error correction compensates such savings; and (ii) a from-scratch one, where we train an OMR system from start, to see if the manual transcription of the training pages is rewarded later with the automatic transcription.

The obtained results estimate that in both cases of the OMR-assisted paradigm, the user time is less than that of the best case of the manual paradigm, indicating that posterior correction of errors in the automatic transcription is more than offset by the time saved when compared with manual transcription. The correct usability of the system to correct errors is an important issue that needs to be taken into account, as the usefulness of any transcription tool affects the process itself. Thanks to the simultaneous rendering of the transcript, the detection, and change of possible errors is a smooth process in the OMR-assisted paradigm.

Despite the first impression that OMR systems fail, they can be considered as a useful transcription tool, as they reduce the cost of the most valuable non-renewable resource, time. As future research, we aim at introducing more complex corpora, such as those involving handwriting and/or polyphonic pieces, as well as more task completers, to go beyond a single case study and provide more general insights into the analyzed question.

6. ACKNOWLEDGMENTS

First author’s research is funded by the Spanish Ministerio de Universidades through grant FPU19/04957.

7. REFERENCES

- [1] L. Pugin, “The Challenge of Data in Digital Musicology,” *Frontiers in Digital Humanities*, vol. 2, p. 4, Aug. 2015.
- [2] M. Good and G. Actor, “Using MusicXML for file interchange,” in *Proceedings Third International Conference on WEB Delivering of Music*, Leeds, UK, Sep. 2003, p. 153.
- [3] A. Hankinson, P. Roland, and I. Fujinaga, “The Music Encoding Initiative as a Document-Encoding Framework,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, Miami (Florida), USA, Oct. 2011, pp. 293–298.
- [4] G. Jones, B. Ong, I. Bruno, and K. Ng, “Optical Music Imaging: Music Document Digitisation, Recognition, Evaluation, and Restoration,” in *Interactive Multimedia Music Technologies*. IGI Global, 2008, pp. 50–79.
- [5] J. Calvo-Zaragoza, J. H. Jr., and A. Pacha, “Understanding Optical Music Recognition,” *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–35, Sep. 2020.
- [6] D. Bainbridge and T. Bell, “The Challenge of Optical Music Recognition,” *Computers and the Humanities*, vol. 35, no. 2, pp. 95–121, May 2001.
- [7] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marçal, C. Guedes, and J. Cardoso, “Optical music recognition: state-of-the-art and open issues,” *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, Oct. 2012.
- [8] M. Urberg, “Pasts and Futures of Digital Humanities in Musicology: Moving Towards a ‘Bigger Tent’,” *Music Reference Services Quarterly*, vol. 20, no. 3-4, pp. 134–150, Oct. 2017.
- [9] L. Chen, E. Stolterman, and C. Raphael, “Human-Interactive Optical Music Recognition,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York City, United States of America, Aug. 2016, pp. 647–653.
- [10] L. Pugin, “Aruspix: An Automatic Source-Comparison System,” *Computing in musicology: a directory of research*, no. 14, pp. 49–59, 2006.
- [11] H. Bitteur, “Audiveris,” 2004. [Online]. Available: <https://github.com/Audiveris/audiveris>
- [12] A. N. Hankinson, “Optical Music Recognition Infrastructure for Large-Scale Music Document Analysis,” Ph.D. dissertation, McGill University, Montréal, Québec, Canada, Dec. 2015.
- [13] D. Rizo, J. Calvo-Zaragoza, and J. M. Iñesta, “MuRET: A music recognition, encoding, and transcription tool,” in *Proceedings of the 5th International Conference on Digital Libraries for Musicology (DLfM 2018)*, Paris, France, Sep. 2018, pp. 52–56.
- [14] B. S. Brook, “The Simplified ‘Plaine and Easie Code System’ for Notating Music: A Proposal for International Adoption,” *Fontes Artis Musicae*, vol. 12, no. 2-3, pp. 156–160, Jan. 1965.
- [15] J. Howard, “Plaine and Easie Code: A Code for Music Bibliography,” in *Beyond MIDI: The handbook of musical codes*. JSTOR, 1997, pp. 362–372.
- [16] D. Huron, “Humdrum and Kern: selective feature encoding,” in *Beyond MIDI: The handbook of musical codes*. MIT Press, 1997, pp. 375–401.
- [17] C. S. Sapp, “Verovio Humdrum Viewer,” in *Music Encoding Conference*, Tours, France, May 2017.
- [18] D. Rizo, N. Pascual-León, and C. S. Sapp, “White Mensural Manual Encoding: from Humdrum to MEI,” *Cuadernos de Investigación Musical*, no. 6, pp. 373–393, 2018.
- [19] L. Pugin, R. Zitellini, and P. Roland, “Verovio: A library for Engraving MEI Music Notation into SVG,” in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, Taipei, Taiwan, Oct. 2014, pp. 107–112.
- [20] T. Dumitrescu, “Computerized Mensural Music Editing (CMME) Project,” 1999. [Online]. Available: <https://www.cmme.org>
- [21] J. Tidwell, *Designing Interfaces*. O’Reilly Media, Inc., 2005.
- [22] J. D. Valor Miró, J. A. Silvestre-Cerdà, J. Civera, C. Turró, and A. Juan, “Efficiency and usability study of innovative computer-aided transcription strategies for video lecture repositories,” *Speech Commun.*, vol. 74, no. C, p. 65–75, Nov. 2015.
- [23] N. Beer, M. Hartwig, and K. Herold, “The MEI Score Editor: Use Cases and Future Perspectives,” in *Music Encoding Conference Proceedings 2013 and 2014*, 2016, pp. 142–146.

DEEPER CONVOLUTIONAL NEURAL NETWORKS AND BROAD AUGMENTATION POLICIES IMPROVE PERFORMANCE IN MUSICAL KEY ESTIMATION

Stefan Andreas Baumann

stefan-baumann@outlook.com

ABSTRACT

In recent years, complex convolutional neural network architectures such as the Inception architecture have been shown to offer significant improvements over previous architectures in image classification. So far, little work has been done applying these architectures to music information retrieval tasks, with most models still relying on sequential neural network architectures. In this paper, we adapt the Inception architecture to the specific needs of harmonic music analysis and use it to create a model (InceptionKeyNet) for the task of key estimation. We then show that the resulting model can significantly outperform state-of-the-art single-task models when trained on the same datasets. Additionally, we evaluate a broad range of augmentation methods and find that extending augmentation policies to include a more diverse set of methods further improves accuracy. Finally, we train both the proposed and state-of-the-art single-task models on differently sized training datasets and different augmentation policies and compare the differences in generalization performance.

1. INTRODUCTION

Determining the key of a music piece is an essential step when analyzing its harmonic properties. Besides theoretical music analysis, this property is used for assessing the harmonic compatibility of music pieces [1], which is crucial when mixing multiple music pieces, as is often done by disc jockeys. As the determination of the key requires expert knowledge, systems for automatic key estimation are crucial for enabling large-scale analyses and enabling everyone, regardless of prior knowledge and skill, to harness key information. This can be especially useful for algorithmically generated mixes/playlists (e.g. Spotify’s Daily Mix¹): by optimizing the ordering of these playlists to maximize harmonic compatibility, the perceived quality of these mixes can potentially be improved. We believe that

¹<https://newsroom.spotify.com/2018-05-18/how-your-daily-mix-just-gets-you/>

for informing these decisions, regardless of whether they are made algorithmically or by a disc-jockey, improving the performance of key estimation algorithms is crucial to enable improvements in the creation of those mixes.

2. METHODS

2.1 Audio preprocessing

Before the audio data is put into the neural network model, it is preprocessed to generate a time-frequency domain representation. To obtain this representation, we use a Constant-Q transform as implemented in the librosa Python package [2] with 24 bands per octave distributed over the range from C_1 to C_8 and downsample to obtain 5 frames per second, the same framerate as used in [3].

2.2 Model

Many state-of-the-art key estimation models (e.g., [3–6]) use sequential convolutional neural networks which work on a time-frequency representation of the audio sample. Recent work has shown that more modern image classification architectures like Inception [7] and ResNet [8] are capable of outperforming at least some sequential architectures such as VGG [9] and AlexNet [10] on the task of audio classification [11]. Inspired by these findings, we present a model, which we call *InceptionKeyNet*. We base it on Inception V3 [7], as this architecture has shown the best performance on 2/3 metrics in an evaluation of audio classification models [11]. Compared to VGG-like state-of-the-art feature extractors in key estimation models, the Inception architecture is appealing as it promises much lower computational cost for similar performance [7]. Furthermore, it also promises to be especially useful in the context of localization [12], a property that could be expected to be advantageous for key estimation, as at least the determination of the key root requires precise localization capabilities. Like in the previously mentioned evaluation, we apply some modifications to the model to adapt it to the task: our primary modification is the removal of the local pooling layers from the model. This modification is necessary, as the exact “location” of features in the spectrogram on the frequency axis decides the corresponding pitch, and applying multiple steps of pooling there would prevent the model from being able to distinguish between single pitches. We also adjust the stride of all convolution layers except for the first one to be 1 for the same rea-

son. Furthermore, we also remove the auxiliary network as in [11]. Finally, we scale down the number of filters by 80% (rounding down to the next integer value) of those from the original model. We do this to reduce the capacity of the model to compensate for the significantly smaller number of output classes and training samples as compared to ImageNet [13], the dataset Inception V3 was optimized for².

2.3 Training method

For training, we split the training samples into 5 folds of equal size. We train 5 separate model instances for each round, where each instance uses a different fold for validation and trains on the samples from the remaining four, as is typical for cross-validation. To increase variety in the data, we only give a random 20s snippet of the whole audio sample to the model during training, as in [3]. We train the models with the Adam [15] optimizer with a learning rate of 0.001 and use a batch size of 32, optimizing the categorical cross-entropy between the 24 output classes – one for each combination of pitch and either major or minor mode – and the ground truth target. During training, we use a dropout rate of $p = 0.5$.

Furthermore, we use early stopping to determine the end of the training process, stopping the training when the loss does not improve over 50 epochs. For the resulting model, the weights from the epoch with minimal validation loss are used. Those instances are then combined into an ensemble, where the class probabilities are averaged to obtain a single prediction, to further improve performance.

2.4 Datasets

Similar to previous works like [3], we utilize various datasets spanning multiple genres, for both training and testing. For training, we use the following datasets:

GiantSteps MTG Key: A dataset of 1486 key annotations [16] generated from user corrections from the Beatport service in the same way as the GiantSteps Key dataset [17]. The music pieces are primarily focused on electronic dance music [3].

McGill Billboard: A dataset of 742 songs from the American Billboard charts between 1958 and 1991 [18] [19]. While the keys are not annotated in the original version, there exists a version with key annotations for a subset of 625 songs [20], which we use. Of these, we were able to obtain audio excerpts for 617 pieces.

For some trainings, we add an additional dataset, created using data mining. It consists of 3410 additional key annotations (overlaps with other datasets have been excluded) for which we have audio excerpts available, primarily focused on popular music pieces, but also including some classical pieces.

As only the GiantSteps datasets have audio available as a part of the dataset, we collect 30-second excerpts for all dataset entries, resulting in up to two such excerpts

² Multiple other approaches including modifying the stem and leaving out some mixed blocks were evaluated, too, but the best performance was achieved when just scaling down the number of filters.

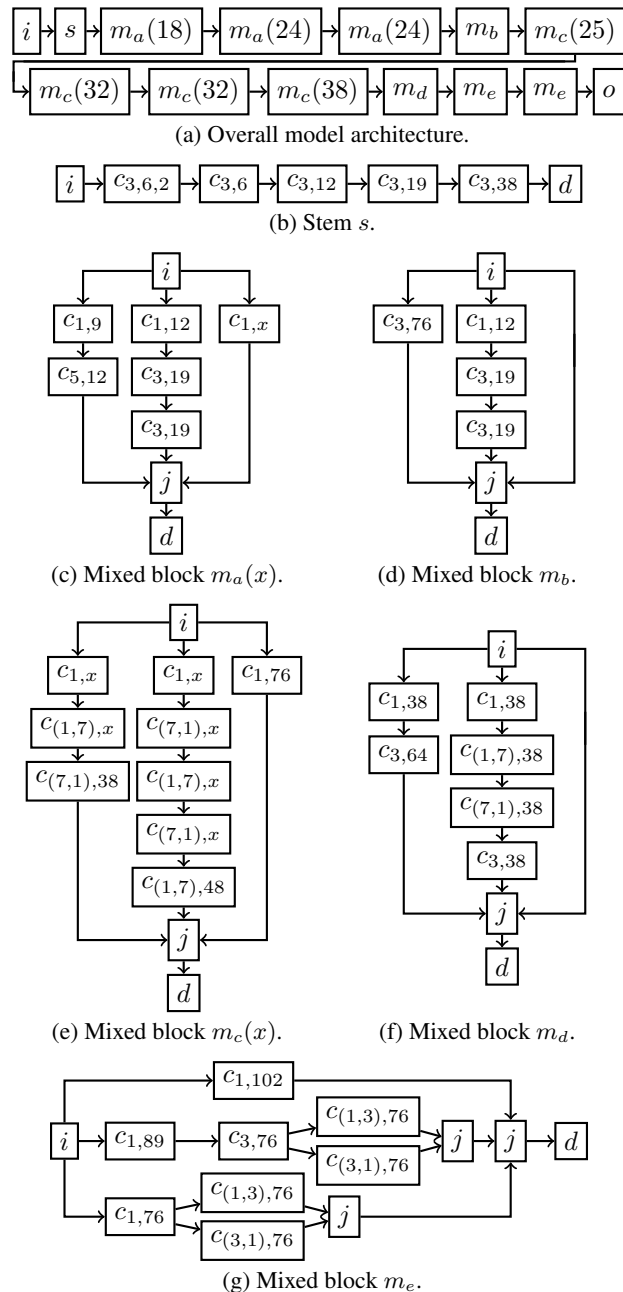


Figure 1: The architecture of the feature extractor of the InceptionKeyNet model. i denotes the input of the model or a section, $c_{k,n,s}$ (or $c_{(k_1,k_2),n,s}$) denotes a 2-dimensional convolution layer with a kernel size of $k \times k$ (or $k_1 \times k_2$), a stride of $s \times s$ and n filters, followed by batch normalization and ReLU activation. If s is not given, it defaults to 1. j denotes a layer that joins all preceding layers along their filter axis. d denotes a Spatial Dropout [14] layer with Dropout probability p . Different output structures o are presented and evaluated in 3.1.

per entry. If multiple excerpts are available for a single music piece, we randomly choose the excerpt to use for each epoch when training. When validating and testing, we use the longer 120-second excerpt from the GiantSteps datasets if available, and choose randomly otherwise, ensuring that the same excerpts are used between tests.

2.5 Metrics

For evaluating the performance of key classification models, we use the same score as used in MIREX evaluations [21], which we refer to as the ‘‘MIREX score’’. For this, we use the `mir_eval` library³ [22]. When computing the score, the predictions are divided into the following categories to consider how close the classification result is to the ground truth key:

Correct: Predictions where the root and mode are classified correctly. These predictions give a full point.

Perfect fifth: Predictions where the mode has been classified correctly, and the predicted root is either a fifth (7 semitones) higher or lower than the ground truth. These predictions give 0.5 points.

Relative major/minor: Predictions where the major or minor key relative to the ground truth key have been classified. These predictions give 0.3 points.

Parallel major/minor: Predictions where the root has been classified correctly while the mode differs. These predictions give 0.2 points.

Any predictions that do not fall under any of the previously listed categories, give no points. These point scores are then averaged to obtain the overall MIREX score.

3. EXPERIMENTS

3.1 Model Output Structures

While the original Inception architecture is already theoretically capable of processing inputs of arbitrary size when some minimum dimensions are met, we also evaluate the output structure used in the AllConv model and a modified version of it, to find out whether different output architectures affect the model performance. We evaluate the following three different output structures:

Original: The original output structure from the Inception v3 architecture, which consists of global average pooling, followed by a fully-connected layer with one neuron for each class and softmax activation [7].

AllConv-A: A 1×1 convolution layer with one filter for each class, followed by global average pooling and softmax activation as found in the AllConv model [3].

AllConv-B: Our modified variant of the AllConv-A structure, where the frequency dimension of the convolution layer is extended to cover the whole frequency axis from the previous layer.

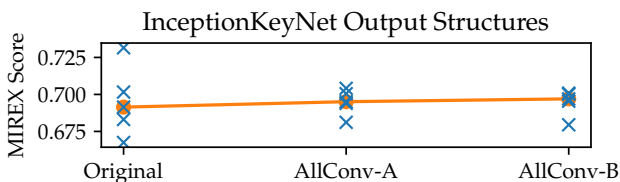


Figure 2: The validation MIREX score achieved with three different output structures. Blue crosses correspond with one model, orange dots represent the median scores.

³ The scoring of fifth errors has to be changed to also accept descending ones to match the scoring used in the latest MIREX evaluations.

The results from the evaluation are shown in figure 2. It can be seen that while all three output structures show comparable average performance, the original output structure has a significantly higher variance in validation performance than the other two variants. Overall, the performance achieved by the different folds is the most consistent with the AllConv-B variant, so we choose to proceed using it as the model output structure.

3.2 Augmentation Methods

To artificially increase the diversity of the training dataset, augmentation can be applied to the input data. While previous works on key estimation models primarily relied on pitch-shifting as the sole augmentation method (e.g. [3–6]), we evaluate a range of different augmentation methods to see whether a more diverse set of augmentation methods can help improve generalization. We evaluate the following augmentation methods (see figure 3 for visualizations of some of these methods):

Pitch-Shifting: A method that has already been used in previous works on key classification models [3–6]. It works by shifting all of the sounds in a recording by a number of semitones and adjusting the target key root accordingly. For our evaluation, we test symmetric pitch shifting ranges, which are defined via the hyperparameter $\Delta f_{max} \in \mathbb{N}_0$, which controls the discrete uniform distribution $X_{\Delta f, \text{pitch shift}} \sim \mathcal{U}\{-\Delta f_{max}, \Delta f_{max}\}$ used to randomly determine the pitch shift $\Delta f_{\text{pitch shift}}$. We precompute all of the potential pitch-shifted versions for all training samples ahead-of-time and apply it in the time-domain using SOX⁴ [24] before applying the Constant-Q transform.

Time-Warping: Based on the time-warping methodology used in the SpecAugment augmentation policy [25] for speech recognition models, we create a single-parameter version of their method: we choose six reference points along the border of the spectrogram, four at the corners and one in the center of the time axis on each side. The center points are then randomly moved along the time axis by a distance of w , with the spectrogram being warped accordingly. The augmentation hyperparameter w_{max} determines the bounds of the uniform distribution $X_w \sim \mathcal{U}(-w_{max}, w_{max})$, from which the random values for w are sampled.

Frequency and Time Masking: Two other augmentation methods used in the SpecAugment augmentation policy [25], where a part of the frequency spectrum and a part of the time axis are omitted. We use two hyperparameters, f_{max} and t_{max} , and choose the width of the range to be omitted randomly from the uniform distributions $X_f \sim \mathcal{U}(0, f_{max})$ and $X_t \sim \mathcal{U}(0, t_{max})$. We then choose two random starting points f_0 and t_0 and finally omit the frequency bands in the range $[f_0, f_0 + f]$ the time steps in $[t_0, t_0 + t]$ from the spectrogram.

⁴ Pitch-shifting with Rubberband [23] and by shifting the spectrogram were also tested, but it was found that the models trained with these alternative methods showed significantly worse performance.

Loudness Augmentation: We implement loudness augmentation by generating a factor k , with which all of the magnitudes in the spectrogram are multiplied. This factor is sampled from a uniform distribution $X_k \sim \mathcal{U}(k_{max}^{-1}, k_{max})$, whose bounds are given via the hyperparameter k_{max} .

Additive Gaussian Noise: Adding noise with a mean of zero to the input samples is a very straightforward way of augmentation: we use a Gaussian distribution $X \sim \mathcal{N}(\mu = 0, \sigma^2)$ from which a separate value is sampled randomly and then added to each data point in the spectrogram, with the noise intensity being controlled via the hyperparameter σ .

Frequency Filtering: Random frequency filtering, as previously used in [26], is a method where a random filter is generated, which amplifies or dampens a range of frequencies. The range of affected frequencies is defined via a Gaussian function, resulting in the amplitude response

$$A(f_{st}) = \max \left(1 + \frac{s}{\sigma\sqrt{2\pi}} \times e \left(-\frac{(f_{st}-f_{0,st})^2}{2\sigma^2} \right), 0 \right),$$

with which the values of the spectrogram are multiplied. We define it in semitone space for simplicity and clamp the values of the amplitude response at 0 to avoid negative amplitudes. Each filter is defined via three parameters σ , s , and f_0 , which determine the width, amplification and location respectively. These parameters are randomly sampled from three separate uniform distributions $X_s \sim \mathcal{U}(-s_{max}, s_{max})$, $X_\sigma \sim \mathcal{U}(0, \sigma_{max})$ and $X_{f_0} \sim \mathcal{U}(0, f_{max})$. s_{max} and σ_{max} are given as hyperparameters, while f_{max} is defined as the maximum frequency represented in the spectrogram.

Generally, the mentioned augmentation methods are applied at runtime with random parameters, introducing random variations into the input data. The one exception to this is pitch-shifting, where $2\Delta f_{max}$ additional variations per sample are generated ahead-of-time. These also have different key root labels, which can help compensating imbalances in the datasets' root distribution.

We evaluate these methods by performing a Bayesian optimization on all of the augmentation hyperparameters, maximizing the median validation MIREX score over a 5-fold cross-validation for each evaluated set of hyperparameters. The resulting hyperparameters define our augmentation policy. Optimizing all of the augmentation method hyperparameters together as compared to doing so independently is important, as different hyperparameter values for one augmentation method might influence another. By evaluating all methods simultaneously, we can thus get more accurate results than with independently optimized hyperparameters. To get an insight into how each hyperparameter influences model performance, we look at the fitted Gaussian process from the Bayesian optimization and compute the conditional probability distribution for that hyperparameter given that the others are set to "good" values and did not fail to train. As our condition for a hyperparameter value being "good", we check whether it is

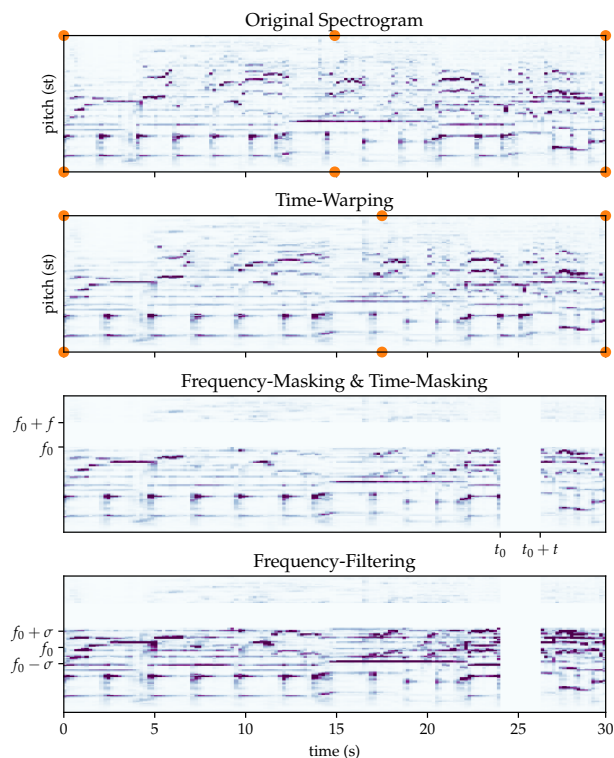


Figure 3: A visualization of some of the augmentation methods. The first row shows an entire unmodified spectrogram with the reference points for time-warping in their default position. In the second spectrogram, time-warping ($w = 2.6s$) has been applied, and the reference points have been moved accordingly. The third row shows the warped spectrogram after frequency-masking ($f = 15st$) and time-masking ($t = 2.2s$) have been applied at random positions, and the final spectrogram shows the result after applying frequency filtering ($\sigma = 10st$; $s = 90$).

at most 5% larger or smaller than the optimal value we obtained during our optimization. This way, we get evaluations of the performance of our hyperparameters that do not depend on exact values for the other hyperparameters.

A number of graphs visualizing the effect, which the hyperparameters corresponding to the different augmentation methods have on the validation performance, are shown in figure 4. From this, it is clear that loudness augmentation and additive Gaussian noise negatively affect the validation performance of our model, while pitch-shifting, time-warping, frequency-masking, and time-masking can all help improve performance on unseen samples.

For pitch-shifting, it seems that the chosen evaluation range from 0st to 12st was too small, and even more extreme values might prove beneficial to model performance. As we did not evaluate higher values in the full optimization process, we confine ourselves to the range as mentioned earlier and choose a value of 12st for the augmentation policy. For time-warping, frequency-masking, and time-masking, optimal values can be observed inside the evaluated range, although the performance improvement is significantly smaller than with pitch-shifting (especially so for frequency masking). For frequency filtering, there

seems to be a “sweet spot” near the middle of the evaluated range, where validation performance is increased slightly.

The final optimal values we obtained from the Bayesian optimization process are shown in table 1.

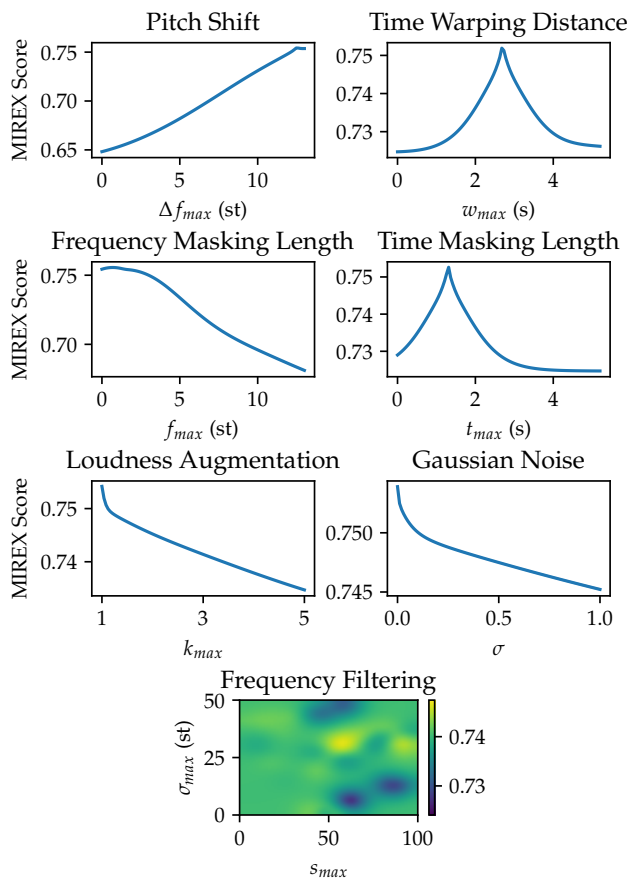


Figure 4: Estimated effect on the validation MIREX score for different augmentation methods. The blue line represents the mean validation performance for the line graphs. The frequency filtering parameters are presented as a 2-dimensional map, where the color corresponds to the mean validation performance.

Method	Parameter	Value
Pitch-Shifting	Δf_{max}	12st
Time-Warping	w_{max}	2.6s
Frequency-Masking	f_{max}	1st
Time-Masking	t_{max}	1.2s
Loudness-Augmentation	k_{max}	not applied
Additive Gaussian Noise	σ	not applied
Random Filtering	σ_{max}	36.25st
Random Filtering	s_{max}	30

Table 1: The optimal values for each augmentation hyperparameter as determined by a Bayesian optimization process, for which 178 hyperparameter combinations (resulting in 890 separate models with cross-validation) were evaluated.

4. EVALUATION

In this section, we evaluate the performance of the InceptionKeyNet model and compare it to two existing state-of-the-art single-task models. For these comparisons, we use the following datasets:

GiantSteps Key (GS): A dataset of 604 key annotations generated from user corrections from the Beatport service and a range of smaller datasets, with no overlap with the GiantSteps MTG Key dataset used for training [17].

KeyFinder (KF): A dataset of 1000 key annotations from multiple genres. [29]. We were able to obtain audio excerpts for 833 of the music pieces.

Isophonics (I): Four datasets containing songs by The Beatles, Queen, Zweieck, and Carole King. [30]. As we do not have access to the full recordings, we only use songs where a single key is annotated, resulting in 151 songs by The Beatles, 8 by Queen, 7 by Zweieck, and 2 by Carole King.

RockCorpus (RC): A dataset with annotations for 200 entries from the Rolling Stone “500 Greatest Hits of All Time” list [31]. As only the key root is given, we use a method proposed in [3] to obtain mode annotations: if at least 80% of the annotated tonic chords are of one mode, that one is selected for the overall key; otherwise, the sample is excluded. This results in 188 music pieces, of which we have audio excerpts for 186.

As we are only working with excerpts and do not have excerpts available for all entries of the various datasets, the test scores for the KeyFinder, Isophonics and RockCorpus datasets are not necessarily directly comparable to results obtained by other publications. To be able to give fair comparisons to other models on all datasets, we replicate or test them on our datasets when comparing different models. For the GiantSteps Key dataset, the scores are comparable with those in other publications, as we have the original audio files available.

We compare our model with two other single-task models: the AllConv [3] and the JXC1⁵ [5] models. To obtain fair comparisons, we replicate the models using their respective preprocessing methods as specified in the original publications [3, 5]. We then train each model on our two different training datasets, and using either only pitch-shifting or our augmentation policy. For more direct comparability, we used the same pitch-shifting range from -6st to +6st for all three models, which results in a range 1st wider than used in the original publications for the AllConv and JXC1 paper. We use the method of stopping the training described in subsection 2.3 for the AllConv model, as testing showed better performance than the default method. Finally, as we use cross-validation ensembles for our own model, we also use them for the AllConv and JXC1 models to make the comparison as fair as possible. Introducing ensembles leads to an average absolute increase in MIREX score of 3.1% and 3.6% for the AllConv and JXC1 models respectively. This is significantly

⁵ We use JXC1 instead of the JXC2 model as the latter requires training in a multi-task setting, which would go beyond the scope of this paper.

Model Architecture	Augmentation	Test MIREX Scores [%]				
		GS	KF	I	RC	Avg
Small Training Dataset (GiantSteps MTG Key and McGill Billboard)						
InceptionKeyNet (ours, ensemble, 1.7M parameters per model)	pitch-shift (−6st to +6st)	73.94	71.31	79.23	78.49	73.69
	our policy	75.50	71.94	78.93	78.28	74.46
AllConv [3] ($N_f = 20$, ensemble, 462k parameters per model)	pitch-shift (−6st to +6st)	74.27	67.78	75.83	77.58	71.74
	our policy	73.38	66.76	75.36	80.70	71.25
JXC1 [5] (ensemble, 12.5M parameters per model)	pitch-shift (−6st to +6st)	72.27	69.11	76.61	75.48	71.54
	our policy	73.66	68.03	72.80	78.49	71.46
Large Training Dataset (GiantSteps MTG Key, McGill Billboard, and our data mining dataset)						
InceptionKeyNet (ours, ensemble, 1.7M parameters per model)	pitch-shift (−6st to +6st)	74.35	70.58	80.24	83.92	74.14
	our policy	75.68	70.49	81.61	84.62	74.75
AllConv [3] ($N_f = 20$, ensemble, 462k parameters per model)	pitch-shift (−6st to +6st)	74.44	68.45	77.14	78.76	72.36
	our policy	73.06	66.15	74.88	80.70	70.81
JXC1 [5] (ensemble, 12.5M parameters per model)	pitch-shift (−6st to +6st)	74.44	69.27	81.85	81.34	73.45
	our policy	74.62	69.40	79.70	81.61	73.39
Reference Models (weights as in the original publication; trained on other datasets)						
AllConv [3, 27] (single model)	pitch-shift (−4st to +7st)	74.62	63.76	75.83	73.49	69.56
QM Key Detector v5 [28] (single model)	-	57.76 ⁶	48.12	64.40	60.48	54.15

Table 2: The results of our evaluation. The models are separated into groups depending on the datasets they have been trained on. The GS, KF, I, and RC columns show the test MIREX score on each test dataset; the rightmost column shows the average of those scores, weighted by dataset size. The best test scores in each group of models is marked in bold font.

larger than the increase of 2.0% that the InceptionKeyNet model shows, which means that any performance lead of the InceptionKeyNet model would be larger when comparing single models. The results from this evaluation are shown in table 2. As an additional reference, we also include the performance of the original trained version [27] of the AllConv model, which shows comparable average performance to our reproduction when considering the previously mentioned gain from using ensembles. Furthermore, we also include the performance of the QM Key Detector v5 [28] model on our versions of the test datasets, even though this model shows significantly worse performance across all of our test datasets.

It can be seen that when trained on the same datasets, the InceptionKeyNet model is able to outperform the AllConv and JXC1 models, with the difference between it and the next best model being 2.72% for the small and 1.30% for the large training dataset. While all models show improved performance when trained on a larger dataset, only InceptionKeyNet sees increased performance when our augmentation policy is applied. This suggests that, while broader augmentation can improve performance in at least some cases, the parameters possibly have to be tailored to each model. It can also be seen that the performance advantage given by the broad augmentation policy decreases slightly when the amount of training data increases. This likely means that the broad augmentation policy is particularly useful when little data is available.

There also seems to be no clear correlation between model parameter count and performance, which suggests that the architecture choice probably plays an important

role in deciding a models performance.

5. CONCLUSION

This paper presented an adaptation of the Inception V3 [7] architecture for harmonic music analysis tasks and used it to create a model for the key estimation task. We proceeded to evaluate a broad range of augmentation methods to find a tailored augmentation policy, and finally evaluated our model and compared it to two state-of-the-art single-task models, training all of them on the same datasets to obtain fair comparisons. These comparisons showed that the InceptionKeyNet model is capable of significantly outperforming state-of-the-art single-task models when trained on the same data with the same augmentation methods, and that further improvements are possible when applying a broad augmentation policy.

Ultimately, we believe that the main contribution of this work is showing that there is still potential to outperform state-of-the-art models in common harmonic music analysis tasks when deeper neural network architectures and matching extensive augmentation schemes are used. We hope for the presented findings to inspire further research applying similar approaches to other related MIR tasks, potentially even combining them into multi-task models. These have shown promising performance in the past, outperforming their single-task variant in at least one case [5].

The source code to train and run our model, and the subsets of public datasets where we had audio excerpts available for training, are available online⁷

⁶ We assume that the very slight deviation compared to the value in [32] is due to different versions of dependencies of sonic-annotator.

⁷ <https://github.com/stefan-baumann/inceptionkeynet>.

6. REFERENCES

- [1] I. Sha'ath, "Estimation of Key in Digital Music Recordings," Master's thesis, Birkbeck College, University of London, 2011.
- [2] B. McFee, C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and Music Signal Analysis in Python," in *Proceedings of the 14th Python in Science Conference*, vol. 8, 2015, pp. 18–25.
- [3] F. Korzeniowski and G. Widmer, "Genre-Agnostic Key Classification With Convolutional Neural Networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*. ISMIR, Sep. 2018, pp. 264–270.
- [4] —, "End-to-end musical key estimation using a convolutional neural network," in *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 966–970.
- [5] J. Jiang, G. G. Xia, and D. B. Carlton, "MIREX 2019 Submission: Crowd Annotation for Audio Key Estimation," *Music Information Retrieval Evaluation eXchange*, 2019.
- [6] H. Schreiber and M. Müller, "Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters," in *Proceedings of the 16th Sound & Music Computing Conference*, 2019, pp. 47–54.
- [7] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings*, 2015.
- [10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.
- [11] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 131–135.
- [12] "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 1–9.
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.
- [14] J. Tompson, R. Goroshin, A. Jain, Y. Lecun, and C. Bregler, "Efficient Object Localization Using Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015, pp. 648–656.
- [15] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.
- [16] Á. Faraldo, "giantsteps-mtg-key-dataset," GitHub, last accessed on 2021-01-23. [Online]. Available: <https://github.com/GiantSteps/giantsteps-mtg-key-dataset>
- [17] P. Knees, Á. Faraldo Pérez, H. Boyer, R. Vogl, S. Böck, F. Hörschläger, M. Le Goff *et al.*, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 364–70.
- [18] K. Shaffer, E. Vasiete, B. Jacquez, A. Davis, D. Escalante, C. Hicks, J. McCann, C. Noufi, and P. Salmiinen, "A cluster analysis of harmony in the McGill Billboard dataset," *Empirical Musicology Review*, vol. 14, no. 3-4, p. 146, 2020.
- [19] "The McGill Billboard Project," last accessed on 2020-10-28. [Online]. Available: [https://ddmal.music.mcgill.ca/research/The_McGill_Billboard_Project_\(Chord_Analysis_Dataset/\)](https://ddmal.music.mcgill.ca/research/The_McGill_Billboard_Project_(Chord_Analysis_Dataset))
- [20] F. Korzeniowski, "bb.zip," last accessed on 2021-01-23. [Online]. Available: <http://www.cp.jku.at/people/korzeniowski/bb.zip>
- [21] "2020 Audio Key Detection," MIREX Wiki, 2020, last accessed on 2020-10-26. [Online]. Available: https://www.music-ir.org/mirex/wiki/2020:Audio_Key_Detection
- [22] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "MIR_EVAL: A Transparent Implementation of Common MIR Metrics." in *Proceedings of the 15th International Society for Music Information Retrieval Conference*. ISMIR, 2014, pp. 367–372.
- [23] C. Cannam, "Rubberband," sourcehut, last accessed on 2021-01-31. [Online]. Available: <https://hg.sr.ht/~breakfastquay/rubberband/>

- [24] C. Bagwell, “SoX - Sound eXchange,” SourceForge, last accessed on 2021-01-31. [Online]. Available: <http://sox.sourceforge.net/>
- [25] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, “SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,” in *Proc. Interspeech 2019*, 2019, pp. 2613–2617.
- [26] J. Schlüter and T. Grill, “Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference. ISMIR*, 2015, pp. 121–126.
- [27] “madmom_models/key/2018,” GitHub, last accessed on 2021-02-17. [Online]. Available: https://github.com/CPJKU/madmom_models/tree/master/key/2018
- [28] C. Cannam, E. Benetos, M. E. P. Davies, S. Dixon, C. Landone, M. Levy, M. Mauch, K. Noland, and D. Stowell, “MIREX 2019: Vamp plugins from the Centre for Digital Music,” *Music Information Retrieval Evaluation eXchange*, 2019.
- [29] I. Sha’ath, “KeyFinder v2 Dataset,” 2014, last accessed on 2021-01-15. [Online]. Available: <http://ibrahimshaath.co.uk/keyfinder/KeyFinderV2Dataset.pdf>
- [30] “Isophonics Reference Annotations,” last accessed on 2021-02-26. [Online]. Available: <http://isophonics.net/content/reference-annotations>
- [31] T. De Clercq and D. Temperley, “A corpus analysis of rock harmony,” *Popular Music*, vol. 30, no. 1, pp. 47–70, jan 2011.
- [32] “2019 Audio Key Detection Results,” MIREX Wiki, 2019, last accessed on 2020-10-26. [Online]. Available: https://www.music-ir.org/mirex/wiki/2019:Audio_Key_Detection_Results

THE MUSIC PERFORMANCE MARKUP FORMAT AND ECOSYSTEM

Axel Berndt

Center of Music and Film Informatics
Ostwestfalen-Lippe University of Applied Sciences and Arts
Detmold University of Music
axel.berndt@th-owl.de

ABSTRACT

Music Performance Markup (MPM) is a new XML format that offers a model-based, systematic approach to describing and analysing musical performances. Its foundation is a set of mathematical models that capture the characteristics of performance features such as tempo, rubato, dynamics, articulations, and metrical accentuations. After a brief introduction to MPM, this paper will put the focus on the infrastructure of documentations, software tools and ongoing development activities around the format.

1. MOTIVATION

The performance of a musical piece transforms the musical raw material (typically a symbolic representation such as Common Western Music Notation) into a sounding output or equivalent audio signal. One and the same piece of music can be performed in many different ways. Playing the raw material exactly as notated is one special kind of performance, often referred to as “robotic” or “machine-like”. Human musicians’ performances typically involve more complex transformations that only partly derive from the notation. Those are subject to active research in musicology, in particular in the fields of performance research, Historically Informed Performance Practice, and Music Information Retrieval.

Typical questions are: How can the connection between audio document and musical score be drawn? How can the musical realization of a performer be described and put in relation to printed performance instructions and the performances of other musicians? Especially in the light of the musical culture of the past century, which has been strongly coined by audio media, this research field gains in importance and demands more and more urgently for suitable tools. Yet there is still a lack of a common, open data standard that would allow systematic and comprehensive access to the phenomena of music performance and facilitate the publication and re-use of research results in the variety of application and research contexts.

Based on several years of prior basic research and model development, the Music Performance Markup

(MPM) format was developed to address this problem. MPM enables several novel research designs. E.g., the “simulation” or “reconstruction” of a performance allows the experimental testing of auditory impressions and hypotheses about how a performance was precisely made and by which expressive means a certain effect was achieved.

However, in order for MPM to become an accepted and useful tool in the communities, we have made further efforts beyond the mere definition of the format. An ecosystem of software tools, guidelines, sample encodings and tutorials was developed of which this paper will report, thus, providing an overview and good entry point for new users.

2. DESCRIBING MUSICAL PERFORMANCES

How can a musical performance be described? The description of acoustic phenomena on a colloquial level is problematic in scientific discourse. A subjective listening impression may be described as a “strong ritardando from measure x to measure y”, but lacks the necessary precision in several respects, particularly in terms of initial and final tempo, initial and final timing position, and the specific course of the tempo transition. Another performer may play a ritardando at the same musical position that is just as strong but completely different in its execution and effect. On the other hand, a precise tempo curve may be derived from timing measurements in the audio data. This would usually render a very noisy tempo curve, because it is the sum of several timing features (tempo, rubato, asynchrony, agogic accents, and unsystematic elements often referred to as human imprecision) and measurement imprecision.

These two opposing perspectives can be found in all the formats used in music description and performance research. Accordingly, they can be roughly divided into (1) measurement series formats and (2) symbolic formats.

The most extreme form of a measurement series format is certainly the audio recording itself. Without a considerable effort of analysis, however, abstract performance concepts such as tempo, rubato, dynamics, etc. cannot be inferred from it. This is therefore a low-level representation, which by itself does not provide any information about higher-level structures in the series of measurements.

The first step in inferring volume and tempo characteristics is audio signal analysis. Note onset positions are determined algorithmically or manually, inter-onset intervals are measured, spectra are evaluated, amplitude curves are



generalized to envelopes, peak and sustain levels of individual tones are measured, etc. Examples of such analyses and the resulting complex series of measurements can be found in performance research publications [1]. A tool widely used in empirical performance research is the Sonic Visualiser [2]. The resulting measurement series are often stored in CSV (Comma-Separated Values) format [3]. An extensive data collection in this format is the MazurkaBL dataset with roughly 2000 sound recordings, annotated with loudness and tempo information [4]. The CSV format is often used for comparing analysis algorithms, such as in MIREX [5, 6].

For more complex data structures that can no longer be represented in CSV, the JSON format is a popular alternative [7]. It is easy to process, human readable and is frequently used to relate measurement series to each other, such as in audio-to-audio and audio-to-score alignment [8].

In contrast to the very detailed data of measurement series formats, which provide no immediate information on high-level structures, are the symbolic formats. They describe the abstract structures, but leave out any specific detail. Although they enable efficient communication about music, listening impressions and performance concepts, which for the most part suffices everyday use, they lack the aforementioned precision required in scholarly discourse. To take up the above example: When does a *ritardando* become a “strong” *ritardando*? The degree of abstraction becomes particularly striking when a computer generates music playback directly from the score and the result is described as “mechanical”.

Symbolic formats are used mostly to encode music notation. Most notation software and Digital Audio Workstations (DAWs) use proprietary formats that are optimized for their particular needs. Open formats are MusicXML [9], ABC [10], and Lilypond [11]. In addition, specialized formats such as Humdrum [12] and MEI [13] have become established in musicology and music editing [14].

Compared to other symbolic formats, the Standard MIDI File format [15, 16] has a special position. It represents musical information as control messages. Each note is represented by a NoteOn-NoteOff pair. The volume of notes is specified by numerical values which are interpreted by synthesizers and converted into actual amplitude values. Temporal indications are made on a tempo-independent grid similar to the musical time measure and are converted into physical time values (milliseconds) only in connection with tempo messages. Also other parameters of musical expression (tonality, mixing, DSP effects etc.) can be implemented as a series of controller messages. However, meta-structures in those domains are not represented, so that MIDI clearly corresponds to a measurement series format in this respect and is also practically used for such purposes. An example for this is CrestMusePEDB, a database of several hundred piano performances [17]. Such recordings are typically made with MIDI-fied instruments. The relatively low numerical resolution of MIDI’s 8 and 7-bit numerical values and the fact that volume and controller values are interpreted differently by each syn-

thesizer [18] are two main criticisms of the format, which is nevertheless indispensable in today’s practice.

MEI and Humdrum, too, can be extended into hybrids, as proposed by Devaney and Gauvin [19]. This is achieved by augmenting the symbolic data by measurement data, e.g. timestamps for notes. However, these measurements are not decoded and linked to corresponding larger structures leaving the gap between both perspectives open.

The essential achievement of MPM is to combine low-level and high-level perspective. Here, the high-level description serves to systematically break down the complex interplay of various performance concepts and features, which manifests itself in such detailed measurement series. This descriptive approach opens up new perspectives for a broader, scientific discourse on music performance.

3. A BRIEF INTRODUCTION TO MPM

MPM is designed as a tandem partner or complement for symbolic music formats such as MIDI, MusicXML and MEI. While these encode the score to be interpreted, MPM describes its musical performance. MPM chooses a model-based approach for this. Each description primitive is based on a mathematical model that emulates its corresponding performance feature. For a detailed introduction to these models see [20]. Currently, the corpus of models defined in MPM and supported by fundamental research comprises the following feature types:

Timing: tempo (incl. discrete and continuous tempo changes), rubato, asynchrony, random/unsystematic deviations from exact timing,

Dynamics: macro dynamics (incl. discrete and continuous dynamics changes), metrical accentuations, random/unsystematic deviations from exact dynamics,

Articulation: with absolute and relative effects on tone duration, loudness, tuning and timing (e.g. agogic accents) as well as random/unsystematic fluctuations of duration and tuning.

These models enable MPM to bridge the previously described gap between the two opposed approaches (measurement series versus symbolic representation) as they disambiguate the high-level concepts by explicit mappings. Conversely, they can also be used for resynthesis.

MPM allows to define several performances for one and the same piece of music. In the basic structure of such a performance a first peculiarity of MPM’s conception becomes apparent. Performance instructions can be defined *globally* for all parts and *locally* for a single part. If local and global information of the same domain compete, the local dominates. This makes it possible to describe polyphonic performances in which, e.g., a solo instrument has its own freedom of expression while the accompanying orchestra follows the global instructions of a conductor.

Both, the global and part environment, subdivide into *header* and *dated* information. The header contains style definitions, i.e. lookup tables to map literals (“Allegro”,

```

<?xml version="1.0" encoding="UTF-8"?>
<mpm xmlns="http://www.cemfi.de/mpm/ns/1.0">
  <performance name="a performance"
    pulsesPerQuarter="720">
    <global>
      <header>
        <tempoStyles>
          <styleDef name="famous conductor">
            <tempoDef name="Allegro"
              value="133.0"/>
            <tempoDef name="Adagio"
              value="67.4"/>
          </styleDef>
        </tempoStyles>
      </header>
      <dated>
        <tempoMap>
          <style date="0.0"
            name.ref="famous conductor"/>
          <tempo date="0.0" bpm="Allegro"
            transition.to="Adagio"
            meanTempoAt="0.7"
            beatLength="0.25"/>
          <tempo date="28800" bpm="87.45"
            beatLength="0.25"/>
        </tempoMap>
        <dynamicsMap>
          <dynamics date="0.0" volume="80.0"/>
        </dynamicsMap>
      </dated>
    </global>
    <part name="Solo Violin" number="1"
      midi.channel="0" midi.port="0">
      <header/>
      <dated>
        <dynamicsMap>
          <dynamics date="0.0" volume="92.0"/>
        </dynamicsMap>
      </dated>
    </part>
  </performance>
</mpm>

```

Listing 1: A short MPM code example.

“mf”, “staccato” etc.) to numeric values. The *dated* environment, on the other hand, is the place where performance instructions are specified and assigned to metrical positions (corr. MIDI ticks) and musical elements (e.g. articulations to notes via XML IDs). These instructions are organized in sequential lists, so-called *maps*, one for each feature type (e.g. tempoMap, dynamicsMap, rubatoMap, metricalAccentuationMap). Listing 1 demonstrates this structure. The parameterization of the style definitions and performance instructions derives from their underlying mathematical models and translate to attributes in the XML encoding.

To give an impression of one of MPM’s feature models, the code example involves a tempo slowdown from “Allegro” to “Adagio”. The course of tempo curves is modelled with power functions in the interval [0.0, 1.0]. Attribute “meanTempoAt” specifies the relative position between start and end date of the transition where the curve passes the mean tempo, in this case after 70% of the time frame (0.7). Detailed documentation of all features, syntax and models is given on the official website.¹

For music editions, MPM is a tool for the philological

¹ <https://axelberndt.github.io/MPM/>, last access: July 2021.

registration and critical examination. This also leads directly to use cases in library and archiving contexts. For the analysis of individual performances, the model-based approach of MPM provides feature classes that enable an abstracted and differentiated review. MPM does not primarily serve a purely positivistic quantification of music. Coupled with the possibilities of applying the modeled performances to symbolic music data to output them as expressive MIDI and audio, it offers, in the sense of *transformative digital intermedia studies* [21] a tool for a hermeneutic approach to performance analysis. As an experimental tool, it can serve to (re)construct performances that have not survived as audio documents but in textual form, e.g. in music-practical treatises and performance scores. Beyond a purely scientific use, this also shows potential for application in digital music production.

The model-based description approach of MPM also motivates new research questions that could not be systematically addressed so far. For instance, the sharp differentiation of timing, dynamics, and articulation into several subcategories and their description by means of mathematical models had led to questioning the common understanding of *inégalité* (the unequal playing of notes of equal value) as a pure timing feature [22, 23], which is of relevance for Historically Informed Performance Practice. Studies based on the models were eventually able to demonstrate a multifaceted interplay of micro-timing shifts, accentuation, and changes in tone duration [24]. Further research questions address, e.g., playing inaccuracies beyond classical timekeeping and synchronization studies (incl. variations in loudness, intonation, and articulation), the interaction of an ensemble depending on exterior conditions (such as acoustics and mutual visibility), and related questions of timbre research. In addition, analyses on larger corpora of performances can lead to new and more differentiated insights into music-historical change processes and formative characteristics of individual performers and schools.

4. DEVELOPING AN ESCOSYSTEM FOR MPM

The primary application areas of MPM are musicological edition and performance research as well as computer-aided music production. The development of tools for the creation, analysis, presentation and further use involve also computer science and especially MIR are further application areas. When designing the ecosystem around MPM, the goal was to reduce the barriers to adoption and productive use of MPM as much as possible.

4.1 Supporting Work in XML Editors

In the domain of digital music editing, working with XML code and dedicated editors such as Oxygen XML is common practice and requires a detailed format documentation and guidelines. This work is supported by convenience tools, namely code completion and live validation. Both require an appropriately comprehensive schema definition. The MPM schema including its documentation was speci-



Figure 1. Screenshot of the meicoApp. From the MEI data (left, green) MSM (dark blue) and MPM (light blue) are exported. The MPM here contains one performance, “MEI export performance”, from which the MSM can be rendered “to Expressive MIDI” (top, yellow). The MIDI data is rendered to audio (top right, red). At the bottom right, two external soundfonts are included, the left one is activated.

fied in the TEI ODD² meta-language, which can be compiled into several other schema languages, such as RNG, XSD and DTD. This provides a maximum compatibility with all commonly used XML parsers. In addition to the purely syntactic validation, several Schematron rules enable content-based validation. Typical errors, such as missing references, value range violations, and invalid value combinations, are detected during validation and communicated by meaningful warning and error messages.

The documentation is supplemented by several sample encodings, i.e. example projects with which users can experiment. A continuous integration and custom-developed transformer translate updates to the MPM schema immediately into the website (documentation and guidelines) and release assets (incl. an RNG and XSD compilation). The format is published under the open source licenses BSD-2 and CC-BY-4.0 and is open to future extensions from the communities.

4.2 Software Development Tools

The basis for efficient software development for a data format and its integration into existing software is the Application Programming Interface (API). The API enables convenient data access and processing (e.g. by means of predefined data structures and functionality for parsing, creating, processing and storing) without the application developers having to implement the underlying XML processing or mathematical models themselves. Thus, the API also represents a reference implementation. The MPM API was

implemented in Java as part of the converter framework *meico*³ [25]. Meico is an established conversion tool in the music encoding community and comes with some features that mesh well with MPM-related functionality.

In particular, meico offers the currently most comprehensive MEI-to-MIDI export. Therefore, it utilizes a proprietary intermediate format, Musical Sequence Markup (MSM). Its basic structure (global/part, header/dated) parallels that of MPM. In MSM all note information (without performance data) is represented. Meico’s MEI-to-MSM export was extended to convert all performance instructions to MPM data. MIDI data can also be converted to MSM. Thus, via MSM, MPM can already be used in tandem with MEI, MIDI and MSM itself.

Furthermore, a full-featured *performance rendering engine* has been integrated into the MPM API. This allows the MPM performances to be applied to MSM-encoded scores and rendered into expressive MIDI sequences. These can then be played directly in meico, exported as MIDI files and converted to audio (WAV, MP3). For MIDI playback and MIDI-to-audio rendering, third-party soundfonts (SF2, DLS) can be used or MIDI playback can be passed on to an external MIDI port. Thus, further processing (e.g. by external synthesizers) and music production in a DAW are immediately feasible.

However, meico is not only a programming library, but also provides two application programs in the form of the *meicoApp*. The command line application is integrated by some users into their XML editor as an external call (e.g. for proof-listening of music encodings) and is also used

² <https://wiki.tei-c.org/index.php/ODD>, last access: July 2021.

³ <https://github.com/cemfi/meico>, last access: July 2021.

in scripting environments (e.g. in Python programs). The graphical application is used as a stand-alone tool and offers more interaction possibilities as well as more versatile conversion paths, see figure 1. The meico library is also the core component of the following software tool.

4.3 Graphical Editor and Analysis Tool

For end users, work with MPM should not be restricted to XML editors, even if this is principally always possible. Rather, productive authoring and analysis work is to be facilitated by an efficient, graphical user interface. This motivated the development of the MPM Toolbox application. We followed a rather traditional user interface engineering process. First, paper mockups were used to try out different interface approaches and play through usage scenarios. This was complemented by experiences and feedback from a workshop with musicologists and editors during Edirom Summer School 2020. This resulted in the following observations and corresponding design decisions.

1. A schematic representation of the MPM data must be present for navigation purposes, and it must also be fully interactive. But the visual focus is on the graphical score.
2. The MSM data, i.e. the pure note information, is needed for matching time indications, voice assignments and references, so it must also be represented. However, since the focus is on work in the performance domain, the MSM does not need to be interactive, nor should it take up much display space.
3. The interaction is to be focused on the score display. This is where new performance instructions are created and performance instructions that already exist in the MPM are positioned. This is done in two different application contexts, (a) the free creation of performances (creative use case) and (b) the analysis of performance scores, i.e. the interpretation of signs in the autograph (analytical use case).
4. Not all attributes of performance instructions can and should be visualized in the score, as this quickly overloads the display space and impairs readability. Consequently, dedicated editor dialogs are needed for creating and editing them, which may employ their own visualizations to illustrate the values set.
5. The visual placement of performance instructions in the score is of limited value with respect to their assignment to musical parts and temporal placement. For instance, an instruction may clearly precede a note, but apply only from that note on.

In accordance with the first two points, the MSM and MPM data were each implemented in a tree visualization, placed fairly slim on the left and right border of the application window. These interface widgets can be both minimized and detached from the layout, allowing users to freely place them and arrange their workspace. The MPM tree is fully interactive. So it is possible to do all work directly

in it. For some types of information, such as style definitions, there is no visual representation in the musical notation practice anyway. These can only be located in the MPM tree.

The third point presented a particular challenge. While a generated score image would suffice for the creative use case (3a), the analytical use case (3b) involves working with a preexisting score image. In addition, the initial plan was only to generate the score image from MEI data using the Verovio music engraving software [26], which practically excludes some other input sources which MPM Toolbox should also support. Therefore, it was decided that the score image will be imported as image data (currently supported are JPG, GIF, PNG, BMP, and PDF). Those can be generated from all other formats with widely available tools and converters. Thus, the MPM Toolbox satisfies both use cases (3a and b) equally. A built-in score rendering solution may, nonetheless, be added later on.

For interaction in the image space, it is necessary to know the positions of notes and performance instructions and to link them to the MSM and MPM data. This is an opportunity to incorporate Optical Music Recognition (OMR) techniques. Depending on the condition of the autograph (clean print versus handwritten manuscripts with many additional markings), the recognition performance will be of different quality. Therefore, it must always be possible to perform or correct this work manually. Consequently, this manual linking was implemented with an efficient input procedure. It automatically iterates over the notes (MSM) and plays them while the user marks their position in the score image; the same for MPM. An OMR solution remains open as a future extension or third party contribution.

The linking of the note and performance information in the image space creates a point cloud. If, in the further course, performance instructions are created or linked in the image space, these can be set in relation to the surrounding elements and added to the point cloud. In doing so, voice assignment and time position are read from the surrounding linked elements and set as default values. This eliminates the need for additional user input in most cases. According to point 5, however, this can nonetheless be changed in the editor dialog.

On large and complex score page, such a point cloud can become very extensive. During interactions in the score, the points located in the local environment of the cursor must regularly be determined. To ensure smooth interaction, the point cloud must be organized in an efficient data structure. For this purpose, different standard approaches (such as BSP Trees and Quadrees) were analyzed. A peculiarity of the application context here is that interactions mostly take place in a local environment, are not scattered “chaotically” across the music sheet, but often even follow the musical sequence. The next interaction is very likely to occur near the position of the previous. An efficient data structure for this type of interaction is the Orthant Neighborhood Graph [27], which was eventually implemented in the MPM Toolbox.

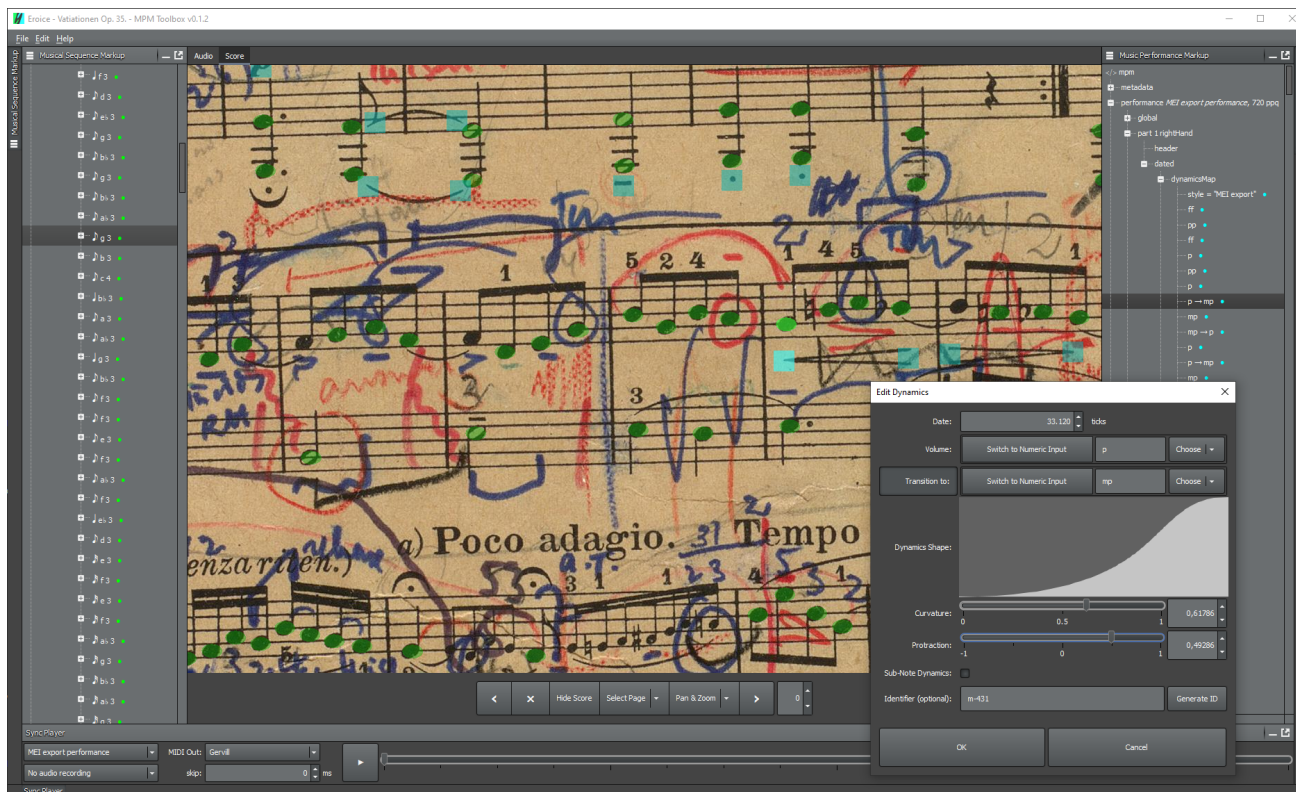


Figure 2. Screenshot of the MPM Toolbox. The green note overlays link notes to pixel positions. The blue squares link performance instructions. In the dynamics editor dialog the course of a crescendo is specified.

According to point 4, it is not practical to display all the information of the performance instructions directly in the score, since it is usually printed very compactly anyway and the remaining space is important for readability. For this reason, the annotations created with MPM Toolbox are implemented as semi-transparent and compact overlays. The detailed settings of the performance instructions are done in separate editor dialogs. Each of these dialogs not only provides the respective input options and visualizations, but also actively ensures that the input is valid, i.e. validates against the MPM schema. Numerous smaller convenience functions support the user in this process.

Of course, the created performances can also be listened to via the built-in player widget, sent to an external MIDI port (synthesizer, DAW) and exported as expressive MIDI and audio file. The player widget further allows the playback of audio recordings. Both, performance rendering and audio recording can even be played synchronously. This feature serves the purpose of listening analysis. A performance description can be iteratively developed and adjusted to approximate the listening impressions of the audio recording.

Figure 2 shows a screenshot of MPM Toolbox’s graphical user interface. Source code and executable release assets are published under the GNU GPL 3.0 license.⁴

⁴ <https://github.com/axelberndt/MPM-Toolbox>, last access: July 2021.

5. SUMMARY AND NEXT STEPS

MPM allows for the creation of highly detailed music performances. The format is accompanied by a comprehensive schema definition, documentation and sample encodings. An API, including converter and performance rendering engine, provides the basis for efficient software development around MPM. For example, the API is already in use as a generator for expressive music performances. For productive authoring and analysis work apart from XML editors, the graphical editor software MPM Toolbox was developed.

As a supplement to the MPM documentation, as well as a practical introduction to working with MPM Toolbox, a tutorial project is currently being conceived. In addition, MPM Toolbox will be supplemented by a comprehensive module for performance analyses in audio recordings. Options for interoperability with Sonic Visualizer are being investigated, e.g. import of onset detection data.

We offer introductory workshops for users and collect feedback on possible enhancements from the community regarding new performance features for MPM as well as its tools. The MPM project is open source and welcomes suggestions and contributions from the communities.

Acknowledgements: The author wishes to thank Tilo Hähnel, Benjamin W. Bohl, Simon Waloschek, and Peter Stadler for their support and contributions. This project is funded by the Fritz Thyssen Foundation.

6. REFERENCES

- [1] H. von Loesch and S. Weinzierl, Eds., *Gemessene Interpretation: Computergestützte Aufführungsanalyse im Kreuzverhör der Disziplinen*, Staatliches Institut für Musikforschung Preußischer Kulturbesitz. Mainz, Germany: Schott, 2011.
- [2] C. Cannam, C. Landone, M. B. Sandler, and J. P. Bello, “The Sonic Visualiser: A Visualisation Platform for Semantic Descriptors from Musical Signals,” in *7th Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2006.
- [3] Y. Shafranovich, “Common Format and MIME Type for Comma-Separated Values (CSV) Files,” *RFC 4180 (Informational)*, Oct. 2005.
- [4] K. Kosta, O. F. Bandtlow, and E. Chew, “MazurkaBL: Score-aligned loudness, beat, expressive markings data for 2000 chopin mazurka recordings,” in *Proc. of the 4th Int. Conf. on Technologies for Music Notation and Representation (TENOR)*, Montréal, Canada, 2018, pp. 85–94.
- [5] J. S. Downie, K. West, A. F. Ehmann, and E. Vincent, “The 2005 Music Information retrieval Evaluation Exchange (MIREX 2005): Preliminary Overview,” in *Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2005, pp. 320–323.
- [6] J. S. Downie, “The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research,” *Acoustical Science and Technology*, vol. 29, no. 4, pp. 247–255, 2008.
- [7] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” Internet Engineering Task Force (IETF), Tech. Rep., Dec. 2017, rFC 8259.
- [8] S. Waloschek and A. Hadjakos, “Driftn’ down the scale: Dynamic time warping in the presence of pitch drift and transpositions,” in *Proc. of the 19th Int. Society for Music Information Retrieval Conf. (ISMIR)*. Paris, France: Int. Society for Music Information Retrieval, Sept. 2018.
- [9] M. Good, “MusicXML specification,” <https://github.com/w3c/musicxml> [last access: May 2021], Dec. 2017.
- [10] C. Walshaw, “The ABC Music Notation,” <http://abcnotation.com/>, last access: May 2021, 2017.
- [11] H.-W. Nienhuys and J. Nieuwenhuizen, “LilyPond, A System for Automated Music Engraving,” in *Proc. of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, Firenze, Italy, May 2003, pp. 167–171.
- [12] D. Huron, “Music Information Processing using the Humdrum Toolkit: Concepts, Examples, and Lessons,” *Computer Music Journal*, vol. 26, no. 2, pp. 11–26, 2002.
- [13] A. Hankinson, R. P., and I. Fujinaga, “The Music Encoding Initiative as a Document-Encoding Framework,” in *Int. Society for Music Information Retrieval Conf. (ISMIR)*. Miami, Florida, USA: Int. Society for Music Information Retrieval, Oct. 2011, pp. 293–298.
- [14] A. Hadjakos, J. Iffland, R. Keil, A. Oberhoff, and J. Veit, “Challenges for Annotation Concepts in Music,” *Int. Journal of Humanities and Arts Computing*, vol. 11, no. 2, pp. 255–275, 2017.
- [15] R. A. Moog, “MIDI: Musical Instrument Digital Interface,” *Journal of the Audio Engineering Society (JAES)*, vol. 34, no. 5, pp. 394–404, 1986.
- [16] MIDI Manufacturers Association, “The Complete MIDI 1.0 Detailed Specification. v. 96.1,” MIDI Manufacturers Association, La Habra, CA, Tech. Rep., 1996.
- [17] M. Hashida, E. Nakamura, and H. Katayose, “Constructing PEDB 2nd Edition: A Music Performance Database with Phrase Information,” in *14th Sound and Music Computing Conf. (SMC-17)*. Espoo, Finland: Aalto University, July 2017, pp. 359–364.
- [18] R. B. Dannenberg, “The Interpretation of MIDI Velocity,” in *Proc. of the Int. Computer Music Conf. (ICMC)*. Tulane University, New Orleans, USA: International Computer Music Association, Nov. 2006, pp. 193–196.
- [19] J. Devaney and H. L. Gauvin, “Encoding music performance data in Humdrum and MEI,” *Int. Journal on Digital Libraries*, pp. 1–11, Oct. 2017.
- [20] A. Berndt, “Formalizing Expressive Music Performance Phenomena,” in *Works in Audio and Music Technology*, A. Berndt, Ed. Dresden, Germany: TUDpress, Sept. 2015, ch. 4, pp. 97–128.
- [21] O. Eide, *Media Boundaries and Conceptual Modelling: Between Texts and Maps*. New York, NY: Palgrave MacMillan, 2015.
- [22] J. J. Quantz, *Versuch einer Anweisung, die Flöte traversière zu spielen*. Bärenreiter, 1752, Reprint (1997), H. Augsbach.
- [23] S. E. Hefling, *Rhythmic Alteration in Seventeenth- and Eighteenth-Century Music. Notes Inégales and Over-dotting*. New York, NY: Schirmer Books, 1993.
- [24] A. Berndt and T. Hähnel, “Studying Music Performance and Perception via Interaction,” in *Works in Audio and Music Technology*, A. Berndt, Ed. Dresden, Germany: TUDpress, Sept. 2015, ch. 5, pp. 129–153.
- [25] A. Berndt, S. Waloschek, and A. Hadjakos, “Meico: A Converter Framework for Bridging the Gap between Digital Music Editions and its Applications,” in *Audio Mostly 2018: 13th Conf. on Interaction with Sound—Sound in Immersion and Emotion*, Glyndŵr University. Wrexham, North Wales, UK: ACM, Sept. 2018.

- [26] L. Pugin, R. Zitellini, and P. Roland, “Verovio: A Library For Engraving MEI Music Notation Into SVG,” in *Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR)*. Taipei, Taiwan: Int. Society for Music Information Retrieval, Oct. 2014.
- [27] T. Germer and T. Strothotte, “The Orthant Neighborhood Graph: A Decentralized Spatial Data Structure for Dynamic Point Sets,” *Communications in Computer and Information Science*, vol. 21, pp. 41–55, 01 2009.

IDENTIFICATION OF RHYTHM GUITAR SECTIONS IN SYMBOLIC TABLATURES

David Régnier¹

Nicolas Martin²

Louis Bigo¹

¹ Univ. Lille, CNRS, Centrale Lille
UMR 9189 CRIStAL, F-59000 Lille, France

² Arobas Music, Lille, France

`louis@algomus.fr`

ABSTRACT

Sections of guitar parts in pop/rock songs are commonly described by functional terms including for example *rhythm guitar*, *lead guitar*, *solo* or *riff*. At a low level, these terms generally involve textural properties, for example whether the guitar tends to play chords or single notes. At a higher level, they indicate the function the guitar is playing relative to other instruments of the ensemble, for example whether the guitar is accompanying in *background*, or if it is intended to play a part in the *foreground*. Automatic labelling of instrumental function has various potential applications including the creation of consistent datasets dedicated to the training of generative models that focus on a particular function. In this paper, we propose a computational method to identify rhythm guitar sections in symbolic tablatures. We define rhythm guitar as sections that aim at making the listener perceive the chord progression that characterizes the harmony part of the song. A set of 31 high level features is proposed to predict if a bar in a tablature should be labeled as rhythm guitar or not. These features are used by an LSTM classifier which yields to a F_1 score of 0.95 on a dataset of 102 guitar tablatures with manual function annotations. Manual annotations and computed feature vectors are publicly released.

1. INTRODUCTION

1.1 Guitar tablatures

As many multi-stringed instruments, the guitar allows to play a same note in multiple locations on the neck. The location where the note is played, commonly designated by the term *position*, is specified by the combination of a string name and a fret number. For example, the pitch A3 can be played at fret 2 of the G string or at fret 7 of the D string. Guitar tablatures, as illustrated in Figure 1, aim at disambiguating these positions by indicating the string/fret combinations on which notes must be played. The choice

of the positions relates to playability and to some extent to the guitarist playing style [1].

1.2 Functions in guitar tablatures

Similarly to other instruments like the piano, the role of the guitar in a pop/rock ensemble can potentially be associated with different functions over a song. Most of the time, these functions can be gathered within two categories being accompaniment and melody, generally designated by the terms *rhythm guitar* and *lead guitar*. Although not central in this paper and less frequent in the context of a pop/rock ensemble, it is worth noting that the guitar, as the piano, can simultaneously perform accompaniment and melody. The piano will typically split the two functions into left hand and right hand while the guitar will generally use a specific playing technique called *finger picking*.

A more general way to describe the function of the guitar is to estimate if it is thought to be perceived in the *background* or in the *foreground* of the song. Accompaniment parts will generally fit the first category as they often aim at supporting a main musical part like a singing part or an instrumental solo. Although melodic parts are generally thought to be perceived in the foreground, it is not uncommon for a lead guitarist to play an accompanying melody, possibly improvised, during singing sections. Examples of this behavior include the verses of the song *What's Up* (4 Non Blondes) or the bridge of the song *Cryin'* (Aerosmith).

Rhythm guitar sections in the pop/rock repertoire mostly consist in (repetitively) realizing a chord sequence. Figure 1a illustrates two bars of rhythm guitar. In contrast, *lead guitar* appears to be less well-defined as it can be alternately associated with solo parts, as in Figure 1b, *riffs* and *licks*, or hybrid accompanying parts not directly related to the underlying chord sequence. In this work, we focus on the detection of rhythm guitar sections. Rhythm guitar sections are defined as guitar sections that aim at making the listener perceive the chord progression that characterizes the harmony part of the song. In pop/rock style, such chord progressions can often be indicated independently as chord symbols accompanying melodies and lyrics.

Although rhythm guitar looks more easily definable than lead guitar, it is common to find ambiguous guitar



© David Régnier, Nicolas Martin, Louis Bigo. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** David Régnier, Nicolas Martin, Louis Bigo, "Identification of rhythm guitar sections in symbolic tablatures", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

(a) Extract of a rhythm guitar section from *Space Oddity* (David Bowie)

 (b) Extract of a solo section from *Another Brick In The Wall* (Pink Floyd)

 (c) An ambiguous extract from *Sultans of Swing* (Dire Straits)

Figure 1. Three guitar tablature extracts.

sections standing at the border of what rhythm guitar could be. Figure 1c illustrates this ambiguity with an extract of *Sultans of Swing* (Dire Straits). A rock song can typically begin with a guitar riff played as a foreground part, which is then repeated as a background accompaniment of a vocal verse. One example of this is the famous introducing riff of the song *Highway to Hell* (AC/DC) which switches from foreground to background as the vocal part begins. Ambiguities can also appear with punctual arrangement parts that are generally added during studio recording sessions.

The way ambiguous sections are labelled should be carefully considered if this labelling aims at separating a sub-corpus intended to train a rhythm guitar generative model. On one hand, a strict labelling would reach to a consistent sub-corpus with limited variety. On the other hand, a more flexible labelling would reach to a sparser sub-corpus but richer in variety. This aspect will be further discussed in Section 4.3.

1.3 Applications in MIR

Modeling instrumental function contributes to improve various applications in Music Information Retrieval including computational music analysis and generation. Identifying textural features that contribute to a function improves our knowledge in music theory and our understanding of musical style. Systematic studies bring our attention on unexpected and ambiguous cases which eventually encourage reconsiderations of common definitions.

Automatic function identification can also guide the division of large corpora into function-specific sub-corpora that will facilitate the effective training of machine learning models. For instance, a model trained exclusively on

rhythm guitar sections might be more performant in generating, analyzing, or transcribing such sections. In contrast, studying guitar playing techniques like bends, hammer-on/pull-off, and tapping will benefit from being done on a corpus of guitar solos as they appear predominantly in these sections.

2. RELATED WORK

2.1 Guitar tablature modelling

MIR research on guitar tablatures predominantly relates to automatic fingering, style analysis, and generation.

The automatic fingering task results from the fact that a same note can generally be played at multiple locations on the neck of the guitar. This task therefore consists in estimating a string/fret combination for each note of a score in order to optimize its playability. The fingering problem has been approached with various methods including HMM from audio signal [2] or symbolic scores [3], and visual detection [4].

Guitar tablature automatic analysis includes the detection in audio recordings of specific playing techniques (bends, hammer-on, pull-off, etc.) [5, 6]. Analysis of audio guitar recordings also include automatic transcription of tablatures [7] based on the training of CNNs on guitar recording spectrograms, that tackle both the pitch and fingering estimation. Automatic analysis of symbolic tablatures include guitarist style modeling with Markov models [1] or directed graphs [8], as well as the study of predominant fretboard positions [9].

Guitar tablature generation has been approached with various methods including HMMs to generate guitar arrangements from audio polyphonic music [10], integer programming to generate blues solos [11], and transformer neural networks to generate fingerpicking tablatures [12]. Of particular relevance in the context of this research, guitar tablature generation has also been limited rhythm guitar and lead guitar [13, 14] with probabilistic methods.

2.2 Musical function identification

The complementarity between *rhythm* and *lead* guitar sections in pop/rock tablatures can be generalized to the notion of musical function in musical scores. Identifying whether a section of a part corresponds to background accompaniment or to foreground melody relates to *texture modeling* [15, 16] which has been rarely addressed in symbolic scores so far. In audio recordings however, a number of works has been achieved to detect solo sections [17–19], which can employ similar techniques as vocal activity detection [20]. Solo detection contributes to the task of structure estimation for which a number of research has been done either on symbolic [21] and audio data [22].

Particularly related to this research, guitar playing modes (bass lines, chord comping and solo melody improvisations) can be detected in audio recordings with signal processing features [23] but to the best of our knowledge, there is no research detecting guitar-playing modes from symbolic tablatures.

3. HIGH LEVEL FEATURES

Rhythm guitar is considered in this work as a category of tablature sections that aim at making the listener perceive the chord progression underlying to the song. This section presents a set of 31 features that are designed and evaluated to detect such a behavior.

3.1 Bar-level labels

Although the role of a guitarist in a pop/rock song can strictly be limited to *rhythm* or *lead* guitar, it is common to see guitar tablatures switching between *rhythm* parts and *lead* parts over a same song. A number of bands have a single guitarist who alternates during a song between accompanying *rhythm* parts, and *riff/solo lead* parts. A global labelling of guitar tablatures as *rhythm* or *lead* might therefore lead to approximations and wrong interpretations. In contrast, trying to characterize the role of the guitar at the beat level would require unnecessary complexity as these functional labels tend to span over much larger time frames. In this work, we propose to assign rhythm guitar labels to *bars* of a tablature.

3.2 High level features

The 31 high level features described in this section and summarized in Table 1 are intended to be computed at each bar from raw tablature informations. These informations include note pitches, onsets, durations, string and fret indications, as well as occurrences of some technical playing modes specific to the guitar. Note that some features may derive from combinations of others. For example, the pitch of a note can be deduced from its string and fret value.

3.2.1 Note-related features

Note related features include the number of notes in the bar, as well as the presence of single notes (*i.e.*, not played simultaneously to any other). Pitch-related features include mean/min/max pitch, pitch ambitus and pitch variety (*i.e.*, number of distinct pitches). Pitch interval related features include min/max interval found between 2 successive single notes and interval variety. Finally we added the variety of note durations found in the bar.

3.2.2 Chord-related features

A chord is considered here as a set of at least two notes that are plucked simultaneously. Note that arpeggiated chords are generally notated in guitar tablatures as successive single notes labeled with a *let-ring* indication. Arpeggios are therefore not included in this definition of chords.

Chord related features include the presence of chords, the number of distinct chords and more specifically the number of n -note chords with n in [2..6]. Two additional features indicate whether a triad (either minor or major) or a fifth interval can be formed with the whole set of notes in the bar.

note features		chord features		tab features	
# notes	(7e+2)	chords*	(2e+3)	min fret	(2e+3)
single notes*	(1e+3)	# 2-chords	(1e+1)	max fret	(2e+3)
min pitch	(3e+3)	# 3-chords	(3e+2)	mean fret	(2e+3)
max pitch	(8e+2)	# 4-chords	(5e+2)	min string	(3e+3)
mean pitch	(2e+3)	# 5-chords	(2e+2)	max string	(4e0)
pitch ambitus	(1e+3)	# 6-chords	(9e+1)	mean string	(7e+2)
pitch variety	(2e+3)	chord variety	(9e+2)	<i>l-r(s)</i> *	(1e+2)
min interval	(3e+1)	m/M triad*	(5e+2)	<i>l-r (100%)</i> *	(1e+2)
max interval	(1e-1)	fifth interval*	(1e+2)	<i>w.b(s)</i> *	(6e0)
interval var	(2e+2)			<i>bend(s)</i> *	(2e+3)
duration var	(1e+2)			<i>l-h vibr(s)</i> *	(8e+2)

Table 1. Features describing tablature bars for the rhythm guitar detection task. Binary features are indicated with a *. The importance of each feature in the dataset is indicated by its ANOVA *F-value*.

3.2.3 Guitar tablature specific features

For each bar, the min/max/mean values of both frets and string are computed. Playing technique features respectively include the presence of at least one *let-ring (l-r)*, *vibrato*, *whammy bar (w.b)* and *bend* indication. A feature indicating whether the whole bar is covered by a *let-ring* indication is added.

4. EXPERIMENTS

The detection of rhythm guitar bars is formulated as a binary classification problem with two classes being *rhythm-guitar* and *other*. Each bar is described by the set of features presented above. A classifier is then trained to predict the label of a bar from its feature values.

4.1 Annotated dataset

For this work, 102 guitar tablatures in the *Guitar Pro* format from the *mySongBook* corpus¹ were analyzed, annotated and checked by two musicians experts in the pop/rock style. Selected tablatures are mostly in the pop/rock style with a few exceptions in swing/jazz. Only tablatures of six strings with standard tuning (E³ A³ D⁴ G⁴ B⁴ E⁵) were included in the annotated dataset. Among the 7487 non-empty bars (60% of the whole dataset), 6051 (82%) were labeled as *RhythmGuitar* (the other 1368 ones were complementarily labeled as *other*). Different functions were identified within this complementary class including solos, licks, riffs and studio arrangements. No finger-style tablatures were included as this playing style generally mixes both accompaniment and lead melody, making its annotation ambiguous.

Raw tablatures are not available due to legal constraints. However, computed features and manual annotations are released² in an open licence .

4.2 Feature analysis

File parsing and feature computation were performed with the *music21* python library [24] using a dedicated

¹ <https://www.mysongbook.com/>

² <http://algomus.fr/data/>

parser [25]. Figure 2 shows the value distribution of a selection of features extracted from bars of both classes in the annotated dataset. To facilitate the comparison of the two classes, the histograms indicate the proportion of feature values in each class rather than the actual number of bars. As expected, rhythm guitar and non-rhythm guitar bars appear to be respectively correlated with the presence of chords (80% of rhythm guitar bars) and the presence of single notes (92% of non-rhythm guitar bars). Non-rhythm guitar can also be distinguished by a lower number of notes and distinct chords. Rhythm guitar bars can finally be distinguished by a lower register that appears in pitch, fret, and string related features. An ANOVA Fischer test is performed for each feature as an indication of its correlation with the two classes. The results are displayed on Table 1.

4.3 Rhythm guitar prediction

4.3.1 Evaluation measure

The choice of an evaluation measure of the performance of classifier that predicts whether a guitar tab bar is rhythm guitar or not varies depending on the way the result of the classification is intended to be used.

On one hand, maximizing *precision* penalizes false positives and potentially leads to a consistent rhythm guitar sub-corpus although possibly small and uniform. Such a corpus would facilitate the training of a model that is expected to produce *typical*, but not necessary surprising, rhythm guitar tablatures. On the other hand, maximizing *recall* penalizes false negatives and potentially leads to a larger sub-corpus with more diversity although more sparse and including more debatable rhythm guitar examples. Such a corpus would be appropriate for the training of a model that aims at generating creative rhythm guitar tablatures, at the expense of outputs that possibly diverge from the common definition of rhythm guitar. Note that for a classifier that outputs a probability (like neural networks) moving the decision threshold, that is generally set by default to 0.5, could also be a way to balance between consistency and variety.

From an analysis point of view, improving our comprehension of what makes a rhythm guitar bar requires to take into account both false negatives and false positives, which could be achieved by using accuracy. As the dataset is unbalanced, we propose to evaluate the F_1 score which is defined by the harmonic mean of precision and recall.

4.3.2 Leave-one-piece-out evaluation

Training a machine learning model is often performed by splitting the dataset into a training set and a validation set. As bars can highly repeat, in particular in rhythm guitar sections, all bars belonging to the same piece should belong to the same subset to avoid overfitting. The small size of our dataset lets us adopt a *leave-one-piece-out* validation process: given the dataset of n pieces, the model is trained on $n - 1$ pieces and then evaluated on the remaining one. The process is repeated for the n pieces and the evaluation is therefore performed on the whole set of pieces of the dataset. The *leave-one-piece-out* method allows to

	<i>r.g</i> precision	<i>r.g</i> recall	F_1 score
chords/single notes presence	0.86	0.88	0.87
note + chord features	0.95	0.94	0.94
tab features	0.95	0.93	0.94
all features	0.96	0.94	0.95

Table 2. Precision, recall and F_1 score obtained for the detection of rhythm guitar (*r.g*) with a LSTM trained on different set of features.

maximize the quantity of training datas and evaluate the model on the whole dataset.

Different classifiers were tested including logistic regression, SVM, decision tree and random forest thanks to the scikit-learn framework [26]. A Long Short-Term Memory model (LSTM) implemented with the Keras framework [27] happened to provide the best results. The LSTM has 2 hidden layers of 75 and 10 units. An early stopping process was used to identify the optimal number of 12 epochs. A batch size of 32 was used and bars were presented to the model by subsequences of 5. It is not surprising to see a recurrent model outperforming standard classifiers given that bars of the same label are likely to occur consecutively in the piece as outlined in section 5. The code is publicly provided³.

5. RESULTS AND DISCUSSIONS

Different sets of features among those presented in Section 3.2 were tested to evaluate the model. We first consider a baseline model that only looks at the presence of chords and single notes in each bar. We then evaluate score based features (first two columns of Table 1). We then evaluate tablature based features only (third column of Table 1). Finally, we evaluate a model taking into account the whole set of features. In addition to F_1 score, Table 2 displays the *precision* and the *recall* on rhythm guitar label predictions.

The LSTM baseline model achieves a F_1 score of 0.87. The LSTM model combining the whole set of features reaches a F_1 score of 0.95, which outperforms other tested models including logistic regression ($F_1=0.93$), decision tree ($F_1=0.91$) and random forest ($F_1=0.93$). Although disjoint, the score feature set and the tab feature set interestingly achieve similar performance. This can partly be explained by the fact that pitch informations in score features can be derived from string+fret combinations in tab features. It is interesting to observe that string/fret and playing technics indications seem to counterbalance the absence of chord related informations, although presumably crucial for rhythm guitar detection. It also worth to note that both these two feature sets almost yield to the score obtained with the whole set of features which means that none of them much improves the other. In the following, we present wrong predictions obtained with the model trained with the whole set of features.

³ <https://gitlab.com/lbigo/rhythm-guitar-detection>

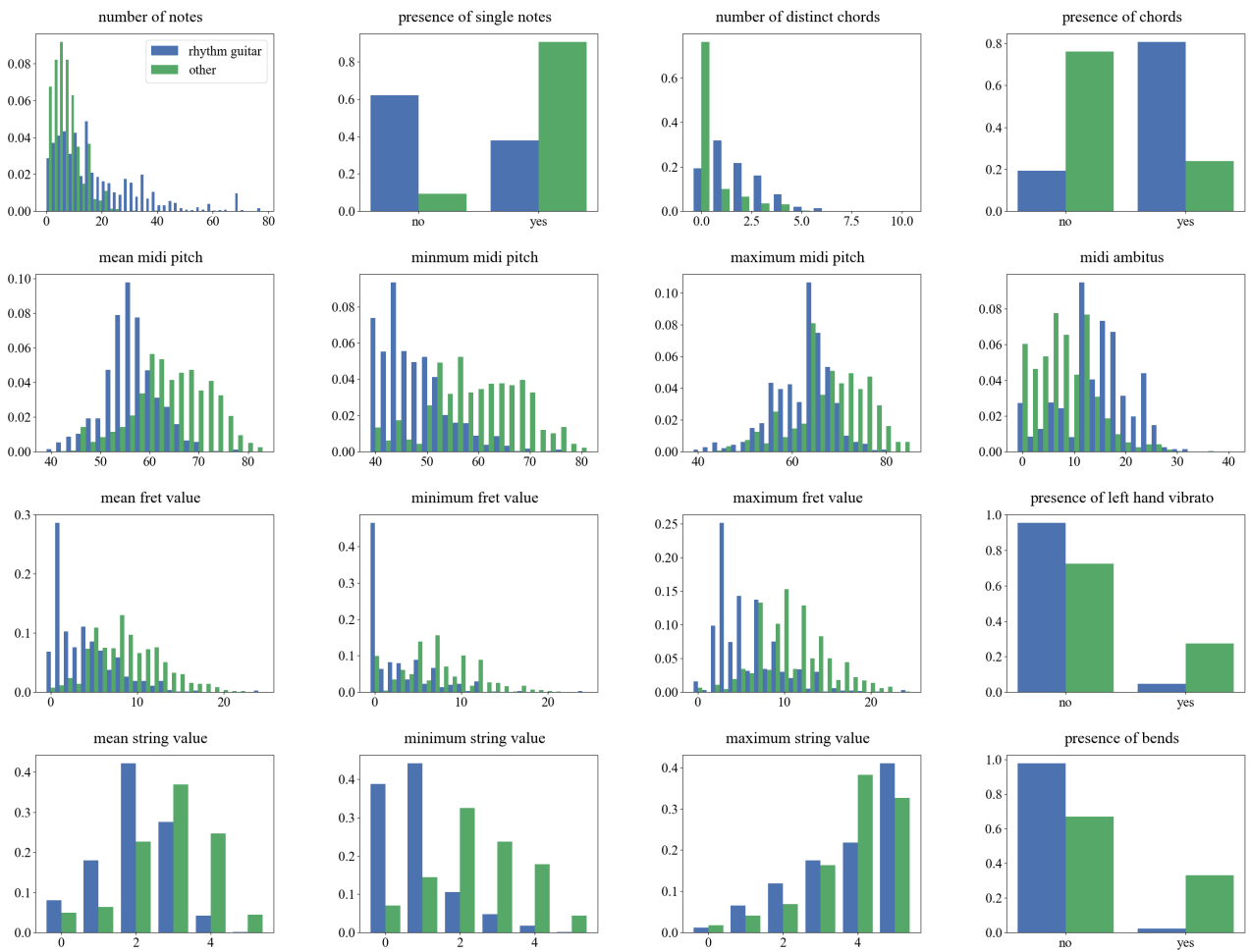


Figure 2. Disitribution of some features on bars annotated with label *Rhythm guitar* (blue) and *other* (green).

Figure 3 displays a comparison between reference annotations (top line) and predictions (bottom line) for a selection of tablatures of the corpus. Although the model succeeds in identifying large scale sections, it can still predict unlikely short sections, sometimes for one unique bar. For example, the model wrongly predicts unlikely short rhythm guitar sections in the song *Sultans of Swing* (Dire Straits). Similarly, it wrongly predicts unlikely short non-rhythm guitar sections in the songs *Stairway To Heaven* (Led Zeppelin) and *You Only Live Once* (The Strokes) as discussed below.

Figure 4 illustrates three examples of false negatives, *i.e.* rhythm guitar bars predicted as being non-rhythm guitar bars. Examples 4a and 4b are extracted from songs *You Only Live Once* (The Strokes) and *Stairway To Heaven* (Led Zeppelin). In these two examples, only the middle bar is wrongly estimated as non-rhythm guitar. Both these bars have the particularity to be the final bar of a musical phrase, leading to a new phrase beginning on the next bar. In these cases, the rhythm guitar punctually plays a short melodic lick often referred as a *fill*, which is not identified as rhythm guitar by the model. This kind of wrong predictions could probably be avoided by improving the faculty of the model to capture the tendency of adjacent bars to

have the same label and avoid the prediction of isolated labels, for example using a bidirectional LSTM. Example 4c is extracted from the song *When The Sun Goes Down* (Arctic Monkeys). In this example, the guitar starts to play bass single notes and produces a melodic line which is wrongly estimated by the model as non-rhythm guitar. This behavior could arguably be qualified as being at the edge of the common definition of rhythm guitar and it would be difficult to avoid this kind of wrong predictions without looking at the other tracks of the song (in particular the singing part), which is out of the scope of this work.

Figure 5 illustrates three examples of false positives, *i.e.* non-rhythm guitar bars predicted as rhythm guitar bars. Example 5a illustrates an extract of a solo part of the song *Hotel California* (Eagles) where the guitar repetitively plays arpeggios of the underlying chord sequence. Although the played notes belong to a rather high register, the model is probably misled by the repetitiveness, low variety and the presence of perfect triad as these features are often correlated with rhythm guitar sections. Example 5b consists in a short interlude between a solo section and the bridge of the song *La Grange* (ZZ Top). In this case, the function of the guitar seems to consist in doing a transition between two sections and could hardly be unam-



Figure 3. Comparison of manual annotations (top lines) and predictions (bottom lines) of a some tablatures of the dataset. Sections labelled as rhythm guitar are displayed in blue. Other sections are displayed in green. Empty bars are left in gray.

	<i>r.g</i> measures	<i>r.g</i> sections	mean <i>r.g</i> section length	isolated <i>r.g</i> measures
reference	6051	101	77	0
prediction	5923	223	34	44

Table 3. Comparison of consecutiveness of annotated and predicted rhythm guitar (*r.g*) bars.

biguously described as rhythm guitar or not. Example 5c is extracted from a solo section of the song *Minor Swing* (Django Reinhardt). The model is clearly misled by the sudden occurrence of chords here. As it is often the case gypsy jazz, the guitar punctually includes series of chords within a solo, that do not necessarily precisely feet the underlying chord sequence. This behavior typically lasts a few bars before the guitar goes back to melody.

Table 3 illustrates the difficulty of the model to reconstruct continuous rhythm guitar sections. Although the proportion of rhythm guitar bars predicted by the model is close to the one of the reference, these bars are grouped in smaller and more numerous sections. The model particularly tends to detect isolated rhythm guitar bars although the reference annotation do not include any of them.



Figure 4. Examples of false negatives. The second bar is wrongly predicted as non-rhythm guitar on each example.



Figure 5. Examples of false positives. Second and third bars are wrongly predicted as rhythm guitar.

6. CONCLUSIONS

This study improved our understanding of which features contribute to a rhythm guitar section. We believe that this approach can be used to separate a corpus of pop/rock guitar tablatures into consistent sub-corpora dedicated to tablature generation limited to a specific function.

The method presented here could benefit from several improvements. A finer tuning of the LSTM, or the use of a bidirectional LSTM, would probably better capture the tendency of adjacent bars to have the same label and therefore to limit isolated predictions which appear to be very unlikely across the corpus. Futur works also include adding features that look at more structural aspects of the song like bar location and activity of other tacks, in particular singing tracks as rhythm guitar if often intended to accompany singing.

Acknowledgements. The authors are grateful to anonymous reviewers and all the Algomus team for fruitful comments on the manuscript. This work is partially funded by French CPER MAuVE (ERDF, Région Hauts-de-France).

7. REFERENCES

- [1] O. Das, B. Kaneshiro, and T. Collins, “Analyzing and classifying guitarists from rock guitar solo tablature,” in *Proceedings of the Sound and Music Computing Conference, Limassol, Chypre*, 2018.
- [2] A. M. Barbancho, A. Klapuri, L. J. Tardón, and I. Barbancho, “Automatic transcription of guitar chords and fingering from audio,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 915–921, 2011.
- [3] G. Hori and S. Sagayama, “Minimax viterbi algorithm for hmm-based guitar fingering decision.” in *ISMIR*, 2016, pp. 448–453.
- [4] A.-M. Burns and M. M. Wanderley, “Visual methods for the retrieval of guitarist fingering,” in *Proceedings of the 2006 conference on New Interfaces for Musical Expression*. Citeseer, 2006, pp. 196–199.
- [5] L. Reboursière, O. Lähdeoja, T. Drugman, S. Dupont, C. Picard-Limpens, and N. Riche, “Left and right-hand guitar playing techniques detection.” in *NIME*, 2012.
- [6] Y.-P. Chen, L. Su, Y.-H. Yang *et al.*, “Electric guitar playing technique detection in real-world recording based on f0 sequence pattern recognition.” in *ISMIR*, 2015, pp. 708–714.
- [7] A. Wiggins and Y. Kim, “Guitar tablature estimation with a convolutional neural network.” in *ISMIR*, 2019, pp. 284–291.
- [8] S. Ferretti, “Guitar solos as networks,” in *2016 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2016, pp. 1–6.
- [9] J. Cournut, M. Giraud, L. Bigo, N. Martin, and D. Régnier, “What are the most used guitar positions?” in *International Conference on Digital Libraries for Musicology (DLfM 2021)*, Online, United Kingdom, 2021. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-03279863>
- [10] S. Ariga, S. Fukayama, and M. Goto, “Song2guitar: A difficulty-aware arrangement system for generating guitar solo covers from polyphonic audio of popular music.” in *ISMIR*, 2017, pp. 568–574.
- [11] N. d. S. Cunha, A. Subramanian, and D. Herremans, “Generating guitar solos by integer programming,” *Journal of the Operational Research Society*, vol. 69, no. 6, pp. 971–985, 2018.
- [12] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, “Automatic composition of guitar tabs by transformers and groove modeling,” *arXiv preprint arXiv:2008.01431*, 2020.
- [13] M. McVicar, S. Fukayama, and M. Goto, “Autorhythmguitar: Computer-aided composition for rhythm guitar in the tab space,” in *ICMC*, 2014.
- [14] —, “Autoleadguitar: Automatic generation of guitar solo phrases in the tablature space,” in *2014 12th International Conference on Signal Processing (ICSP)*. IEEE, 2014, pp. 599–604.
- [15] B. Duane, “Texture in eighteenth-and early nineteenth-century string-quartet expositions,” Ph.D. dissertation, Northwestern University, 2012.
- [16] M. Giraud, F. Levé, F. Mercier, M. Rigaudière, and D. Thorez, “Towards modeling texture in symbolic data,” in *ISMIR*, 2014, pp. 59–64.
- [17] K. A. Pati and A. Lerch, “A dataset and method for guitar solo detection in rock music,” in *Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.
- [18] G. Peterschmitt, E. Gomez, and P. Herrera, “Pitch-based solo location,” in *Proc. of MOSART Workshop on Current Research Directions in Computer Music*, 2001.
- [19] F. Fuhrmann, P. Herrera, and X. Serra, “Detecting solo phrases in music using spectral and pitch-related descriptors,” *Journal of New Music Research*, vol. 38, no. 4, pp. 343–356, 2009.
- [20] M. Mauch, H. Fujihara, K. Yoshii, and M. Goto, “Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music.” in *ISMIR*, 2011, pp. 233–238.
- [21] P. Allegraud, L. Bigo, L. Feisthauer, M. Giraud, R. Groult, E. Leguy, and F. Levé, “Learning sonata form structure on mozart’s string quartets,” *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 2, no. 1, pp. 82–96, 2019.
- [22] K. Ullrich, J. Schlüter, and T. Grill, “Boundary detection in music structure analysis using convolutional neural networks.” in *ISMIR*, 2014, pp. 417–422.
- [23] R. Foulon, P. Roy, and F. Pachet, “Automatic classification of guitar playing modes,” in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2013, pp. 58–71.
- [24] M. S. Cuthbert and C. Ariza, “music21: A toolkit for computer-aided musicology and symbolic music data,” 2010.

- [25] J. Cournut, L. Bigo, M. Giraud, and N. Martin, “Encodages de tablatures pour l’analyse de musique pour guitare,” in *Journées d’Informatique Musicale (JIM 2020)*, Strasbourg, France, 2020. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02934382>
- [26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [27] F. Chollet *et al.* (2015) Keras. [Online]. Available: <https://github.com/fchollet/keras>

ON-LINE AUDIO-TO-LYRICS ALIGNMENT BASED ON A REFERENCE PERFORMANCE

Charles Brazier¹ Gerhard Widmer^{1,2}

¹Institute of Computational Perception, Johannes Kepler University Linz, Austria

²LIT AI Lab, Linz Institute of Technology, Austria

firstname.lastname@jku.at

ABSTRACT

Audio-to-lyrics alignment has become an increasingly active research task in MIR, supported by the emergence of several open-source datasets of audio recordings with word-level lyrics annotations. However, there are still a number of open problems, such as a lack of robustness in the face of severe duration mismatches between audio and lyrics representation; a certain degree of language-specificity caused by acoustic differences across languages; and the fact that most successful methods in the field are not suited to work in real-time. Real-time lyrics alignment (tracking) would have many useful applications, such as fully automated subtitle display in live concerts and opera. In this work, we describe the first real-time-capable audio-to-lyrics alignment pipeline that is able to robustly track the lyrics of different languages, without additional language information. The proposed model predicts, for each audio frame, a probability vector over (European) phoneme classes, using a very small temporal context, and aligns this vector with a phoneme posterigram matrix computed beforehand from another recording of the same work, which serves as a reference and a proxy to the written-out lyrics. We evaluate our system’s tracking accuracy on the challenging genre of classical opera. Finally, robustness to out-of-training languages is demonstrated in an experiment on Jingju (Beijing opera).

1. INTRODUCTION

Audio-to-lyrics alignment aims at synchronizing an audio recording with its corresponding lyrics, in order to retrieve the position of spoken or sung textual units in the recording. The task has been widely researched in the context of speech data [1, 2], and recently there has also been very promising work on polyphonic music [3–6], even on multi-lingual alignment in a single framework [7]. Robust alignment methods would be useful for applications such as automatic karaoke captioning [8], music or video cutting based on the lyrics, or automatic subtitling in music videos.

Other tasks in Music Information Retrieval (MIR), such as cover detection or score following, could also benefit.

All proposed audio-to-lyrics alignment methods are composed of an acoustic model, classifying each audio frame into a set of textual units, and an alignment procedure to obtain the desired lyrics timings. Previous works in the field [6, 7] use source separation systems as a pre-processing step to extract the singing vocals beforehand, even if in [5], the authors mention that the vocal extraction algorithms can add artifacts in the vocals. In [5], the authors improved their aligners by modeling vowel durations in their lexicons [9], which permits taking into account certain pronunciation aspects. Also, in [6], the alignment is done in several passes, to first spot keyword positions in the audio, and then consider several smaller alignments in between the keywords.

The challenging question of real-time audio-to-lyrics alignment remains open and has not yet been tackled in the literature. This type of application would have great value, especially in live concerts and operas where fully automated subtitle displays could directly help the audience in following the live story. However, due to the architecture of existing acoustic models, the types of alignment algorithms conventionally used, and the additional steps detailed above, previous methods are not suited to work in real-time.

In this work, we propose a first audio-to-lyrics alignment pipeline that can operate in real-time,¹ in a language-independent way. Instead of using a pronunciation dictionary to translate the lyrics into phonemes, we propose to use another recording of the target piece as a reference and proxy to the lyrics.² This method has been widely used in the domain of score following, for robust tracking during live orchestra [10] or opera [11] performances. To this end, we first design an acoustic model that predicts a frame-wise probability distribution over a pre-defined set of phonemes. Each prediction is based on a very limited temporal window, using a future context of 280 ms which defines the delay of our system. Then, saving all the predictions in a posterigram matrix, we perform an OnLine Time Warping (OLTW) alignment between this (inremen-



© C. Brazier, and G. Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** C. Brazier, and G. Widmer, “On-Line Audio-to-Lyrics Alignment Based on a Reference Performance”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ We will not actually measure runtimes in this paper; the important aspect of our method is that it solves the problem in an on-line fashion, without access to future information, and that we can quantify its theoretical latency, based on how it processes the input data.

² Of course, this will only be practicable in certain domains, where reference recordings are available.

tally computed) posterioigram and the one from another performance that has been generated beforehand.

In this paper, after presenting acoustic models and alignment strategies of existing works in Section 2, we will present our on-line alignment system in Section 3. The robustness and accuracy of our tracker will be evaluated on two distinct datasets of ‘art’ music (opera) to be described in Section 4. Results and a discussion will be given in Section 5. Finally, our conclusions and open questions will be presented in Section 6.

2. RELATED WORK

Audio-to-lyrics alignment is an active research topic that is constantly stimulated by a yearly musical challenge³ and the appearance of new open-source training datasets such as DALI [12] or DAMP [13]. Recent works follow a common pipeline. First, an acoustic model is trained to extract from the audio signal a ‘posterioigram’ that represents the frame-wise probability distribution over textual units through time. On the lyrics side, the text is first translated into a sequence of textual units that correspond to the classes of the trained acoustic model. Finally, an alignment algorithm is applied between the posterioigram and the lyrics’ representation, to retrieve textual unit timings in the audio. In this section, we present existing acoustic models, detail the alignment process, and show their limitations to operate in real-time.

2.1 Acoustic Model

Acoustic models are Deep Neural Networks whose task is to classify audio inputs into a sequence of probabilities over a set of predefined textual units representing the lyrics present in the audio. They are trained with datasets that include a multi-modal mapping between lyrics and audio. Due to the difficulty (or near impossibility) of obtaining ground truth frame-level annotations, acoustic models are generally trained with weak annotations, at the sentence or word level, where the precise alignment between audio frames and lyrics remains unknown. Inspired by Speech Recognition [14], models can be trained in different ways. A first strategy, used in [5, 6], fits a Gaussian Mixture Model Hidden Markov Model (GMM-HMM) that force-aligns the lyrics with the audio to generate phone labels at the frame level. Then, the acoustic model is trained at the sequence level with the Lattice-Free-Maximum Mutual Information (LF-MMI) loss function [15], considering the output of the GMM-HMM as ground truth. [6] combine this objective with the Cross-Entropy (CE) loss to train the model at the frame level. Another strategy, used in [4, 7], aims at directly aligning the audio with the lyrics, using the *Connectionist Temporal Classification (CTC) loss* [16].

Acoustic models classify each audio frame into a set of textual units, which are intermediate lyrics representations. In [4], the lyrics are represented as a sequence of characters, whereas [5–7] use a phoneme representation. In the

case of a single multilingual acoustic model, using a character representation is delicate. Even within one language, a letter can be pronounced differently depending on the context, which can confuse the acoustic model that tries to classify audio frames into letters. The phoneme representation is more consistent across different languages and provides better performance since it is not language-specific. In [7], the authors report better results in using phonemes as the intermediate representation.

Acoustic models can employ different network architectures. Existing architectures have been designed to take advantage of future information to improve the prediction at each time step, which limits their use to offline applications. In [4], the authors build a Wave-U-Net that takes as input windows of raw audio and encodes the information at different scales. [5, 6] employ a combination of Convolutional Neural Networks and Time Delay Neural Network [17] (TDNN-F) layers to model long temporal context. [6] add CNN layers at the beginning of their model to speed up the training, and a multi-head attention layer at the end to help focusing on different parts of the input for each prediction. Each layer is also responsible to extend the scope of input frames that have a direct influence on the output frame prediction, that is, its Receptive Field (RF). From the descriptions of these model architectures, we derive RF values higher than $1.5s$ ⁴, which is not suitable for a real-time application. Finally, the authors in [7] use a Bidirectional Long Short-Term Memory (BLSTM) to model the temporal dependencies, which combines backward and forward information about the sequence for every prediction. Another downside of this architecture is that it is much slower to train [18].

2.2 Alignment

In the next step, an alignment algorithm is applied between the posterioigram generated by the acoustic model and the lyrics. To compare the two modalities, the lyrics must first be translated into a sequence of textual units matching with the classes of the trained acoustic model. This is generally done by using open-source pronunciation dictionaries such as CMUdict⁵, for English only, or Phonemizer⁶, which covers several languages. Then, considering the posterioigram as our observation sequence and the target lyrics, Viterbi-based forced alignment is applied to find the most probable path in the posterioigram that generates the lyrics. In [19], the author compares two trackers, one based on Dynamic Time Warping (DTW) between posterioigrams and binary posterioigrams generated from the lyrics; and one based on the Viterbi algorithm between the decoded lyrics from the posterioigrams and their ground truth. The author reports that the first approach, analogous

⁴This rough calculation is only based on the respective stack of TDNN layers and is much higher in practice. For a full description of their architectures, we refer the reader to the scripts available at respectively <https://github.com/chitralkha18/AutoLyrixAlign> and <https://github.com/emirdemirel/ALTA>.

⁵<https://github.com/cmuspinx/cmudict>

⁶<https://github.com/bootphon/phonemizer>

³https://www.music-ir.org/mirex/wiki/2020:Automatic_Lyrics-to-Audio_Alignment_Results

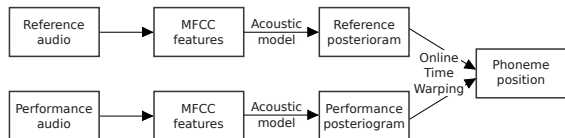


Figure 1. Integrated Model.

to the audio-to-midi alignment task, does not perform as well as the Viterbi-based method.

The alignment used in all the above works suffers from two main limitations. First, the alignment algorithm works on the full audio recording and the entire lyrics, which does not permit real-time or on-line application. Second, the phoneme-based approaches that yield the best alignment accuracies [7] are dependent on text-to-phoneme tools to translate the lyrics into a phoneme sequence, according to the corresponding language. This limits the scope to languages that are covered by these tools.

3. PROPOSED SYSTEM

Our proposed on-line audio-to-lyrics alignment system is illustrated in Figure 1. It is composed of an acoustic model that classifies in real-time each audio frame into a set of predefined phonemes. Then, instead of using the lyrics sequence for the alignment, we use a posterioqram matrix that is computed beforehand from another recording of the same work, which serves as a reference and a proxy to the written-out lyrics. Finally, we apply an OLTW alignment algorithm to align the two posterioqrams, which permits to retrieve the position of the lyrics in the live recording with the help of manual lyrics annotations affixed to the reference.

3.1 Acoustic Model

For our acoustic model, we select the CP-ResNet [20] architecture that has already proven to perform well in Acoustic Scene Classification [21] and in Emotion and Theme Recognition in Music [22]. Based on the ResNet architecture, the model stacks convolutional layers with additional residual connections between layers. The CP-ResNet is designed in such a way that the maximum time *receptive field* (RF) is controlled by a hyper-parameter ρ_t that defines the architecture of the model. For our experiments, we fix $\rho_t = 6$, and our network architecture is given in Table 1. The RF of the model can be recursively calculated with stride and kernel size of each layer (see equation (1) in [20]). The corresponding RF is equal to 57 frames. This means that each output vector is dependent on 57 input frames centered around its time position, defining the latency of our model to 28 frames. The architecture of the deep network is specified in Table 1.

The acoustic model takes as input 80 Mel-Frequency Cepstral Coefficients (MFCC) features extracted from the audio signal, with a sampling rate of 16 kHz, and computed with a window size of 20 ms and a hop size of 10 ms. It corresponds to a model latency equal to 280 ms, which we

Layer	Filters	Kernel	Stride	Pad
Conv2d+BN+ReLU	64	5×5	2,2	1,1
Conv2d+BN+ReLU	64	3×3	1,1	1,1
Conv2d+BN+ReLU	64	1×1	1,1	1,1
MaxPool2d	1	2×2	2,2	0
Conv2d+BN+ReLU (x6)	64	3×3	1,1	1,1
Conv2d+BN+ReLU (x2)	128	3×1	1,1	1,0
Conv2d+BN+ReLU (x2)	128	1×1	1,1	0,0
Conv2d+BN+ReLU	60	1×1	1,1	0,0
LogSoftmax	-	-	-	-

Table 1. CP-Resnet with $\rho_t = 6$.

consider acceptable for real-time applications such as, e.g., opera subtitling.

The model outputs a vector every 40 ms. The vector is of length 60 and includes 57 phonemes representing the union of all phonemes present in the English, German, French, Spanish and Italian languages, the space token, the instrumental token, and the mandatory blank token for CTC training. The five languages correspond to the most dominant languages present in the DALI dataset [12], with a bias towards English: the dataset includes 225 hours of English songs, 20 hours of German, 10 hours of French, 10 hours of Spanish, and 10 hours of Italian, for a total of 275 hours with hierarchical annotations at the sentence, word, or note level. The dataset only covers Western musical genres. The choice of phoneme representation has been motivated by the multilingual aspect of our work, permitting to train a single model on different languages. The instrumental token proves to be useful to label audio inputs that do not contain singing voice, especially during silence or instrumental passages.

The model is trained on the DALI 1.0 [12] dataset with a CTC objective, which permits us to train a model with weak annotations between audio and lyrics. Each song in the dataset is cut into windows of 20 seconds with a hop size of 10 seconds, which limits the size of the input audio feature sequence to a maximum length of 2000 frames. Due to the real-time constraints, we do not extract the vocals from the audio mixture but we train our model with original mixtures of singing vocals and polyphonic music. The corresponding labels of each window are extracted with the word-level annotations provided by the dataset. Each word starting and ending in the corresponding time interval is part of the target annotation sequence of the corresponding audio window. Then, the character sequence is transformed into a phoneme sequence using Phonemizer, specifying the correct language (which is known at training time). Empty sequences are classified with the instrumental token. During training, the weights of the model are tuned to maximize the probability of getting the correct label sequence (or all derivative sequences that have inserted repetitions or blank symbols), given the input feature sequence.

3.2 Alignment

The real-time alignment is realized between two audio performances of the same work, both containing the same sung lyrics. One performance, the *target*, corresponds to the live performance we want to align to the lyrics. The other performance, the *reference*, serves as a proxy to the written-out lyrics. This strategy has two main advantages. First, it is not dependent on a text-to-phoneme tool anymore. The pronunciation of the lyrics is contained in the reference recording and thus, can be decoded by the acoustic model. Even if the language of the target song is not included in the training set, the acoustic model maps both reference and target into the European phoneme set used during training. Also, the duration of the phoneme units is implicitly included in the reference recording, which means we do not need complex, explicit duration modeling techniques [9]. As a consequence, we can expect the reference posterioqram to be more similar to the posterioqram of the target recording we want to align to the lyrics. However, the reference has to be linked to the lyrics beforehand, generally with manual annotations at the word or sentence level – but this really only depends on the requirements of the specific application.

We use the OLTW [23] algorithm to align reference and target posterioqrams, skipping blank tokens in both reference and target sequence. The reference posterioqram is generated beforehand, feeding only the reference audio feature sequence to the acoustic model. Then, we generate in real-time, also with our acoustic model, the probability vectors representing the target posterioqram. For each new vector, we calculate its cosine distance with a range of 8000 posterioqram vectors, centered around the expected position in the reference posterioqram. This corresponds to a context of 320 seconds. Then, we calculate recursively the global cost, applying the standard DTW formula (equation (4.5) in [24]). The index representing the minimum of the global cost represents the current time position in the reference posterioqram.

4. DATA DESCRIPTION

In this work, we select opera recordings to evaluate our system, for three reasons. First, live opera would be a direct beneficiary of this tracking method, which would support a fully automatic subtitle display in the opera house (or in a live streaming application). Secondly, opera lyrics are challenging to track. Indeed, the genre of classical music has been considered by far as the least intelligible genre among eleven other genres [25]. Thus, evaluating our system on opera data is a good robustness indicator. Finally, opera is a musical genre that consistently produces identical works in several copies, with the change of the entire set of artists. Popular datasets in audio-to-lyrics alignment such as Hansen [26], Mauch [27] and Jamendo [4] do not include duplicate entries. The two opera datasets currently in our possession are described in Table 2.

The first is a subset of the Italian opera *Don Giovanni* by W.A.Mozart that covers all the *recitativo* sections. These

Opera	Name	Duration	# Annot.
Don Giovanni	Ref_Karajan	0:30:03	639
	Targ_Fischer	0:34:58	639
	Targ_Manacorda	0:30:40	639
Jingju	Ref_Jingju	1:53:53	3,975
	Targ_Jingju	3:22:03	9,567

Table 2. Description of Don Giovanni and Jingju datasets.

have been manually annotated with bar lines, making it possible for us to test the lyrics tracker by measuring how precisely it aligns target and reference at bar boundaries. Recitatives, an essential opera component of that period, have recently been in the focus of opera score following research [11], but trackers remain brittle. This is due to the liberty that singers can take in terms of timing, singing style, etc., and the fact that musical accompaniment is often improvised and played by different instruments in different recordings. Thus, it would be helpful to be able to follow the performance based on the content of the sung lyrics. As reference and proxy to the lyrics, we use a CD recording conducted by Herbert von Karajan in 1985. The two live targets are performances that were recorded at the Vienna State Opera in 2018 and 2019 and conducted, respectively, by Adam Fischer and Antonello Manacorda, with completely different casts of singers. For each performance, the complete subsections comprising the recitativo sections only, contain 639 manual bar-level annotations for a duration of approximately 30 minutes.

The second dataset we will use is a subset of the Jingju (Beijing Opera) A Capella Singing Audio Dataset [28, 29]. It has been recorded in a teacher/student manner, collecting a capella recordings from professional singers and singing students, which permits to get pairs of recordings. It is composed of two opera role types, *dan* and *laosheng*, and includes 20 different reference melodic lines of each role type with corresponding syllable-level annotations. The dataset has initially been recorded to evaluate the singing quality of the students compared to professional singers. This implies that the recordings sometimes contain mistakes in the lyrics, and breaks in between sentences. For each reference line, we count between 1 and 10 target versions that serve as target in our experiments.

5. EXPERIMENTS AND DISCUSSION

5.1 Acoustic Model Training

For our experiments, we train two distinct acoustic models, based on different training subsets of DALI dataset⁷. The first model, *5lang*, includes songs from five languages, namely English, German, French, Spanish, and Italian. The second model, *english*, uses only English data, the most represented language in the dataset. The different

⁷ We also tried to train a third acoustic model only on Italian data, which is the target language of the “Don Giovanni” opera. However, all alignments diverged. We believe that this is mainly due to the low amount of training Italian data, 10.8 h, in the DALI dataset, which is significantly lower than the other proposed languages.

Dataset	Train songs (duration)	Valid. songs (duration)
5lang	4027 (259.4 h)	149 (9.4 h)
english	3257 (210.8 h)	39 (2.6 h)

Table 3. Acoustic Model training and validation datasets.

train and validation splits were made publicly available by [7]⁸ and are described in Table 3. Each acoustic model is trained with the CTC loss, a learning rate of 10^{-4} , and the ADAM optimizer.

5.2 Evaluation Metrics

As mentioned above, we evaluate our lyrics trackers by quantifying the precision of the alignment between target and reference that they produce. The granularity of our ground truth annotations is at the bar level for Don Giovanni and at the syllable level for the Jingju dataset. We use the standard evaluation metrics from the field of score following [30]. For each alignment, we report the mean tracking error, in seconds, between timestamp annotations and times detected by our aligner. We also report the proportion of annotations that are detected with an error less than 1s.

5.3 Lyrics Tracking in Opera

To evaluate our system, we compare its performance with other following techniques working in real-time. To this end, we select the State-Of-The-Art (SOTA) live opera tracker that has recently proved to be robust to track, from beginning to end, complete live “Don Giovanni” performances [11, 31]. The opera tracker applies an OLTW algorithm to align reference and live target audio. Instead of using posteriograms as input to the OLTW algorithm, it takes audio features that are directly computed from the audio recordings. For the study, we compute two types of features that will serve as tracking baselines and inputs to our alignment algorithm. The first feature, *baseline*, has been inspired from music tracking systems performing on orchestral performances [32]. The feature calculates 120 MFCCs, but discards the first 20, and is computed at a sampling rate of 44.1 kHz, a window size of 20 ms, and a hop size of 10 ms. The second feature, *recitative*, was designed specifically with recitatives in mind [31]. It was tuned to perform best on the recitative subset of the Fischer recording and was shown to generalize well to the recitative subset of Manacorda. The feature is composed of 25 MFCCs extracted from Linear Predictive Coefficients (LPC) that aim at extracting the phoneme information from the audio. It is computed at a sampling rate of 1500 Hz, with the same previous window size and hop size.

The results are given in Table 4. Looking at Don Giovanni / Fischer, we see that both new models, using our *5lang* and *english* acoustic models, outperform the SOTA opera tracker based on *baseline* and *recitative* features,

⁸ <https://github.com/deezer/MultilingualLyricsToAudioAlignment>

Opera	Name	Feature	Mean(ms)	$\leq 1s$
DG	Targ_Fischer	baseline	1,915	66.1%
		recitative	955	76.5%
		5lang	846	80.5%
		english	818	80.5%
	Targ_Manacorda	baseline	1,503	62.0%
		recitative	1,023	69.7%
Jingju	Targ_Jingju	5lang	824	77.6%
		english	963	76.1%
		baseline	2,943	61.5%
		recitative	3,878	60.0%
	5lang	964	87.5%	
	english	810	89.0%	

Table 4. Tracking error on Don Giovanni (DG) and Jingju opera sub-datasets

the latter of which had been optimized specifically on this dataset. The mean error has been reduced by at least 100 ms and 80.5% of the bars now show an error below 1s. A similar picture emerges with DG / Manacorda, where the mean error goes to 824 ms for the best *5lang* model, and where the 1s threshold improves by 8 percentage points, relative to the recitative feature tracker. It is also important to note that the results of *recitative* and *baseline* were obtained in combination with a dedicated silence detector which halts the tracking process when there are obvious pauses. The two new trackers simply use the posteriograms generated by the acoustic models and still improve tracking accuracy.

Secondly, our two models *5lang* and *english* also perform best, by a large margin, on the Jingju opera sub-dataset. The *baseline* and *recitative* features were designed to extract pitch contours, focusing at different parts of the frequency range. They turn out to be inefficient at tracking Beijing Opera a capella recordings. On that task, the *english* model achieves the best performance, with a tracking accuracy of 810 ms and 89.0% of the syllables being detected below 1s of error.

5.4 Robustness to Different Languages

Comparing the results, we see that alignment accuracies are very similar across the two corpora, even though they contain singing signals in two very different languages, Italian and Chinese. The two acoustic models were not trained on Chinese recordings,⁹ a language that includes new phonemes that do not appear in the phoneme set built from the five European languages present in DALI. Using phoneme posteriograms as a joint intermediate representation of the lyrics and of the live input thus seems to be a remarkably robust choice for multilingual tracking.

Finally, the best results seem to alternate between the *5lang* and *english* acoustic models in the different sub-datasets. Based on the conclusions of [7], we expected the *5lang* model to perform best. However, and especially on the Jingju tracking experiment, *english* yields some-

⁹ and indeed, Don Giovanni’s language – Italian – was also not represented in the *english* acoustic model’s training data.

times better performance. This may be explained by the fact that the mapping into the European phoneme set is by construction incorrect, since the Chinese language is not in the training dataset. We expect that training our acoustic model with additional Chinese data would boost the performance.

6. CONCLUSION

We have presented an on-line audio-to-lyrics alignment method that is capable of operating in a real-time scenario. It involves an acoustic model, built from a ResNet architecture, that classifies each audio frame into a vector representing the probability distribution over a predefined set of phonemes, with a delay of 280 ms. In a second step, a real-time capable alignment algorithm (On-Line Time Warping) aligns the emerging sequence of vectors to a posterioriogram matrix that has been extracted beforehand from a reference performance of the same work, via our acoustic model. In experiments, we showed that our method is robust and reasonably precise in tracking the lyrics in a musical genre where the sung lyrics are known to be hard to understand, and a genre that was not part of the acoustic model training dataset. Additionally, we also showed robustness across languages, even if these are not included in the acoustic model training data. Our results suggest that it might be fruitful to investigate combinations of our system with existing music trackers in the more general task of opera score following.

Moreover, even if our study focused on the specific genre of opera (and on two very specific subsets of it), the method should be directly applicable to other music genres and other languages. The acoustic model was trained on Western musical genres and consequently, we expect it to work even better on those genres. As future work, we plan to evaluate our system on available Western music datasets containing pairs of recordings, such as Covers80 [33], where lyrics are not necessarily identical and where song structures may differ, and to use offline lyrics alignment systems to obtain reference annotations.

7. ACKNOWLEDGMENTS

The research is supported by the European Union under the EU's Horizon 2020 research and innovation programme, Marie Skłodowska-Curie grant agreement No. 765068 ("MIP-Frontiers"). The LIT AI Lab is supported by the Federal State of Upper Austria. Thanks to Andrea Vaglio, Emir Demirel, and Khaled Koutini for our interesting discussions about this work.

8. REFERENCES

- [1] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi," in *Proc. of the International Speech Communication Association Conference (INTERSPEECH)*, Stockholm, Sweden, 2017, pp. 498–502.
- [2] Y. Kida, T. Komatsu, and M. Togami, "Label-Synchronous Speech-to-Text Alignment for ASR Using Forward and Backward Transformers," *Submitted to INTERSPEECH 2021*, 2021.
- [3] G. Dzhambazov, S. Şentürk, and X. Serra, "Automatic Lyrics-to-Audio Alignment in Classical Turkish Music," in *Proc. of the Folk Music Analysis Workshop (FMA)*, Istanbul, Turkey, 2014, pp. 61–64.
- [4] D. Stoller, S. Durand, and S. Ewert, "End-to-end Lyrics Alignment for Polyphonic Music Using an Audio-to-Character Recognition Model," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, United Kingdom, 2019, pp. 181–185.
- [5] C. Gupta, E. Yılmaz, and H. Li, "Automatic Lyrics Alignment and Transcription in Polyphonic Music: Does Background Music Help?" in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 496–500.
- [6] E. Demirel, S. Ahlbäck, and S. Dixon, "Low Resource Audio-to-Lyrics Alignment From Polyphonic Music Recordings," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Toronto, Canada, 2021.
- [7] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. D'alché-Buc, "Multilingual lyrics-to-audio alignment," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020, pp. 512–519.
- [8] M.-Y. Kan, Y. Wang, D. Iskandar, T. L. Nwe, and A. Shenoy, "LyricAlly: Automatic Synchronization of Textual Lyrics to Acoustic Music Signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 338–349, 2008.
- [9] C. Gupta, H. Li, and Y. Wang, "Automatic Pronunciation Evaluation of Singing," in *Proc. of the International Speech Communication Association Conference (INTERSPEECH)*, Hyderabad, India, 2018, pp. 1507–1511.
- [10] A. Arzt and G. Widmer, "Real-Time Music Tracking Using Multiple Performances as a Reference," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015, pp. 357–363.
- [11] C. Brazier and G. Widmer, "Towards Reliable Real-Time Opera Tracking: Combining Alignment with Audio Event Detectors to Increase Robustness," in *Proc. of the Sound and Music Computing Conference (SMC)*, Turin, Italy, 2020, pp. 371–377.

- [12] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, “DALI: A Large Dataset of Synchronized Audio, Lyrics and Notes, Automatically Created using Teacher-Student Machine Learning Paradigm,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 431–437.
- [13] “Smule Sing! Dataset,” <https://ccrma.stanford.edu/damp/>, accessed March 15, 2018.
- [14] H. Sak, A. Senior, K. Rao, O. Irsoy, A. Graves, F. Beaufays, and J. Schalkwyk, “Learning Acoustic Frame Labeling for Speech Recognition with Recurrent Neural Networks,” in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, South Brisbane, Queensland, Australia, 2015, pp. 4280–4284.
- [15] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, “Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI,” in *Proc. of the International Speech Communication Association Conference (INTERSPEECH)*, San Francisco, CA, USA, 2016, pp. 2751–2755.
- [16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” in *Proc. of the International Conference on Machine Learning (ICML)*, Pittsburgh, Pennsylvania, USA, 2006, pp. 369–376.
- [17] V. Peddinti, D. Povey, and S. Khudanpur, “A Time Delay Neural Network Architecture for Efficient Modeling of Long Temporal Contexts,” in *Proc. of the International Speech Communication Association Conference (INTERSPEECH)*, Dresden, Germany, 2015, pp. 3214–3218.
- [18] B. Liu, W. Zhang, X. Xu, and D. Chen, “Time delay recurrent neural network for speech recognition,” in *Proc. of the Conference on Machine Vision and Information Technology (CMVIT)*, Guangzhou, China, 2019, pp. 769–775.
- [19] A. Kruspe, “Lyrics Alignment using HMM, Posteriogram-based DTW, and Phoneme-based Levenshtein Alignment,” in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2017.
- [20] K. Koutini, H. Eghbal-zadeh, M. Dorfer, and G. Widmer, “The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification,” in *Proc. of the European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain, 2019.
- [21] K. Koutini, F. Henkel, H. Eghbal-Zadeh, and G. Widmer, “Low-Complexity Models for Acoustic Scene Classification Based on Receptive Field Regularization and Frequency Damping,” in *Proc. of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)*, Tokyo, Japan, 2020, pp. 86–90.
- [22] K. Koutini, S. Chowdhury, V. Haunschmid, H. Eghbal-Zadeh, and G. Widmer, “Emotion and Theme Recognition in Music with Frequency-Aware RF-Regularized CNNs,” in *Proc. of the MediaEval Workshop*, Sophia Antipolis, France, 2019.
- [23] S. Dixon, “An On-Line Time Warping Algorithm for Tracking Musical Performances,” in *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, Scotland, UK, 2005, pp. 1727–1728.
- [24] M. Müller, “Dynamic Time Warping,” *Information retrieval for music and motion*, pp. 69–84, 2007.
- [25] N. Condit-Schultz and D. Huron, “Catching the Lyrics: Intelligibility in Twelve Song Genres,” *Music Perception: An Interdisciplinary Journal*, vol. 32, no. 5, pp. 470–483, 2015.
- [26] J. K. Hansen, “Recognition of Phonemes in A-Cappella Recordings using Temporal Patterns and Mel Frequency Cepstral Coefficients,” in *Proc. of the Sound and Music Computing Conference (SMC)*, Copenhagen, Denmark, 2012, pp. 494–499.
- [27] M. Mauch, H. Fujihara, and M. Goto, “Lyrics-to-Audio Alignment and Phrase-Level Segmentation using Incomplete Internet-Style chord annotations,” in *Proc. of the Sound and Music Computing Conference (SMC)*, Barcelona, Spain, 2010, pp. 9–16.
- [28] R. C. Repetto and X. Serra, “Creating a Corpus of Jingju (Beijing Opera) Music and Possibilities for Melodic Analysis,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 313–318.
- [29] R. Gong, R. C. Repetto, and X. Serra, “Creating an A Cappella Singing Audio Dataset for Automatic Jingju Singing Evaluation Research,” in *Proc of the Workshop on Digital Libraries for Musicology (DLfM)*, Shanghai, China, 2017, pp. 37–40.
- [30] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, “Evaluation of Real-Time Audio-to-Score Alignment,” in *International Symp. on Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007, pp. 315–316.
- [31] C. Brazier and G. Widmer, “Addressing the Recitative Problem in Real-time Opera Tracking,” in *Proc. of the Frontiers of Research in Speech and Music conference (FRSM)*, Silchar, India, 2020.
- [32] T. Gadermaier and G. Widmer, “A Study of Annotation and Alignment Accuracy for Performance Comparison in Complex Orchestral Music,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 769–775.

- [33] D. P. Ellis and G. E. Poliner, "Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking," in *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Honolulu, Hawaii, 2007, pp. 1429–1432.

VISUALIZING INTERTEXTUAL FORM WITH ARC DIAGRAMS: CONTOUR AND SCHEMA-BASED METHODS

Aaron Carter-Enyi
Morehouse College

aaron.carterenyi@morehouse.edu

Gilad Rabinovitch
Florida State University

grabinovitch@fsu.edu

Nathaniel Condit-Schultz
Georgia Institute of Technology

natcs@gatech.edu

ABSTRACT

The visualizations in Wattenberg's *Shape of Song* (2001) were based on pitch-string matching, but there are many other equivalence classes and similarity relations proposed by music research. This paper applies recent algorithms by Carter-Enyi (2016) and Carter-Enyi and Rabinovitch (2021) with the intention of making arc diagrams more effective for research and teaching. We first draw on Barber's intertextual analysis of Yorùbá *Oríkì*, in which tone language texts are circulated through various performances (Barber 1984). Intertextuality is exemplified through a 2018 composition by Nigerian composer Ayò Olúranti, then extended to Dizzy Gillespie's solo in his recording of "Blue Moon" (ca. 1952). Example visualizations are produced through an open-source implementation, ATAVizM, which brings together contour theory (Quinn 1997), schema theory (Gjerdingen 2007), and edit distance (Orpen and Huron 1992). Applications to the music of Bach and Mozart demonstrate that an African-centered analytical methodology has utility for music research at large. Computational music research can benefit from analytical approaches that draw upon humanistic theory and are applicable to a variety of musics.

1. INTRODUCTION

A promising visualization method for musical form and melodic relationships is the arc diagram, proposed by Wattenberg [1].¹ An arc diagram is a type of network diagram in which the nodes are aligned along a single axis. In the case of a single piece of music (intraopus), the axis represents time and the arcs connect matches between segments of the piece. In the case of multiple works (interopus), pieces might be arranged alphabetically or chronologically as nodes along the axes, and content matches may still be represented through a network of arcs. Applied to music, this visualization technique emphasizes melodic connections that permeate music instead of stressing sectional di-

¹ Wattenberg developed an art exhibit and web app, called the *Shape of Song* [2], which is now defunct.

visions. Understanding form as intertextual (both intraopus and interopus) fits with Africana musics and the zeitgeist of the information age; of being globally connected through the Internet. An intertextual approach is antithetical to western notions of musical works as individual artistic utterances.

In Wattenberg's *Shape of Song*, "Each arc connects two matching passages... [containing] the same sequence of pitches" [1]. This is a simplistic view of melodic associations. Arc diagrams have since been applied to audio and symbolic data to highlight a much greater variety of musical features [3, 4]. We present new visualizations produced using ATAVizM², which brings together contour theory [5, 6], schema theory [7], and edit distance [8]. Because these methods reveal patterns within and between works, we consider them intertextual approaches to music analysis. In Yorùbá *Oríkì*, tone language texts, such as *owé* (proverbs), are circulated through various performances [9]. Intertextuality is exemplified by Nigerian composer Ayò Olúranti's "Òmólúàbí" (2018) and Dizzy Gillespie's solo on "Blue Moon" (ca. 1952). Arc diagrams visualize the output of recent pattern discovery algorithms by Carter-Enyi [10] and with Rabinovitch [11]. The first algorithm was developed specifically for understanding the mapping of speech tone to song in African cultures. Applications of the same implementation to the music of Bach and Mozart demonstrate the extensibility of African-centered methodologies, which may contribute novel approaches to music research at large.

2. UNDERSTANDING FORM AS INTERTEXTUAL

Barber argues against seeing a "work of literature as an isolated artifact," considering the alternative of deconstruction as practiced by Derrida [9]. Taking Yorùbá praise-singing (*Oríkì*) as an example, Barber points out that the concept of "wholeness is simply inappropriate" for oral literature which is "constituted and reconstituted only through the participation of many people," including a "chain of performers" and audiences. Each performance is neither entirely unique nor entirely the same, they are related intertextually. When you deconstruct an *Oríkì* performance to short praise names, proverbs and riddles it is composed of, there is really no difference between interopus and intraopus analysis.

² Desktop and web versions of the app and source code are available: atavizm.org; radar.auctr.edu/atavizm; github.com/carterenyi/atavizm



The concept of intertextuality is not new to music scholarship. *Oríkì* itself has often been considered both a form of literature and a form of music. If Barber's description of *Oríkì* reminds you of the characteristics of jazz or hip hop, we agree. Thinking "interopus" about music challenges notions of authorship, copyright, and intellectual property. Notions that did not exist in many precolonial societies and still struggle to hold sway in many creative economies. Patterns may be found across works and throughout an individual work, repeated with variation. This is exemplified by Yorùbá proverbs Olúranti selected as the basis for "Ọmọ̀lúàbí" (2018). Echoed by Gillespie's improvisational gestures in his recording of "Blue Moon," which both quotes and hints at Lorenz/Hart's melody. Gjerdingen's schema theory [7] suggests that music making in 18th-century Europe relied on a shared stock of skeletal patterns, which explains the rapid creation of compositions and improvisations. Our application to "artworks" by Bach and Mozart will show how the Prinner schema emerges as a skeletal structure in two pieces with highly contrasting surface textures.

3. OPERATIONALIZING INTERTEXTUALITY

Musical themes, motives, and formulas are often transposed or modified and remain recognizable. Such relationships are relevant for both inter- and intraopus analysis. For example, a fugue subject is initially stated in the tonic key in a monophonic texture. The answer is a response in a polyphonic texture in a related key (usually dominant). Sometimes the answer is an exact transposition (real answer), and sometimes the answer is transposed and modified (tonal answer). Wattenberg offers the possibility of matching melodic intervals rather than pitch strings, which would match some subjects and answers, but not a subject and tonal answer, in which the subject is both transposed and modified. There are generic contrapuntal skeletons throughout 18th-century European music (schemata) that are embellished in various ways on the musical surface [7].

How do we formally relate the Yorùbá texts in poetry and in song? The "licks" of bebop musicians? A subject and tonal answer? Or schemata in 18th-century European music? Velardo et al. name four bases for Symbolic Melodic Similarity (SMS) systems: cognition, music theory, mathematics, and hybrid approaches (2017). The authors mention pitch-class set theory (with a basis in music theory and mathematics), but not contour theory (with a foundation in cognition, music theory, and mathematics, e.g., Quinn [6]), or schema theory (Gjerdingen [7]). This paper focuses on a contour-based method [12] and a schema-based method [11] to operationalize intertextuality on and (slightly) underneath the musical surface, respectively. Of the four pieces analyzed, only the Mozart analysis bears resemblance to a conventional form analysis (e.g., ABA for an entire work). Arc diagrams offer the possibility of new perspectives on form that are difficult to communicate without a network. Musical form in this context is an emergent property of intertextual relations.

4. A CONTOUR-BASED METHOD

Carter-Enyi's contour-level algorithm was developed based on an extensive analysis of Ọ̀gbò and Yorùbá speech, chant and music as well as a perceptual study with 1409 participants [12]. Contour levels are a modified contour theory based on a new understanding of tone-level languages and how tone levels are mapped to music. The following analyses demonstrate the method's effectiveness as a basis for analyzing and visualizing musical forms.

4.1 "Ọ̀mọ̀lúàbí" by Ayò Olúranti (2018)

As a composer, Olúranti carefully sets Yorùbá texts based on their implied pitch and rhythmic contour [13]. "Ọ̀mọ̀lúàbí" is based on a set of Yorùbá *owé* (proverbs and adages) about good character. The Yorùbá language has three tone-levels: low (à), mid (a, no accent), and high (á). Less is known about the extent to which rhythm is a contrastive element of speech, but it is clear that lexical tone distinguishes many words (47 % of two-syllable words [12]). The following are the six main proverbs incorporated into the work in order of their appearance. For each, the tone sequence is indicated with L for low tone, M for mid tone, and H for high tone.

1. *Ìwà l'è̀sin*³ (LL LL). Translation: Character is religion (meaning religion devoid of good character is worthless)
2. *Ìwà l'òba àwúre* (LL MM LHM). Translation: Supplication is important, good character even more.
3. *Ìwà l'òrìṣà, bí a bá ti hùú sí ni fíí gbè'ni sí* (LL LLL, H M H M LH H M MH LMH). Translation: Character is god (or salient). One's behavior determines one's attitude.
4. *Ká wí bẹ́ẹ́, ká bá a bẹ́ẹ́, iyí ènìyàn niyẹn* (H H H, H H M H, ML LLL LM). Translation: A man is honorable when his word is his bond.
5. *Ọ̀mọ̀lúàbí* (MMHLH). Translation: A child born of good character
6. *Tí a ó bàà jé Ọ̀sáká ká kúkú jé Ọ̀sáká. Bí a ó bàà jé Ọ̀soko ká kúkú jé Ọ̀soko. Ọ̀sáká-Ọ̀soko ko yẹni* (H M LH H MHH H HH H MHH. H M H LH H LMM H HH H LMM. MHH-LMM M MM). Translation: Be certain without doubt. Be uncertain without doubt. Being neither here nor there is undesirable.

Because Yorùbá is a tone language, the proverbs have implied contours which Olúranti adapts as themes and motives. Some longer proverbs are broken up or abbreviated (particularly 6). Instances of each tone-level sequence (strings and substrings in the symbolic data) are indicated as nodes between color-coded arcs in Figure 1. Because of the similarity in melodic shape but flexibility in pitch height and melodic intervals, the setting of tone language texts has an affinity to contour theory. However, contour theory, as formalized by Morris [5], could not be applied to this music because it first requires manual segmentation (making it incompatible with pattern discovery) and is intended for segments without any repeating pitches (such

³ Subdot indicates an open vowel or an "esh" instead of s. Underlining is substituted for a subdot where diacritics must be combined.

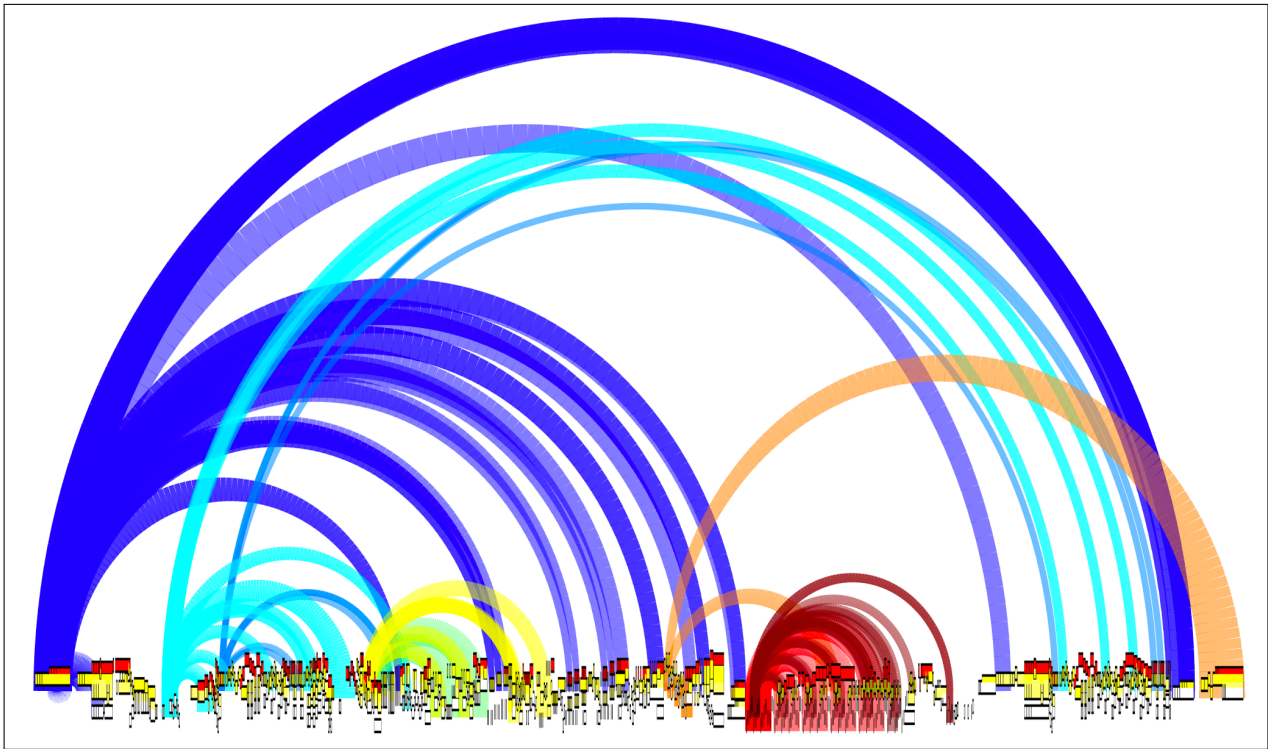


Figure 1. Contour-based intraopus arc diagram of Olúranti’s “Ọmọ́lúàbí” (2018)

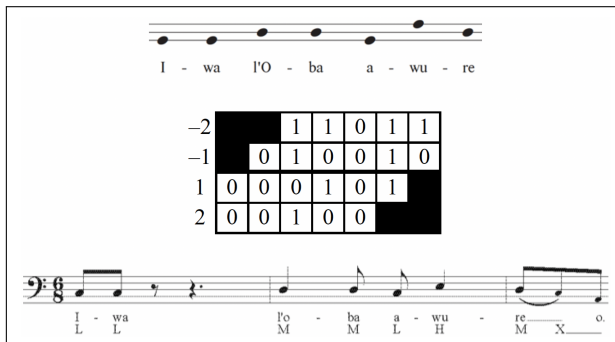


Figure 2. Contour equivalence class for three tone-level Yorùbá proverb and one of Olúranti’s settings.

as a 12-tone row). While Olúranti’s piece may be considered post-tonal and through-composed, repeated pitches are used, often to represent the repetition of linguistic tones.⁴ Carter-Enyi’s addition of “windowing” to create contour levels⁵ is based on the concept of tone levels in African languages [12] and is ideal for finding intertextual proverbs in Olúranti’s polyphonic composition (see Figure 2). Insertions (epenthesis) and deletions (elision) are common in language, so for language-based music allowing for edits to patterns is necessary. Contour level equivalence, as an effective associative method, may be enhanced through similarity metrics such as edit distance [8] when appropriate.

⁴ Schmuckler has applied concepts of contour theory computationally [14]. However, his method deviates from most music theory and has not been implemented as an open-source toolbox or application.

⁵ Quinn’s binary (0,1) annotation of pairwise comparisons was another necessary improvement over Morris [6]

The intertextuality of Olúranti’s piece is revealed in two ways by the diagram. First, the color-coding groups pattern instances by their link to a preexisting proverb (interopus). Second, the networks of equivalence and similarity are linked within the piece by arcs. Some proverbs occur in a limited range of the piece (notably “Kawibe...” and “Osaka...”), so are sectional within the piece, but still intertextually linked to Oríkì performance at large. Several proverbs come together in the final section, a textually and musically cohesive conclusion.

4.2 "Blue Moon" by Dizzy Gillespie (ca. 1952)

The notion of intertextuality is nothing new in jazz scholarship, signifying jazz’s spirit of both continuity and innovation. Carter-Enyi’s algorithm is apt for pattern discovery in jazz improvisation. Unlike Olúranti’s “Ọmọ́lúàbí,” where higher precision in contour matching is desirable, in this case, we applied a more inclusive match by reducing the “window size” and reducing “redundant” contour slices [10]. The degree of flexibility or rigidity in finding melodic relationships depends on the task at hand. In the context of jazz improvisation, more inclusive settings resonate with insider knowledge of listeners to jazz, who may recognize many extrapolations. Figure 4 includes parentheses around repeated notes to indicate that they were ignored in the analysis. To make the contour equivalence class more inclusive, only one degree of adjacency for contour comparison was used (see Figure 5). Additionally, a reduction technique was used where identical columns of the matrix are collapsed to a single column, indicated by multiple notes being mapped to a single contour node in Figure 5.

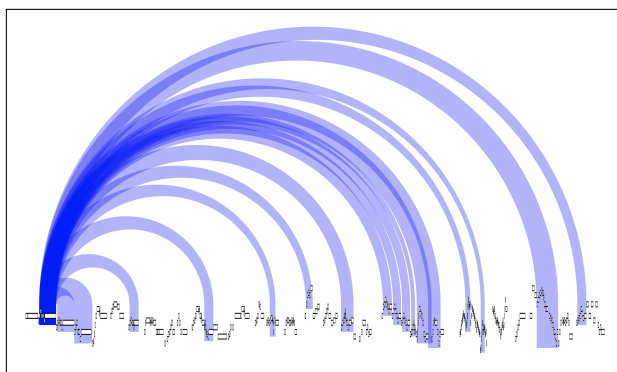


Figure 3. Contour-based inter/intraopus arc diagram of Gillespie's "Blue Moon" (ca. 1952)



Figure 4. Theme from "Blue Moon" (Rodgers/Hart 1934)

Figure 5's class identifies a wide variety of similar shapes as equivalent strings. A contour motive from the opening melody of "Blue Moon" reappears through Gillespie's solo.⁶ The most inclusive settings for Carter-Enyi's algorithm [10] are used to relate contours that are expanded or contracted. While it may be debatable whether some of the instances returned by the algorithm are indeed quotations of the theme, one might indeed question the gesture in Figure 5. However, in mm. 41-44, the quotation of the theme is apparent because of a sequence of descending thirds over iterations of the contour. This sequence is visible in the arc diagram (Figure 3 where there is a cluster of instances about two-thirds away through the piece. Although the contours in mm. 41-44 are in expanded form compared to the "head," these iterations are tied to the theme because of the use of sequence and the salience of similar scale degrees.

⁶ The transcription analyzed is here is by Jacques Gilbert, many more are available on his website [15]

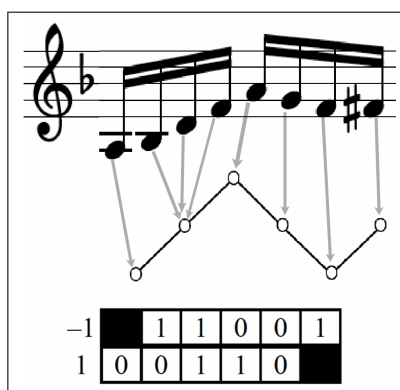


Figure 5. Improvisational gesture reduced to a simple contour equivalence class

Similar to Olúrantí's "Ọmọ̀lúàbí," Dizzy Gillespie's recording of "Blue Moon" is based on a preexisting "text," the Lorenz and Hart song. As in *Oríkí* and in Olúrantí's work, intertextuality is a core part of jazz, which in bebop has an objective quotation or recitation of a complete text (the "head") and a subjective interpolation of the text in the solo. Because the insider knowledge of listeners in the jazz tradition includes recognizing licks that seem to quote the head in the solo, we set the algorithm to be highly inclusive.

4.3 "Fugue in C Minor" by J. S. Bach (BWV 847)

The following applications demonstrate the effectiveness of an African-centered computational approach for discovering meaningful patterns in European classical music. In Figure 6, all statements of the fugue subject of BWV 847 are connected through blue arcs, representing a network between the introduction of the subject and all subsequent statements. The subject has cardinality 20 and, through eight iterations,⁷ accounts for over 20 % of the pitch content of the piece (160 out of 750 notes). Carter-Enyi presents a method for calculating a single equivalence class that incorporates all subject entries and no other segments using his contour level approach [12]. Specifically, a 4-degree window of adjacency is sufficient to recognize all subject statements without any false positives (see Figure 7).

5. A SCHEMA-BASED METHOD

Gjerdingen's schema theory posits shared skeletal contrapuntal patterns in European music of the 18th century. A limited number of skeletal prototypes in composition and improvisation resonates with existing theories of counterpoint and keyboard playing (figured and unfigured bass traditions). Schemata may be more abstract than the contours of Yorùbá proverbs, yet schema theory also suggests that 18th-century European music is also an intertextual tradition rather than a collection of distinct "masterworks." Therefore, schemata are important tools for interopus algorithmic analysis. The following analysis relates pieces by Bach and Mozart that are contrasting on the surface, yet draw extensively on Gjerdingen's Prinner schema: a soprano skeleton of 6-5-4-3 above a bass skeleton of 4-3-2-1 or 4-3-2-5-1, typically harmonized with IV-I6-vii6-I or a similar harmonic progression.

We have operationalized schemata as scale-degree skeletons that may be matched in various reductions of the same piece [11]. The method awards points in a score matrix for different syntagmatic (relational) features of notes. In Figure 8, the first row awards points to notes on a strong beat (beat 1 or 3). The second row awards points to notes within a step of notes in the previous window (delineated by bold lines) that already have points in the first row ("pre-contiguity"). The third row awards points similarly to the second row but for contiguity with high-ranking notes in

⁷ There is agreement among music analysts (including Bruhn and Schenker) where subject statements are in this fugue.

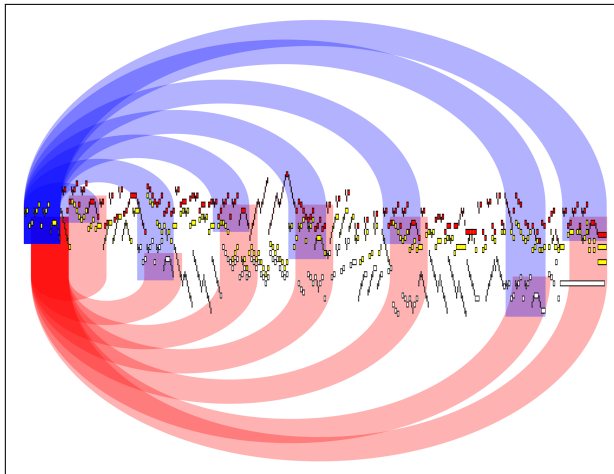


Figure 6. Contour- and Schema-based intraopus arc diagram for Bach's "Fugue in C minor" (BWV 847)



Figure 7. Contour Equivalence Class in BWV 847

the next window ("post-contiguity"). Finally, the points are totalled and the highest ranking note in each window is selected, producing the reduction, in this case 6-5-4-3, similar to Gjerdingen's Prinner schema.

5.1 Convergence of Contour and Schema

In BWV 847, the subject's surface contour and the underlying schema converge with a nearly identical interpretation of the piece, except the contour class has an anacrusis and the schema starts with a strong beat. Figure 6 includes the contour network in blue and the schema network in red. Either the contour-level algorithm [12] or schema-based method [11], when applied to BWV 847, identifies all appearances of the subject as equivalent in all three voices of the fugue, whereas pitch strings, interval strings, or other implementations of contour theory would not identify them fully. Either algorithm enables pattern discovery without any specific information about the genre (e.g., the main theme is at the beginning) to find the most prevalent pattern and visualize it. Depending on whether one wishes to demonstrate interopus connections, intraopus connections, or both, one may prefer the contour or schema-based approach. The contour in Figure 7 is very specific to BWV 847. The reduction in Figure 8 relates the subject to portions of many other pieces, including the primary theme in K545 (Figure 9).

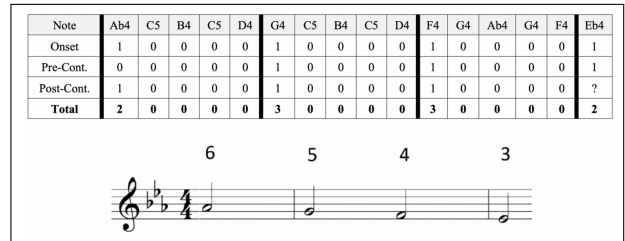


Figure 8. Schema Equivalence Class in BWV 847

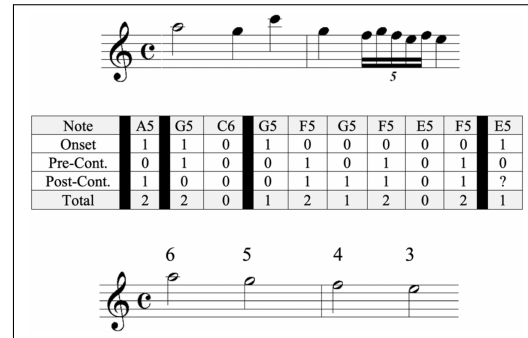


Figure 9. Schema Equivalence Class in K545 (mm. 3-4)

5.2 Divergence of Contour and Schema

In contrast, the contours and schemata suggest very different formal understandings of Mozart's "Sonata in C Major" (K545). The arcs above the piano roll in Figure 10 represent a contour-based network. Thinking in terms of melodic shape highlights the conventional sonata form, emphasized by the return of the primary theme at the beginning of the recapitulation (terminus of the red arc). The development is identified by a distinct motive (in yellow) which forms the core [16]. The schema-based network is quite different, shown as downward consecutive arcs. Initial arcs connect the first appearance of musical material to all other instances. However, consecutive arcs do not place any emphasis on the first instance. Consecutive arcs are appropriate for a schema pattern, which is highly intertextual, but not thematic in the traditional sense. Schemata are more minimalistic or generative because much more material can be accounted for by a single equivalence class.

6. CONCLUSION

We are trying to decenter music research from two angles: exploring intertextuality, debunking notions of art as created by individuals [9]; and, applying African-centered approaches to analysis (based on [12]). An interopus diagram (Figure 11) summarizes the connections between the works discussed, with contour matches in blue and schema similarity in red. Yorùbá Oríkì is not western diatonic music, so schema theory is not applicable. However, the schema approximation of scale-degree skeletons may be relevant to postcolonial music of Africa, such as "Ọmọlúàbí." Vice-versa, one may not expect contour levels (based on tone levels) to apply to European classical music because there is an emphasis on idiosyncratic

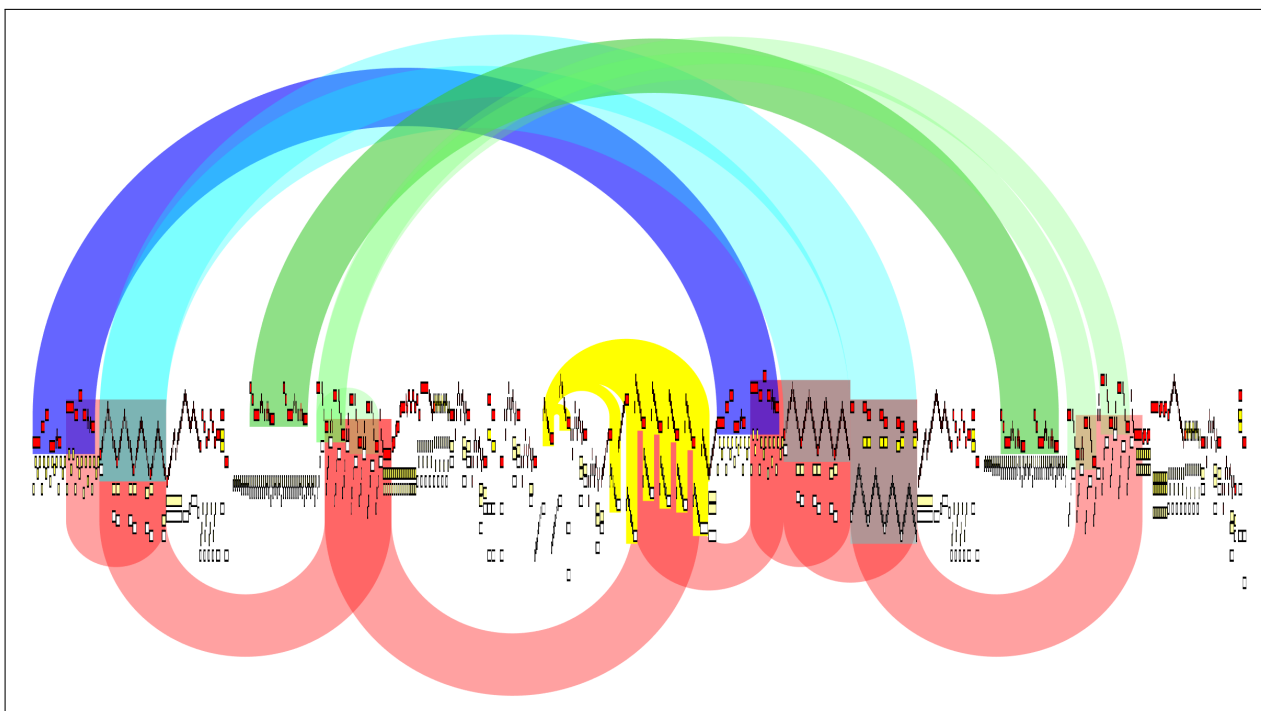


Figure 10. Contour (above) and Schema-based (below) intraopus arc diagram for Mozart's "Sonata in C Major" (K545)

themes. However, part of the take-away of schema theory is that European musicians in the 18th century relied on a small number of skeletal formulas, so contour motives are likely interopus as well, which questions the notion of distinct "masterpieces." It is only in the relationship between Rodgers/Hart's "Blue Moon" and Gillespie's improvisation on it that we (appropriately for jazz) see a match of both contour and schema.⁸ Contour and schema, although copresent, are very independent. Gillespie tests the bounds of the tonal paradigm of the song, making the theme an improvisational gesture that is independent from the harmonic progression, while the rhythm section maintains it with some liberty. Similarly, Mozart used the Prinzer schema with his main theme and independently of it. This contrasts with the total interdependence of contour and schemata in Bach's subject. In improvisation, we expect contrapuntal and harmonic patterns to be stretched by a soloist's wanderings. However, preservation of both contours and scale-degree skeletal schemata suggests intentionality. Bach delineates the subject entrances from the liminal space of the episodes except for a few motivic fragments of the subject. Similarly, Dizzy Gillespie explores the motive of the "Blue Moon" theme, but only once echoes the full sequence.

Beyond pure research, ATAVizM has applications in education and teaching music theory as an application for visualizing musical forms designed to be integrated into the undergraduate theory curriculum. Since 2017, the software has been used to teach Fugue, Sonata, and Post-Tonal music in music major courses and Indian Classical, African

⁸ Although this was not visualized with an arc, the thematic sequence with the model repeated at an interval of a descending diatonic third is replicated at one point by Gillespie in the solo. Both the instance in the theme and the solo form a similar scale-degree skeleton.



Figure 11. Interopus arc diagram

Choral, and popular music in general education courses in several US institutions. For teaching contexts, the feature of the user being able to select which strings are thematic is essential. Students make selections based on their analysis of a piece to produce visualizations for analytical papers. The open-source software presented here includes (1) pattern-matching algorithms based on heuristics from music theory (contour and schema) that may be applied to pitch (and duration in the case of contour), (2) theme identification by user input or selection, and (3) color-coding of arcs and a legend. Combining computationally efficient methods for contour and schema pattern recognition with ample features for user direction has brought us closer to reaching the full potential of Wattenberg's vision for seeing the *Shape of Song*.

7. REFERENCES

- [1] M. Wattenberg, "Arc diagrams: Visualizing structure in strings," in *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002*. IEEE, 2002, pp. 110–116.
- [2] —, "The shape of song," Available at <http://www.bewitched.com/song.html> (2021/05/01).
- [3] H.-H. Wu and J. P. Bello, "Audio-based music visualization for music structure analysis," in *Proceedings of Sound and Music Computing Conference (SMC)*, 2010, pp. 1–6.
- [4] D. F. Silva, C.-C. M. Yeh, G. E. Batista, E. J. Keogh *et al.*, "Simple: Assessing music similarity using sub-sequences joins." in *ISMIR*, 2016, pp. 23–29.
- [5] R. Morris, "Composition with pitch-classes," *Yale University*, 1987.
- [6] I. Quinn, "Fuzzy extensions to the theory of contour," *Music Theory Spectrum*, vol. 19, no. 2, pp. 232–263, 1997.
- [7] R. Gjerdingen, *Music in the galant style*. New York: Oxford University Press, 2007.
- [8] K. Orpen and D. Huron, "Measurement of similarity in music: A quantitative approach for non-parametric representations," *Computers in Music Research*, vol. 4, 1992.
- [9] K. Barber, "Yorùbá 'oríkì' and deconstructive criticism," *Research in African literatures*, vol. 15, no. 4, pp. 497–518, 1984.
- [10] A. Carter-Ényì, "Contour recursion and auto-segmentation," *Music Theory Online*, vol. 22, no. 1, 2016.
- [11] A. Carter-Enyi and G. Rabinovitch, "Onset and contiguity: Melodic feature reduction and pattern discovery," *Music Theory Online*, vol. 27, no. 4, 2021.
- [12] A. Carter-Enyi, "Contour levels: an abstraction of pitch space based on african tone systems," Ph.D. dissertation, The Ohio State University, 2016.
- [13] S. Oluranti, "Polyrhythm as an integral feature of african pianism: Analysis of piano works by akin euba, gyorgy ligeti & joshua uzoigwe and àjùlò kìnìún (original composition)," Ph.D. dissertation, University of Pittsburgh, 2012.
- [14] M. A. Schmuckler, "Melodic contour similarity using folk melodies," *Music Perception*, vol. 28, no. 2, pp. 169–194, 2010.
- [15] J. Gilbert, "Jazz trumpet transcriptions from the trumpet kings of the swing era," Available at http://pubcs.free.fr/jg/jazz_trumpet_transcriptions_jacques_gilbert_english.html (2021/05/01).
- [16] W. E. Caplin, *Classical form: A theory of formal functions for the instrumental music of Haydn, Mozart, and Beethoven*. New York: Oxford University Press, 1998.

UNSUPERVISED DOMAIN ADAPTATION FOR DOCUMENT ANALYSIS OF MUSIC SCORE IMAGES

Francisco J. Castellanos Antonio-Javier Gallego Jorge Calvo-Zaragoza

Department of Software and Computing Systems, University of Alicante, Spain

[fcastellanos, jgallego, jcalvo]@dlsi.ua.es

ABSTRACT

Document analysis is a key step within the typical Optical Music Recognition workflow. It processes an input image to obtain its layered version by extracting the different sources of information. Recently, this task has been formulated as a supervised learning problem, specifically by means of Convolutional Neural Networks due to their high performance and generalization capability. However, the requirement of training data for each new type of document still represents an important drawback. This issue can be palliated through Domain Adaptation (DA), which is the field that aims to adapt the knowledge learned with an annotated collection of data to other domains for which labels are not available. In this work, we combine a DA strategy based on adversarial training with Selectional Auto-Encoders to define an unsupervised framework for document analysis. Our experiments show a remarkable improvement for the layers that depict particular features at each domain, whereas layers that depict common features (such as staff lines) are barely affected by the adaptation process. In the best-case scenario, our method achieves an average relative improvement of around 44%, thereby representing a promising solution to unsupervised document analysis.

1. INTRODUCTION

Optical Music Recognition (OMR) is a computational process that aims to read the music notation from scanned documents and export their content to a structured digital format [1]. The countless number of music manuscripts scattered around the world, along with their high variability due to the different engravings, writing styles, ink colors, notations, or even the period in which they were written, represents a great obstacle to tackle this task in a simple way. In addition, physical formats are inevitably associated with page degradation over time, which is one of the motivations for digitizing them.

Given the complexity of OMR, the process is typically divided into a series of sequential tasks with partial goals.

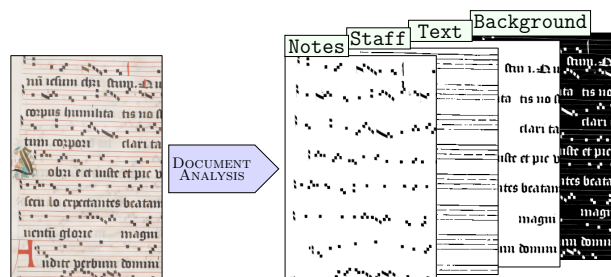


Figure 1: Overview of the document analysis process for music score images.

Document analysis is usually one of the most important tasks, where the relevant elements that make up the image content are recognized and extracted as different layers of information, e.g. by classifying each pixel into a set of categories such as staff lines, music notes, lyrics or background [2, 3], as shown in Figure 1.

Recent advances in machine learning, and particularly in Deep Neural Networks (DNNs), have opened up opportunities to carry out OMR processes effectively [4]. However, in spite of their high performance and demonstrated generalization capability in multiple tasks, this formulation brings an important drawback: the need for training data. Indeed, this is a common issue associated with machine learning, which requires labeling (often manually) a representative part of the data. However, the large number of manuscripts to be digitized contributes to making this an unaffordable task, so it is of great interest to reformulate this supervised problem to an unsupervised one.

Domain adaptation (DA) is a field that studies how to adapt the knowledge learned from a labeled collection of data—source domain—to another related, but different one—target domain—in an unsupervised manner. The idea behind this is the learning of domain-invariant features or a common representation between the source and the target domains. In this way, a model is able to process images from the target domain without using ground-truth information of that domain, thus eliminating the requirement for labeling images given a new domain. Note that, in the DA context, although the source domain labels are available, as the goal is to adapt to the target domain (for which there are no labels) this type of problem is considered unsupervised [5].

In this paper, an unsupervised approach based on Selectional Auto-Encoders (SAEs) and adversarial training



by means of a Gradient Reversal Layer (GRL) is proposed to carry out document layout analysis. The goal of this proposal is to recognize different layers of information—such as staff lines, notes, lyrics, and background—without having to manually label images of each new domain. The goodness of our approach is assessed through experiments with corpora of different music notations, reporting a substantial improvement in the performance depending on the layer at issue.

The rest of the paper is organized as follows. A review of related work is discussed in Section 2. The formulation of the problem and the description of the methodology are included in Section 3. The experimental setup and the empirical results are reported and analyzed in Section 4. A complementary qualitative evaluation is performed in Section 5. Finally, Section 6 summarizes the main conclusions, pointing out some potential future work.

2. RELATED WORK

Document analysis is a well-known stage within OMR [6], already studied in the literature with different strategies. Traditionally, this problem was addressed by dividing the task into several smaller consecutive steps. An example is binarization—used to split foreground and background information—for which we can also find different solutions, including traditional algorithms [7–9] or even specific approaches for music documents [10, 11]. There are also works in which staves and lyrics are split so that they can be processed separately, such as [12]. Another common step is the staff-line removal, where staff lines are eliminated to isolate the music symbols and make easier their classification. Dalitz et al. [13] reviews traditional methods, however, this is an active research field in which new work continually appears [14, 15].

More recently, there is a tendency to formulate document analysis as a machine learning problem. Given its performance and efficiency, the SAE architecture has been explored for related purposes, such as staff-line removal [16]. In addition, a SAE-based framework [17] was proposed to detect different layers of information by training a set of models to recognize each layer separately. However, although these approaches are usually aligned to high performance and generalizability, they entail a drawback derived from supervised learning: the need to manually label a portion of each manuscript to generate training data.

DA aims to palliate this issue by adapting the knowledge learned from a labeled (or source) manuscript to process another but related unlabeled (or target) manuscript in an unsupervised fashion. The adversarial training highlights within this field, which is an adaptation strategy in which different neural networks—or parts of them—are configured as opposing sides, with the aim of learning a common representation equally applicable to both domains. A relevant example is Domain-Adversarial Neural Network (DANN) [18], which presents a categorical neural network combined with a special type of layer named Gradient Reversal Layer (GRL). This layer aims to learn

domain-invariant features to perform the DA process.

In this work, we propose to extend the SAE-based supervised framework proposed in [17], in order to combine it with the GRL so that it performs the learning of domain-invariant features in an unsupervised fashion. While this idea has been proven successful for document binarization [19], we study here its performance for the document analysis of music score images.

3. METHOD

3.1 Problem formulation

Let \mathcal{S} be an annotated or *source* domain composed by a set of images with their corresponding ground truth ($\mathcal{X}_S, \mathcal{Y}_S$), where \mathcal{X}_S contains the scanned images of documents represented as $\mathcal{X}_S^i = [0, 255]^{h_s^i \times w_s^i \times c}$, being \mathcal{X}_S^i the i -th image within \mathcal{X}_S with height h_s^i px., width w_s^i px., and c channels, being $c = 1$ for grayscale and $c = 3$ for colored images; and \mathcal{Y}_S standing for a pixel-wise annotation of each \mathcal{X}_S^i image for a particular layer, with $\mathcal{Y}_S^i = \{0, 1\}^{h_s^i \times w_s^i}$, where 1 represents the foreground—or ink—of the layer at issue (e.g. staff lines, notes, text, etc.) and 0 the background.

Let \mathcal{T} be a non-annotated or *target* domain that consists of a series of images \mathcal{X}_T , being $\mathcal{X}_T^k = [0, 255]^{h_t^k \times w_t^k \times c}$ the k -th image with no labeling data available.

3.2 Document analysis framework with SAEs

Our approach builds upon the supervised state-of-the-art document analysis framework proposed by Castellanos et al. [17], which processes the input images to classify each pixel into a set of possible categories—staff lines, notes, text and background. This method is based on a series of SAE models—namely four models, one per layer—trained to individually recognize each layer of information in a supervised fashion. Note that the number of models represents the number of layers of information of interest, so the method could easily be extended by using as SAEs as layers to detect.

This architecture consists of two parts: an encoder, in which data are processed by a series of consecutive convolution and down-sampling layers, and a decoder, composed of convolution and up-sampling layers, as many as down-sampling layers are in the encoder. The output of the SAE model is a probabilistic map of the same size as the input, but with only one channel, in which the probability of each pixel belonging to a specific layer is computed. This scheme can be successfully trained in a supervised manner when there are ground-truth data available for a portion of the collection to be processed.

As mentioned above, the framework processes each category separately with the aim of modeling specialized SAEs that detect individual layers of information to eventually be processed or combined. In order to combine the individual decisions to provide an actual document analysis result, we eventually label each pixel as that category for which its SAE retrieves the highest probability. This combination is mathematically defined as

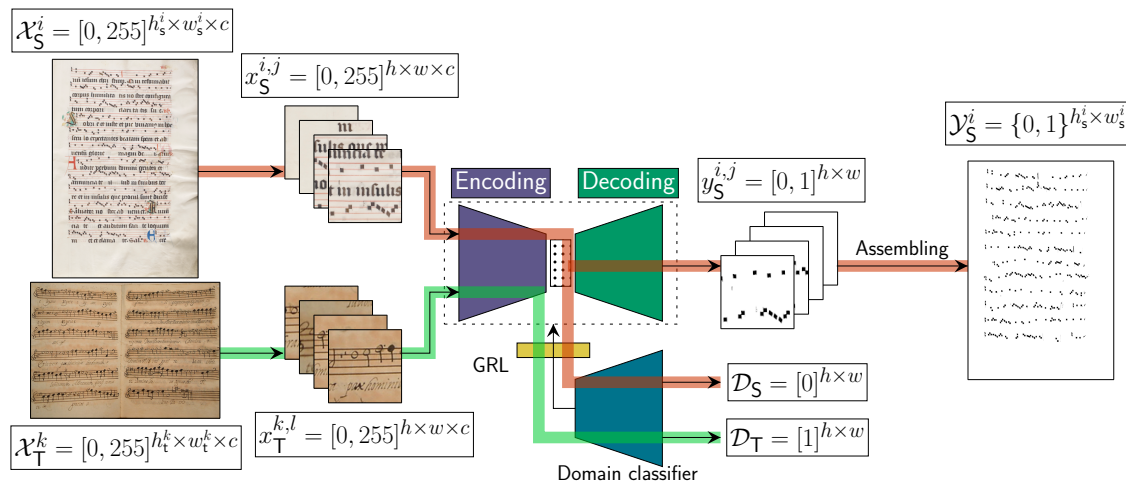


Figure 2: Scheme of our approach for the case of recognition of music notes. It would be repeated for each new layer to be considered by using its own SAE model trained with the source ground-truth data \mathcal{Y}_S^i for each layer.

$$\mathcal{Y}_T^k = \arg \max_{c \in \Sigma} P(c | \mathcal{X}_T^k)$$

where Σ represents the entire set of classes or layers of interest.

It should be noted that the SAE model must be trained by patches, therefore, each input image is split into a set of chunks of $h \times w \times c$ px. that are individually processed. Therefore, once the prediction is made, it is necessary to assemble all these patches to finally build the full layered image.

3.3 Unsupervised domain adaptation approach

The method presented in the above section can successfully deal with the document analysis task when there are representative training data of the collection to be processed. However, this requires manually labeling some images of each target manuscript to provide enough training data for the learning process. Within this context, DA plays an important role to enable the application of recognition models when there are no annotations but for one source domain \mathcal{S} .

Our approach addresses the problem of how to adapt a document analysis model for processing an unlabeled target collection \mathcal{T} . Given such conditions, the model must be adapted in an unsupervised way. We propose the use of GRL [18], originally designed for classification tasks, to face this challenge.

The GRL-based approach makes use of adversarial training to penalize domain-specific features in order to train a neural network model capable of dealing with images from \mathcal{S} or \mathcal{T} , indistinctly. This special layer is connected to a domain classifier that takes advantage of the only information available for \mathcal{T} : the certainty that the \mathcal{X}_T images belong to a different domain than the source. Hence, the domain classifier shall try to identify whether, given an image \mathcal{X}_S^i or \mathcal{X}_T^k , it belongs to \mathcal{S} or \mathcal{T} . This classifier, therefore, will look for domain-specific features that allow the images of both domains to be differenti-

ated. However, as it is connected to the SAE architecture through the GRL, the gradients calculated as consequence of this classification are reversed in the training process. That is, GRL penalizes the domain-specific features found by the domain classifier, thus achieving a SAE model which focuses on domain-invariant features. Note that GRL includes a hyper-parameter λ to adjust the contribution of the domain classifier in the training process, to be empirically studied according to the task.

A graphical outline can be found in Figure 2 with an example of this method for a single layer, that of note symbols. The idea of our approach is to use independent SAEs, each one trained with the ground truth of \mathcal{S} for a specific layer. Thus, our approach applies document analysis through four SAE models, one for each layer of information, and finally combines these results based on the probability of the output layer.

4. EXPERIMENTAL SETUP

4.1 Corpora

For our experiments, we selected three corpora manually labeled for the considered layers. The details of these datasets are listed below (some examples of images can also be found in Figure 3):

- EINSIEDELN: collection of 10 music documents in Neumatic notation, specifically those of Einsiedeln, Stiftsbibliothek, Codex 611(89)¹ with an average size of $6\,496 \times 4\,872$ px.
- SALZINNES: set of 10 music score images in Neumatic notation with an average resolution of $5\,847 \times 3\,818$ px., of Salzinnes Antiphonal (CDM-Hsmu2149.14)².
- CAPITAN: 10 images from a complete *Missa* of the second half of the 17th century [20] in Mensural notation with an average size of $2\,126 \times 3\,065$ px.

¹<http://www.e-codices.unifr.ch/en/sbe/0611/>

²<https://cantus.simssa.ca/manuscript/133/>

Table 1: Description of the SAE architecture considered, implemented as a Fully-Convolutional Network (FCN). Notation: $\text{Conv}(f, h_c, w_c, a)$ represents a convolution operator of f filters, with kernels of $h_c \times w_c$ pixels and an a activation function; $\text{MaxPool}(h_p, w_p)$ indicates a max-pooling operator with a $h_p \times w_p$ kernel; $\text{UpSamp}(h_u, w_u)$ stands for an up-sampling operator of $h_u \times w_u$ px.; ReLU and Sigmoid denote Rectifier Linear Unit and Sigmoid activations, respectively.

Input	Encoding	Decoding	Output
[0, 255] ^{256×256}	Conv(64,3,3,ReLU)	Conv(64,3,3,ReLU)	[0, 1] ^{256×256}
	MaxPool(2,2)	UpSamp(2,2)	
	Conv(64,3,3,ReLU)	Conv(64,3,3,ReLU)	
	MaxPool(2,2)	UpSamp(2,2)	
	Conv(64,3,3,ReLU)	Conv(64,3,3,ReLU)	
	MaxPool(2,2)	UpSamp(2,2)	
		Conv(1,3,3,Sigmoid)	



(a) EINSIEDELN (b) SALZINNES



(c) CAPITAN

Figure 3: Examples of some representative regions of the images from the corpora.

4.2 Metrics

Given the imbalance nature in the distribution of the classes, F-score (F_1) was considered for the evaluation of the method. In a two-class problem, it is defined as:

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}, \tag{1}$$

where TP, FP, and FN stand for *True Positives* or correctly classified elements, *False Positives* or type I errors, and *False Negatives* or type II errors, respectively. However, since the experiments will be conducted as a multiple-class problem, we considered reporting the results in terms of F_1 for each class and macro- F_1 [21] for a global evaluation, which is calculated as the average of the F_1 obtained for each class.

Considering that, in our context, the ground-truth data is often subjective—especially for edge pixels—and that there are also multiple thin elements—such as staff lines—we decided to use more suitable metrics for this task, such as those explained in [2]. Thus, we report the results in terms of this pseudo- F_1 , henceforth ps- F_1 . This metric considers as TP those pixels whose real class matches with the prediction in any vertically and horizontally adjacent pixel.

4.3 Hyper-parameterization

Since this paper addresses an unsupervised formulation for document analysis through a DA scheme, we considered the use of SAE as the basis of the model. Table 1 indicates a detailed description of the neural network, which shall be repeated for each layer of interest like in the state-of-the-art document analysis framework [17]. Note that, for simplification reasons, the input image is given in grayscale, although another color space might be used.

As described, an SAE is trained with patches extracted from the input images. These patches are randomly selected after each training epoch, for the sake of data variability. We considered patches of 256×256 px. Concerning the GRL, we connect it before the last convolutional block of the decoder, with $\lambda = 0.01$ and increments of 0.001 per epoch. These decisions were taken by informal testing. The convolutional weights are optimized by using the well-known stochastic gradient descent [22] with a batch size of 12. We carried out a pre-training step with only \mathcal{S} for 50 epochs, before the GRL and target images become involved, up to a total of 300 epochs, taking 10 000 samples per epoch from each domain. Note that our approach extracts the same number of samples for each domain to properly balance them.

It is worth mentioning that data are divided into partitions for training, validating, and testing, with 60%, 20%, and 20% of the entire collections, respectively. The validation partition is used to choose the best model in \mathcal{S} , assuming the premise that learning domain-invariant features would allow to similarly process source and target images. Although this partitioning is only necessary for the source domain, we applied it in all cases for consistent evaluation.

Table 2: Average results, in terms of ps-F₁ (%), for the SAE-based framework—state of the art—and our document analysis approach based on GRL. The results are organized according to the music notation of \mathcal{S} and \mathcal{T} . The best figures between both models are highlighted in bold.

$\mathcal{S} \rightarrow \mathcal{T}$	Framework	
	SAE-based	SAE-DANN-based
<i>Neumatic</i> → <i>Mensural</i>		
EINSIEDELN → CAPITAN	48.7	60.0
SALZINNES → CAPITAN	31.5	55.6
Avg.	40.1	57.8
<i>Mensural</i> → <i>Neumatic</i>		
CAPITAN → EINSIEDELN	44.7	45.1
CAPITAN → SALZINNES	55.3	53.5
Avg.	50	49.3
Avg.	45.1	53.6

4.4 Results

In this section, we assess our GRL-based approach and compare it with the state of the art [17] in unsupervised domain-adaptation scenarios. Note that our corpora contain different music notational systems—Neumatic and Mensural. Thus, we consider of great interest the applicability of our method to adapt images across different notational systems, depicting obvious differences at the graphic level and making document analysis very challenging.

Table 2 shows the average results for each pair of \mathcal{S} and \mathcal{T} considered for experimentation, in such a way that source and target manuscripts do not match in the type of music notation. Focusing on the first section of the experiments, when $\mathcal{S} \equiv \text{Neumatic}$ and $\mathcal{T} \equiv \text{Mensural}$, we observe a clear improvement of the DA approach with respect to the state of the art. Our approach increases the ps-F₁ from 40.1% to 57.8% on average, which represents a substantial relative improvement of 44.1%.

Concerning the second section of results, those in which \mathcal{S} contains pages in *Mensural* notation and \mathcal{T} consists of *Neumatic* documents, we realize that two different situations are presented: the CAPITAN → EINSIEDELN case, with a slight improvement from 44.7% to 45.1%, and the CAPITAN → SALZINNES, in which the DA technique is not able to learn adequate features for \mathcal{T} , thus reducing until 2% approximately. Despite this drawback, we may consider it as marginal, representing changes barely perceptible in the layered resulting image with respect to the state-of-the-art method.

This phenomenon may be attributed to the fact that the filters of the neural network are not able to extract domain-invariant features by using only the ground truth of the unique labeled domain in these experiments—CAPITAN. For example, it should be noted that the complexity and variability of the types of music symbols in *Mensural* notation are considerably greater than those in *Neumatic* one, which presents very uniform symbols and very different to those in CAPITAN. Besides, the text within CAPITAN shows a certain degree of degradation and different contrast levels with respect to the rest of the ink, even

Table 3: Average results for each layer of information. The figures are reported in terms of ps-F₁ (%). Note that the “Bg.” column represents the background layer. The best results per layer are remarked in bold.

Framework	Staff	Note	Text	Bg.	Avg.
<i>State of the art</i>					
SAE-based [17]	80.6	39.0	8.9	51.8	45.1
<i>Our approach</i>					
SAE-DANN-based	82.3	42.4	23.5	66.0	53.6

within the same page. These aspects, as we shall show in Section 5, may hinder the learning of common features for both domains, since there are layers of information with strong differences between themselves.

In order to complement the analysis, Table 3 shows the average results for each layer of information obtained with the SAE-based framework and with our method. We can observe that the DA approach obtains average improvements for all the layers considered. It is worth mentioning that the staff lines in both music notations are very similar visually. Mainly, this is why the SAE specialized in recognition of staff lines from images of \mathcal{S} may be able to deal with those staff lines of \mathcal{T} . Indeed, focusing on this layer, we realize that the SAE model obtains a high performance of 80.6%. Note that these results are obtained by unsupervised experiments, and also note that the ps-F₁ obtained for the staff layer precisely outperforms all the rest of the layers considered. This means that the SAE model without DA mechanisms is enough for extracting features from \mathcal{S} capable of detecting staves from \mathcal{T} . In spite of this, we can observe a slight improvement, achieving 82.3% of ps-F₁.

Concerning the rest of the layers, the increase of performance of the text and background layers is especially relevant. Although in the case of text, the performance may seem low with only 23.5% of ps-F₁, note the radical enhancement with a relative figure over 164%. Besides, the background layer also has a relative boost of 27%, supposing significant contributions for the unsupervised document analysis task. The global average also obtains an important increase in the results, with a relative enhancement of 18.8% with respect to the state of the art, thus supporting the idea of our proposal.

5. QUALITATIVE EVALUATION

To finalize the analysis of the results, we now shall discuss a qualitative evaluation for different scenarios. Table 4 gathers some selected examples.

As regards the first case, in which the document analysis carries out the extraction of notes in the SALZINNES → CAPITAN scenario, i.e. SALZINNES as source and CAPITAN as target, we observe that the SAE-based framework does not detect most of the elements. Indeed, it confuses parts of staff lines with notes, making the result not even close to what was expected. Oppositely, our method does differentiate the staff from the notes, obtaining a much more reliable result. Note that the detection still fails in many cases, particularly in those in which the sym-

Table 4: Selected examples for notes, text, and staff recognition cases. The table compares the extraction of the layer for the SAE-based framework and our approach. Input images and their corresponding ground-truth data are also provided.

	SALZINNES → CAPITAN (notes)	CAPITAN → SALZINNES (text)
Input		
SAE-based		
Our approach		
Ground truth		

	SALZINNES → CAPITAN (staff)	EINSIEDELN → CAPITAN (text)
Input		
SAE-based		
Our approach		
Ground truth		

bols are hollow. We attribute this issue to the fact that the source images do not contain hollow symbols since they contain *Neumatic* notation, so that the ground truth provided for the training does not include elements with common features to those involved in the hollow symbols presented in the *Mensural* notation. However, in spite of the clear differences between the symbols of both domains, the filled symbols are quite well recognized since SALZINNES also contains ink-filled symbols, but squared, obtaining, in general, a better recognition to that provided by the state of the art. These graphic differences can be seen in Figure 3.

Concerning the second case, we analyze an example of the detection of text when *Mensural* notation is used as source—CAPITAN—and the target manuscript is written in *Neumatic* notation—SALZINNES. As seen, the state of the art does not detect almost any pixel associated with this layer. Although the detection performed by our approach is not perfect, it does obtain a result that is much closer to the expected one. This could be used to detect the regions in which the texts are located to be individually processed by other mechanisms, for instance, Optical Character Recognition techniques.

Another example is the case in which staff lines must be identified. In this example—SALZINNES → CAPITAN—we observe that the staff-line retrieval is not properly performed by the SAE-based framework. Several parts of the staff are detected, but it would be quite difficult to reconstruct the lines due to the poor quality of the prediction. Conversely, our method can solve this issue by clearly improving the staff-line detection and obtaining a result closer to the ground truth. As shown in Table 3, the state-of-the-art framework achieves competitive average figures in terms of ps-F₁, however, note that this is a selected example in which the state of the art does not provide good results to visually analyze the capabilities of our method.

The last example in Table 4 provides another case of

text recognition. In this scenario, the notation types of the *S* and *T* manuscripts are reversed with respect to the second example. Specifically, we use a *Neumatic* manuscript—EINSIEDELN—as source and the *Mensural* one—CAPITAN—as target. We observe that the state of the art recovers the text with various errors, losing part of the text and also confusing the background with text. The DA method improves this result by providing a much more accurate layered version. Note that, despite the great differences between the text of both domains, the method is able to adapt by searching for domain-invariant features in order to recognize these elements.

6. CONCLUSIONS

This work presents an unsupervised DA framework for document analysis of music score images, which builds upon existing approaches and the so-called Gradient Reversal Layer. The idea is based on learning domain-invariant features that allow transferring knowledge from a labeled source domain to an unlabeled target one.

Our experiments reveal that the approach is generally beneficial for the task at issue. The actual improvement depends on the layer of information considered. Substantial benefits are reported for the layers that depict particular features at each domain (such as text), whereas layers that depict common features (such as staff lines) are barely affected by the adaptation process. In addition, we observed that the source domain is indeed relevant to make the DA method successful. In the best case, our approach substantially increases the average performance up to 44% of relative improvement.

In light of these results, future work will focus on evaluating the method with more types of historical manuscripts, as well as studying the applicability of other DA techniques for document analysis, such as those based on Generative Adversarial Networks.

7. ACKNOWLEDGMENT

This work was supported by the University of Alicante project GRE19-04. The first author also acknowledges support from the “Programa I+D+i de la Generalitat Valenciana” through grant ACIF/2019/042.

8. REFERENCES

- [1] D. Bainbridge and T. Bell, “The challenge of optical music recognition,” *Computers and the Humanities*, vol. 35, no. 2, pp. 95–121, 2001.
- [2] J. Calvo-Zaragoza, F. J. Castellanos, G. Vigliensoni, and I. Fujinaga, “Deep neural networks for document processing of music score images,” *Applied Sciences*, vol. 8, no. 5, p. 654, 2018.
- [3] I. Fujinaga and G. Vigliensoni, “The art of teaching computers: The SIMSSA optical music recognition workflow system,” in *27th European Signal Processing Conference, EUSIPCO, A Coruña, Spain, September 2-6*. IEEE, 2019, pp. 1–5.
- [4] J. Calvo-Zaragoza, J. H. Jr., and A. Pacha, “Understanding optical music recognition,” *ACM Comput. Surv.*, vol. 53, no. 4, Jul. 2020.
- [5] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” in *International conference on machine learning*. PMLR, 2015, pp. 1180–1189.
- [6] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. S. Marçal, C. Guedes, and J. S. Cardoso, “Optical music recognition: State-of-the-art and open issues,” *International Journal of Multimedia Information Retrieval*, vol. 1, no. 3, pp. 173–190, 2012.
- [7] J. Sauvola and M. Pietikäinen, “Adaptive document image binarization,” *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.
- [8] B. Gatos, I. Pratikakis, and S. J. Perantonis, “Adaptive degraded document image binarization,” *Pattern Recognition*, vol. 39, no. 3, pp. 317–327, 2006.
- [9] N. R. Howe, “Document binarization with automatic parameter tuning,” *International Journal on Document Analysis and Recognition*, vol. 16, no. 3, pp. 247–258, 2013.
- [10] T. Pinto, A. Rebelo, G. A. Giraldo, and J. S. Cardoso, “Music score binarization based on domain knowledge,” in *5th Iberian Conference on Pattern Recognition and Image Analysis, Las Palmas de Gran Canaria, Spain*, 2011, pp. 700–708.
- [11] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee, “An MRF model for binarization of music scores with complex background,” *Pattern Recognition Letters*, vol. 69, no. Supplement C, pp. 88–95, 2016.
- [12] J. A. Burgoyne and I. Fujinaga, “Lyric extraction and recognition on digital images of early music sources,” in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR, Kobe, Japan, October 26-30, 2009*, pp. 723–728.
- [13] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga, “A comparative study of staff removal algorithms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 753–766, 2008.
- [14] J. Dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. Pinto da Costa, “Staff detection with stable paths,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, pp. 1134–1139, 2009.
- [15] T. Géraud, “A morphological method for music score staff removal,” in *International Conference on Image Processing*, 2014, pp. 2599–2603.
- [16] A. Konwer, A. K. Bhunia, A. Bhowmick, A. K. Bhunia, P. Banerjee, P. P. Roy, and U. Pal, “Staff line removal using generative adversarial networks,” in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 1103–1108.
- [17] F. J. Castellanos, J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga, “Document analysis of music score images with selectional auto-encoders,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR, Paris, France, September 23-27, 2018*, pp. 256–263.
- [18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.
- [19] F. J. Castellanos, A.-J. Gallego, and J. Calvo-Zaragoza, “Unsupervised neural domain adaptation for document image binarization,” *Pattern Recognition*, p. 108099, 2021.
- [20] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, “Handwritten music recognition for mensural notation: Formulation, data and baseline results,” in *14th IAPR International Conference on Document Analysis and Recognition, ICDAR, Kyoto, Japan, November 9-15, 2017*, pp. 1081–1086.
- [21] A. Özgür, L. Özgür, and T. Güngör, “Text categorization with class-based and corpus-based keyword selection,” in *International Symposium on Computer and Information Sciences*, 2005, pp. 606–615.
- [22] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of the 19th International Conference on Computational Statistics, COMPSTAT, Paris, France, August 22-27*. Springer, 2010, pp. 177–186.

CODIFIED AUDIO LANGUAGE MODELING LEARNS USEFUL REPRESENTATIONS FOR MUSIC INFORMATION RETRIEVAL

Rodrigo Castellon*

Chris Donahue*

Percy Liang

Stanford University

{rjcaste, cdonahue, pliang}@cs.stanford.edu

ABSTRACT

We demonstrate that language models pre-trained on codified (discretely-encoded) music audio learn representations that are useful for downstream MIR tasks. Specifically, we explore representations from Jukebox [1]: a music generation system containing a language model trained on codified audio from 1M songs. To determine if Jukebox’s representations contain useful information for MIR, we use them as input features to train shallow models on several MIR tasks. Relative to representations from conventional MIR models which are pre-trained on tagging, we find that using representations from Jukebox as input features yields 30% stronger performance on average across four MIR tasks: tagging, genre classification, key detection, and emotion recognition. For key detection, we observe that representations from Jukebox are considerably stronger than those from models pre-trained on tagging, suggesting that pre-training via codified audio language modeling may address blind spots in conventional approaches. We interpret the strength of Jukebox’s representations as evidence that modeling audio instead of tags provides richer representations for MIR.

1. INTRODUCTION

It is conventional in MIR¹ to *pre-train* models on large labeled datasets for one or more tasks (commonly tagging), and reuse the learned representations for different *downstream* tasks [2–10]. Such *transfer learning* approaches decrease the amount of labeled data needed to perform well on downstream tasks, which is particularly useful in MIR where labeled data for many important tasks is scarce [11, 12]. Historically-speaking, improvement on downstream tasks is enabled by finding ever-larger sources of labels for pre-training—in chronological order: tags [3], metadata [5, 7, 9, 10], and recently, co-listening data [9]. However, it stands to reason that modeling music *audio*

* Equal contribution.

¹ MIR has a broad definition, but in this paper “MIR” refers specifically to making discriminative predictions on music audio.


(as opposed to labels) could yield richer representations. Recently, contrastive learning [13] has been proposed as a pre-training strategy to learn representations from audio for MIR [14], but this approach has yet to exceed the performance of label-based pre-training on downstream tasks.

Outside of the discriminative MIR landscape, a recent system called Jukebox [1] demonstrated promising performance for generating music audio. To achieve this result, Jukebox leverages recent architectural developments from natural language processing (NLP) by *codifying* audio—encoding high-rate continuous audio waveforms into lower-rate discrete sequences which can be fed in directly to NLP models. Specifically, Jukebox trains a Transformer [15, 16] *language model*, an autoregressive generative model, on codified audio from 1M songs. Purely for convenience, we refer to Jukebox’s training procedure as *codified audio language modeling* (CALM).

While Jukebox already demonstrates that CALM is useful for music *generation*, in this work we demonstrate that CALM is also useful as a pre-training procedure for *discriminative* MIR tasks. To this end, we repurpose Jukebox for MIR by first using it to extract audio feature representations, and then training shallow models (*probes* [18, 19]) on downstream tasks using these features as input (Figure 1). Relative to representations from models pre-trained with tagging, we find that representations from Jukebox are 30% more effective on average when used to train probes on four downstream MIR tasks: tagging, genre classification, key detection, and emotion recognition. We also observe that representations from Jukebox are much more useful for key detection than those from models pre-trained on tagging, which suggests that CALM pre-training may be particularly beneficial for tasks which have little to do with tagging. This simple setup of training shallow models on representations from Jukebox is even competitive with purpose-built state-of-the-art methods on several tasks.

To facilitate reproducibility [11], we release all of our code for this project, alongside images for Docker containers which provide full provenance for our experiments.² We note that, while CALM pre-training at the scale of Jukebox requires substantial computational resources, our post hoc experiments with Jukebox only require a single commodity GPU with 12 GB memory.

² Code: <https://github.com/p-lambda/jukemir>
Containers: <https://hub.docker.com/orgs/jukemir>
All experiments reproducible on the CodaLab platform:
<https://worksheets.codalab.org/worksheets/0x7c5afa6f88bd4ff29fec75035332a583>

 © Rodrigo Castellon, Chris Donahue, Percy Liang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Rodrigo Castellon, Chris Donahue, Percy Liang, “Codified audio language modeling learns useful representations for music information retrieval”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

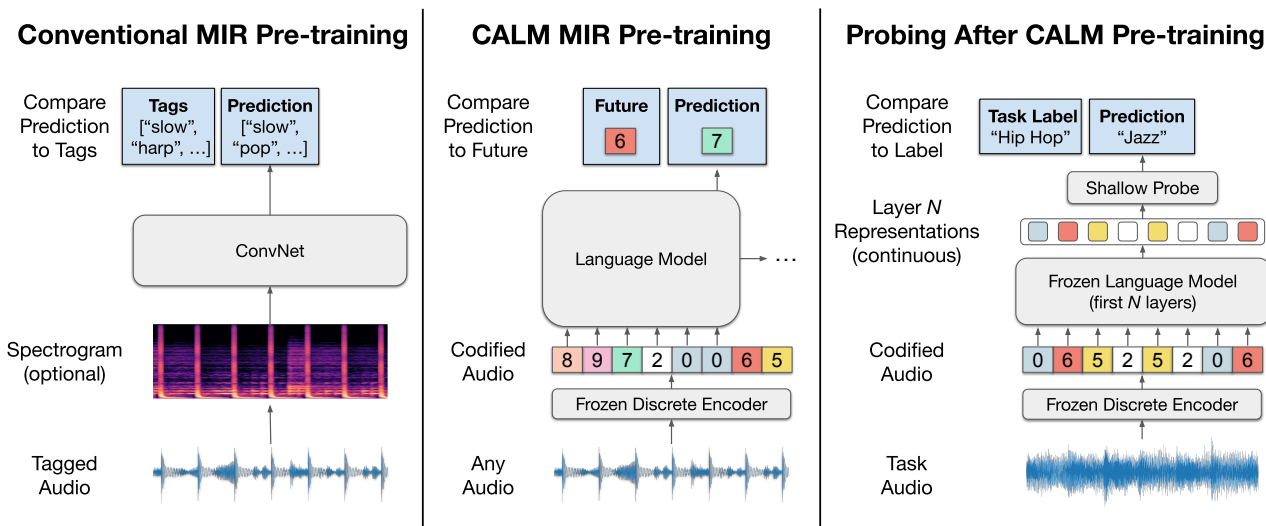


Figure 1. Conventional MIR pre-training (left) trains convolutional neural networks on audio spectrograms using manually-annotated labels from tagging datasets. In contrast, CALM MIR pre-training (middle) involves training a language model on codified audio, which has been previously explored for music generation [17, 1]—here, we propose to use it for discriminative MIR tasks. To determine if CALM pre-training is effective for MIR, we probe for information about particular MIR tasks (right) in resultant representations. Specifically, we extract features from the learned language model for the audio in small, task-specific labeled datasets, and use these features to train shallow probing models on each task.

2. CALM PRE-TRAINING

CALM was first proposed by van den Oord et al. and used for unconditional speech generation [20]. As input, CALM takes a collection of raw audio waveforms (and optionally, conditioning metadata), and learns a distribution $p(\text{audio} \mid \text{metadata})$. To this end, CALM adopts a three-stage approach: (1) *codify* a high-rate continuous audio signal into lower-rate discrete codes, (2) train a *language model* on the resulting codified audio and optional metadata, i.e., learn $p(\text{codified audio} \mid \text{metadata})$, and (3) decode sequences generated by the language model to raw audio.³ The original paper [20] also proposed a strategy for codifying audio called the vector-quantized variational auto-encoder (VQ-VAE), and the language model was a WaveNet [21]. Within music, CALM was first used by Dieleman et al. for unconditional piano music generation [17], and subsequently, Dhariwal et al. used CALM to build a music generation system called Jukebox [1] with conditioning on genre, artist, and optionally, lyrics.

Despite promising results on music audio generation, CALM has not yet been explored as a pre-training strategy for discriminative MIR. We suspect that effective music audio generation necessitates intermediate representations that would also contain useful information for MIR. This hypothesis is further motivated by an abundance of previous work in NLP suggesting that generative and self-supervised pre-training can yield powerful representations for discriminative tasks [22–25].

To explore this potential, we repurpose Jukebox for MIR. While Jukebox was designed only for generation, its internal language model was trained on codified audio from a corpus of 1.2M songs from many genres and

artists, making its representations potentially suitable for a multitude of downstream MIR tasks. Jukebox consists of two components—the first is a small (2M parameters) VQ-VAE model [20] that learns to codify high-rate (44.1 kHz), continuous audio waveforms into lower-rate (~345 Hz), discrete code sequences with a vocabulary size of 2048 (11 bits). The second component is a large (5B parameters) language model that learns to generate codified audio using a Transformer decoder—an architecture originally designed for modeling natural language [15, 16]. By training on codified audio (as in [17, 1]) instead of raw audio (as in [21, 16]), language models are (empirically) able to learn longer-term structure in music, while simultaneously using less memory to model the same amount of audio.

Like conventional MIR models which pre-train on tagging and/or metadata, Jukebox also makes use of genre and artist labels during training, providing them as conditioning information to allow for increased user control for generation. Hence, while CALM in general is an unsupervised strategy that does not require labels, transfer learning from Jukebox specifically should *not* be considered an unsupervised approach (especially for downstream tasks like genre detection). However, by modeling the *audio* itself instead of modeling the *labels* (as in conventional MIR pre-training), we hypothesize that Jukebox learns richer representations for MIR tasks than conventional strategies.

3. EXTRACTING SUITABLE REPRESENTATIONS FROM JUKEBOX

Here we describe how we extract audio representations from Jukebox which are suitable as input features for training shallow models. While several pre-trained Jukebox models exist with different sizes and conditioning in-

³ This third stage is not necessary for transfer learning.

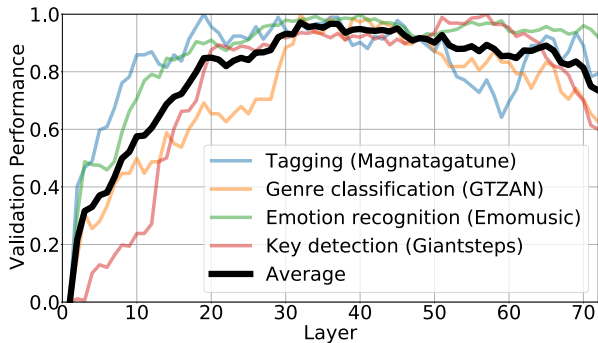


Figure 2. Normalized validation performance of linear models trained on representations from specific layers of Jukebox across four downstream MIR tasks. On average, the strongest representations for these tasks come from the middle of Jukebox.

formation, here we use the 5B-parameter model without lyrics conditioning (named “5b”), which is a sparse transformer [15, 16] containing 72 layers. Each layer yields 4800-dimensional activations for each element in the codified audio sequence, i.e., approximately 345 times per second. To extract representations from this model for a particular audio waveform, we (1) resample the waveform to 44.1kHz, (2) normalize it, (3) codify it using the Jukebox VQ-VAE model, and (4) input the codified audio into the language model, interpreting its layer-wise activations as representations. Jukebox was trained on ~ 24 -second audio clips (codified audio sequences of length 8192)—we feed in this same amount of audio at a time when extracting representations. In addition to the genre and artist conditioning fields mentioned previously, Jukebox expects two additional fields: total song length and clip offset—to ensure that representations only depend on the input audio, we simply pass in “unknown” for artist and genre, one minute for song length, and zero seconds for clip offset.⁴

The Jukebox language model yields an unwieldy amount of data—for every 24-second audio clip, it emits $24 \times 345 \times 72 \times 4800$ numbers, i.e., over 10GB if stored naively as 32-bit floating point. We reduce the amount of data by mean pooling across time, a common strategy in MIR transfer learning [4, 8], which aggregates more than 10GB of activations to around 1MB (72×4800).

3.1 Layer selection

While pooling across time dramatically reduced the dimensionality of Jukebox’s outputs, training shallow classifiers on 72×4800 features is still computationally expensive. To further reduce the dimensionality, we use only one of the layers from Jukebox—the middle layer (36)—yielding a total of 4800 features per 24 second audio clip. Unlike conventional pre-training, where the strongest representations for transfer learning typically lie at the end of the network [26], the strongest representations from pre-trained

⁴ We observed in initial experiments that passing in ground-truth conditioning information had little effect on downstream performance. Hence, we elected to pass in placeholder metadata to maintain the typical type signature for audio feature extraction (audio as the only input).

Task	Size	Metrics	#Out
Tagging [31]	25860	AUC/AP	50
Genre classification [32]	930	Accuracy	10
Key detection [33]	1763	Score	24
Emotion recognition [34]	744	A/V R^2	2

Table 1. Basic information about the four tasks we consider in this work, including the size of each task-specific dataset in terms of number of labeled examples, relevant metrics for each task, and the number of model outputs required for each dataset.

language models tend to lie towards the middle [27–30]. To confirm this observation in our context, we trained linear models using representations from different layers of Jukebox on our downstream MIR tasks—average performance indeed peaked at the middle layers (Figure 2).

In addition to using the middle layer, we experimented with two other layer selection strategies: (1) sub-sampling layers across the network, and (2) selecting relevant layers in a task-specific fashion.⁵ We found that the simplest strategy of using only the middle layer was equally effective and more computationally practical⁶ than the other two layer selection strategies.

4. DOWNSTREAM TASK DESCRIPTIONS

We select four downstream MIR tasks to constitute a benchmark for comparing different audio feature representations: (1) tagging, (2) genre classification, (3) key detection, and (4) emotion recognition. A summary of the datasets used for each task appears in Table 1. These tasks were selected to cover a wide range of dataset sizes (744 examples for emotion recognition vs. 26k examples for tagging) and subjectivity (emotion recognition is more subjective vs. key detection is more objective). Additionally, each task has a public dataset with standard evaluation metrics. We describe each of these tasks below.

4.1 Tagging

Tagging involves determining which tags from a fixed set of tags apply to a particular song. Categories of tags include genre (e.g., jazz), instrumentation (e.g., violin), emotions (e.g., happy), and characteristics (e.g., fast). There are two large datasets for tagging, which both contain human-annotated tags for 30-second clips: MagnaTagATune [31] (MTT) which contains around 26k clips, and a tagged subset of 240k clips from the Million Song Dataset [35] (MSD). While both datasets contain a large vocabulary of tags, typical usage involves limiting the vocabulary to the 50 most common tags in each.

Because it is the largest non-proprietary MIR dataset, MSD is commonly used for pre-training models for transfer learning. To mitigate an unfair advantage of methods

⁵ This procedure selected layers that were the most jointly informative in a greedy fashion, measured by task performance with a linear probe.

⁶ While the entirety of Jukebox does *not* fit on a single commodity GPU with 12GB memory, the first 36 layers *do* fit.

which pre-train on MSD, we use MTT instead of MSD to benchmark representations on tagging performance. While both datasets are superficially similar (choosing from 50 tags for 30-second clips), their label distributions are quite different: MSD is skewed towards genre tags, while MTT is skewed towards instrumentation tags.

We use the standard (12:1:3) train, validation, and test split for MTT [3]. Additionally, we report both common metrics (both are macro-averaged over tags): area under the receiver operating characteristic curve (MTT_{AUC}), and average precision (MTT_{AP}).⁷ Note that inconsistencies in handling unlabeled examples for past work on MTT have been observed [36]—some work discards examples without top-50 tags during training, evaluation, both, or neither. In this work, we do not discard any examples.

4.2 Genre classification

Genre classification involves assigning the most appropriate genre from a fixed list for a given song. For this task, we report accuracy on the GTZAN dataset [37], which contains 30-second clips from 10 distinct genres. We adopt the “fault-filtered” split from [32] which addresses some of the reported issues with this dataset [38]. We note that this task has a high degree of overlap with tagging, as tagging datasets typically have a number of genres within their tag vocabulary. In fact, seven of ten genres in GTZAN are present in the tag list of MSD.

4.3 Key detection

Key detection involves predicting both the scale and tonic pitch class for the underlying key of a song. We investigate the Giantsteps-MTG and Giantsteps datasets [33] which include songs in major and minor scales for all pitch classes, i.e., a 24-way classification task. As in past work [39], we use the former for training and the latter for testing. Because no standard validation split exists for Giantsteps-MTG, we follow [32] and create an artist-stratified 4:1 split for training and validation, which we include in our codebase for reproducibility. The music in this dataset is all electronic dance music, and the clips are two minutes in length. We report the typical weighted score metric for Giantsteps (GS): an accuracy measure which gives partial credit for reasonable mistakes such as predicting the relative minor key for the major ground truth [40].

4.4 Emotion recognition

Emotion recognition involves predicting human emotional response to a song. Data is collected by asking humans to report their emotional response on a two dimensional valence-arousal plane [41], where valence indicates positive versus negative emotional response, and arousal indicates emotional intensity. We use the Emomusic dataset [34], which contains 744 clips of 45 seconds in length. We investigate the static version of this task

⁷ Most past work refers to the quantity of average precision as area under the precision-recall curve.

Representation	Pre-training strategy	Dimensions
CHROMA	N/A	72
MFCC	N/A	120
CHOI [4]	MSD Tagging [3]	160
MUSICNN [8]	MSD Tagging [3]	4194
CLMR [14]	Contrastive [13]	512
JUKEBOX [1]	CALM [20]	4800

Table 2. Basic statistics about the six representations we examine in this work.

where original time-varying annotations are averaged together to constitute a clip-level annotation. Because this dataset does not have a standard split, it is difficult to directly compare with past work. To simplify comparison going forward, we created an artist-stratified split of Emomusic, which is released in our codebase. We take the highest reported numbers from past work to characterize “state-of-the-art” performance, though we note that these numbers are not directly comparable to our own due to differing splits. We report the coefficient of determination between the model predictions and human annotations for arousal (Emo_A) and valence (Emo_V).

5. PROBING EXPERIMENTS

Here we describe our protocol for probing for information about MIR tasks in representations from Jukebox and other pre-trained models, i.e., measuring performance of shallow models trained on these tasks using different representations as input features. We borrow the term “probing” from analogous investigations in NLP [19,42,43], however such methodology is common in MIR [2–5,7–10].

5.1 Descriptions of representations

In addition to probing representations from Jukebox (an exemplar of CALM pre-training), we probe four additional representations which are emblematic of three other MIR pre-training strategies (Table 2). Before pre-training, hand-crafted features were commonplace in MIR—as archetypal examples, we probe constant-Q chromagrams (CHROMA) and Mel-frequency cepstral coefficients (MFCC), extracted with librosa [49] using the default settings. As in [4], we concatenate the mean and standard deviation across time of both the features and their first- and second-order discrete differences. We also probe two examples of the current conventional paradigm which pre-trains on tagging using MSD: a convolutional model proposed by Choi et al. [4] (CHOI), and a more modern convolutional model from [8] (MUSICNN). Finally, we compare to a recent MIR pre-training strategy called *contrastive learning of musical representations* [14] (CLMR), though we note that CLMR was trained on far less audio (a few thousand songs) than the other pre-trained models.

All of these strategies operate at different frame rates, i.e., they produce a different number of representation vectors for a fixed amount of input audio. To handle

Approach	Tags		Genre	Key	Emotion		Average
	MTT _{AUC}	MTT _{AP}	GTZAN	GS	Emo _A	Emo _V	
(No pre-training) Probing CHROMA	77.6	18.5	32.8	56.5	29.3	5.9	38.7
(No pre-training) Probing MFCC	85.8	30.2	44.8	14.6	47.9	26.5	38.7
(Tagging) Probing CHOI [4]	89.7	36.4	75.9	13.1	67.3	43.4	51.9
(Tagging) Probing MUSICNN [8]	90.6	38.3	79.0	12.8	70.3	46.6	53.7
(Contrastive) Probing CLMR [14]	89.4	36.1	68.6	14.9	67.8	45.8	50.8
(CALM) Probing JUKEBOX [1]	91.5	41.4	79.7	66.7	72.1	61.7	69.9
State-of-the-art [9, 8, 6, 44–46]	92.0	38.4	82.1	79.6	70.4*	55.6*	72.5*
Pre-trained [9, 14, 6, 45, 45, 47]	92.0	35.9	82.1	75.8	67.1*	55.6*	70.8*
From scratch [8, 8, 48, 44, 44, 39]	90.7	38.4	65.8	74.3	70.4*	50.0*	66.2*

Table 3. Comparing performance of probes on representations from a model pre-trained with CALM to other pre-trained MIR models (top section) to reported state-of-the-art performance (bottom section) across four tasks: (1) tagging (MTT_{AUC}/MTT_{AP}), (2) genre classification (GTZAN), (3) key detection (GS), and (4) emotion recognition (Emo_A/Emo_V). For all six metrics, the max score is 100 and higher is better—see Section 4 for a full description of tasks/metrics. For each metric, the best probing-based approach and the best approach overall are **bolded**. We also report an average score across all four tasks; tasks with multiple evaluation metrics are averaged beforehand. On all metrics, probing JUKEBOX is more effective than probing representations from other pre-trained models. Probing JUKEBOX is competitive with task-specific state-of-the-art approaches for all tasks/metrics except key detection (GS). Note that the ordering of citations in the bottom section corresponds to respective column ordering. * indicates that past work on Emomusic evaluates on different subsets of the dataset than our work and hence numbers are not directly comparable—see Section 4.4 for details.

this, we follow common practice of mean pooling representations across time [4, 8]. While CHROMA, MFCC, and CLMR produce a single canonical representation per frame, we note that the other three produce multiple representations per frame, i.e., the activations of individual network layers. For CHOI, we concatenate all layer activations together, which was shown to have strong performance on all downstream tasks in [4]. For MUSICNN, we concatenate together the mean and max pool of three-second windows (before mean pooling across these windows), i.e., the default configuration for that approach. For JUKEBOX, we use the middle layer activations as motivated in Section 3.1. By using a single layer, we mitigate the potential of a superficial dimensionality advantage for JUKEBOX, as this induces a dimensionality similar to that of MUSICNN (4800 and 4194 respectively; see Table 2).

Unlike other representations which operate on short audio windows, CHOI and JUKEBOX were trained on long windows (29 seconds and 24 seconds respectively). Accordingly, for the three datasets with short clips (tagging, genre classification, and emotion recognition all have clips between 30 and 45 seconds in length), we adopt the policy from [4] and simply truncate the clips to the first window when computing representations for CHOI and JUKEBOX. Because clips from the key detection dataset are longer (two minutes), we split the clips into 30-second windows for all methods and train probes on these shorter windows. At test time, we ensemble window-level predictions into clip-level predictions before computing the score.

5.2 Probing protocol

To probe representations for relevant information about downstream MIR tasks, we train shallow supervised mod-

els (linear models and one-layer MLPs) on each task using these representations as input features. As some representations may require different hyperparameter configurations for successful training, we run a grid search over the following hyperparameters (216 total configurations) for each representation and task (24 total grid searches), using early stopping based on task-specific metrics computed on the validation set of each task:

- Feature standardization: {off, on}
- Model: {Linear, one-layer MLP with 512 hidden units}
- Batch size: {64, 256}
- Learning rate: {1e-5, 1e-4, 1e-3}
- Dropout probability: {0.25, 0.5, 0.75}
- L2 regularization: {0, 1e-4, 1e-3}

While we use this same hyperparameter grid for all tasks, the learning objective varies by task (cross-entropy for genre classification and key detection, independent binary cross-entropy per tag for tagging, and mean squared error for emotion recognition) as does the number of probe outputs (Table 1). Some tasks have multiple metrics—we early stop on MTT_{AUC} for tagging as it is a more common metric than MTT_{AP}, and on the average of Emo_A and Emo_V for emotion recognition. We take the model with the best early stopping performance from each grid search and compute its performance on the task-specific test set.

6. RESULTS AND DISCUSSION

In Table 3, we report performance of all representations on all tasks and metrics, as well as average performance across all tasks. Results are indicative that CALM is a promising paradigm for MIR pre-training. Specifically, we observe that probing the representations from JUKEBOX

(learned through CALM pre-training) achieves an average of 69.9, which is 30% higher relative to the average of the best representation pre-trained with tagging (MUSICNN achieves an average of 53.7). Performance of JUKEBOX on all individual metrics is also higher than that of any other representation. Representations from all pre-trained models outperform hand-crafted features (CHROMA and MFCC) on average. Note that these results are holistic comparisons across different model architectures, model sizes, and amounts of pre-training data (e.g., CLMR was trained on far less data than JUKEBOX), and hence not sufficient evidence to claim that CALM is the “best” music pre-training strategy in general.

We also observe that JUKEBOX contains substantially more information relevant for key detection than other representations. While CHROMA (spectrogram projected onto musical pitch classes) contains information relevant to key detection by design, all other representations besides JUKEBOX yield performance on par with that of a majority classifier (outputting “F minor” for every example scores 15.0)—hence, these representations contain almost no information about this task. For models pre-trained with tagging (CHOI and MUSICNN), intuition suggests that this is because none of the tags in MSD relate to key signature. For CLMR, we speculate that the use of transposition as a data augmentation strategy removes information about key signature. While tagging and CLMR were not designed with the intention of supporting transfer to key detection, we argue that it is generally desirable to have a unified music representation which performs well on many downstream MIR tasks. Hence, we interpret the comparatively stronger performance of JUKEBOX on key detection as evidence that CALM pre-training addresses blind spots present in other MIR pre-training paradigms.

In the bottom section of Table 3, we also report state-of-the-art performance for purpose-built methods on all tasks, which is further broken down by models which use any form of pre-training (including pre-training on additional task-specific data as in [47]) vs. ones that are trained from scratch. Surprisingly, we observe that probing JUKEBOX is competitive with state-of-the-art for all tasks except for key detection, and achieves an average only 4% lower relative to that of state-of-the-art. On tagging, probing JUKEBOX achieves similar MTT_{AUC} to a strategy which pre-trains on a proprietary dataset of 10M songs using supervision [9]. We interpret the strong performance of this simple probing setup as evidence that CALM pre-training is a promising path towards models that are useful for many MIR tasks.

We believe that CALM pre-training is promising for MIR not just because of the strong performance of an existing model (Jukebox), but also because there are numerous avenues which may yield further improvements. Firstly, CALM could be scaled up to pre-train even larger models on more data (Jukebox was trained on 1M songs, while Spotify has an estimated 70M songs in its catalog). In [50], it is observed that increasing model and dataset size yields predictable improvements to cross-entropy for language modeling in NLP, an insight which may also

hold for CALM pre-training for MIR. Secondly, we anticipate that fine-tuning a model pre-trained with CALM would outperform our probing setup. Finally, taking a cue from related findings in NLP, we speculate that CALM pre-training with a bidirectional model and masked language modeling (as in BERT [23]) would outperform the generative setup of Jukebox (that of GPT [51]).

7. RELATED WORK

Transfer learning has been an active area of study in MIR for over a decade [52, 53, 2, 54]. The predominant strategy for MIR pre-training using large tagging datasets was first proposed by van den Oord et al. 2014 [3]. This work pre-trained deep neural networks on MSD and demonstrated promising performance on other tagging and genre classification tasks. Choi et al. 2017 [4] pre-trained a convolutional neural network on MSD and also explored a more diverse array of downstream tasks—we use their pre-trained model as a baseline. More recent improvements use the same approach with different architectures [6, 8].

Other strategies for MIR transfer learning have been proposed. Some work pre-trains on music metadata (e.g., artist, album) instead of tags [5, 7]. In contrast to the manual annotations required for tagging-based pre-training, metadata is much cheaper to obtain, but performance of pre-training on metadata is comparable to that of pre-training on tagging. Kim et al. 2020 [10] improve over Choi et al. 2017 [4] using a multi-task approach. Huang et al. [9] demonstrate that metadata can be combined with proprietary co-listening data for pre-training on 10M songs to achieve state-of-the-art performance on MTT—probing representations from CALM pre-training on 1M songs achieves comparable performance (Table 3). Finally, contrastive learning [13] has been proposed for MIR pre-training [55, 56, 14]—we compare to [14].

While CALM has not previously been explored for MIR transfer learning, it has been explored for other purposes. van den Oord et al. 2017 [20] first proposed CALM and used it for unconditional speech generation. Variations of CALM have been explored for speech recognition [57, 58] and urban sound classification [59]. CALM has also been explored for music generation [17, 1]. Language models have been used extensively for symbolic music [60–62], including work on pre-training [63, 64].

8. CONCLUSION

In this work we demonstrated that CALM is a promising pre-training strategy for MIR. Compared to conventional approaches, CALM learns richer representations by modeling audio instead of labels. Moreover, CALM allows MIR researchers to repurpose NLP methodology—historically, repurposing methodology from another field (computer vision) has provided considerable leverage for MIR. Finally, CALM suggests a direction for MIR research where enormous models pre-trained on large music catalogs break new ground on MIR tasks, analogous to ongoing paradigm shifts in other areas of machine learning.

9. ACKNOWLEDGEMENTS

We would like to thank Nelson Liu, Mina Lee, John Hewitt, Janne Spijkervet, Minz Won, Jordi Pons, Ethan Chi, Michael Xie, Ananya Kumar, and Glen Husman for helpful conversations about this work. We also thank all reviewers for their helpful feedback.

10. REFERENCES

- [1] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv:2005.00341*, 2020.
- [2] P. Hamel, M. Davies, K. Yoshii, and M. Goto, "Transfer learning in MIR: Sharing learned latent representations for music audio classification and similarity," in *ISMIR*, 2013.
- [3] A. van den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *ISMIR*, 2014.
- [4] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *ISMIR*, 2017.
- [5] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," *arXiv:1710.06648*, 2017.
- [6] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, 2018.
- [7] J. Lee, J. Park, and J. Nam, "Representation learning of music using artist, album, and track information," *arXiv:1906.11783*, 2019.
- [8] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," *ISMIR Late-breaking Demos*, 2019.
- [9] Q. Huang, A. Jansen, L. Zhang, D. P. Ellis, R. A. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *ICASSP*, 2020.
- [10] J. Kim, J. Urbano, C. C. Liem, and A. Hanjalic, "One deep music representation to rule them all? A comparative analysis of different representation learning strategies," *Neural Computing and Applications*, 2020.
- [11] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, "Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research," *IEEE Signal Processing Magazine*, 2018.
- [12] W. Chen, J. Keast, J. Moody, C. Moriarty, F. Villalobos, V. Winter, X. Zhang, X. Lyu, E. Freeman, J. Wang *et al.*, "Data usage in MIR: history & future recommendations," in *ISMIR*, 2019.
- [13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020.
- [14] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *arXiv:2103.09410*, 2021.
- [15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv:1706.03762*, 2017.
- [16] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv:1904.10509*, 2019.
- [17] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: modelling raw audio at scale," in *NIPS*, 2018.
- [18] G. Alain and Y. Bengio, "Understanding intermediate layers using linear classifier probes," *arXiv:1610.01644*, 2016.
- [19] D. Hupkes, S. Veldhoen, and W. Zuidema, "Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure," *Journal of Artificial Intelligence Research*, 2018.
- [20] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," *arXiv:1711.00937*, 2017.
- [21] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv:1609.03499*, 2016.
- [22] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.
- [23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.
- [24] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.
- [25] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT understands, too," *arXiv:2103.10385*, 2021.
- [26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, 2014.

- [27] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith, “Linguistic knowledge and transferability of contextual representations,” *arXiv:1903.08855*, 2019.
- [28] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, “Generative pretraining from pixels,” in *ICML*, 2020.
- [29] E. A. Chi, J. Hewitt, and C. D. Manning, “Finding universal grammatical relations in multilingual bert,” in *Association for Computational Linguistics*, 2020.
- [30] A. Rogers, O. Kovaleva, and A. Rumshisky, “A primer in BERTology: What we know about how BERT works,” *Transactions of the Association for Computational Linguistics*, 2020.
- [31] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, “Evaluation of algorithms using games: The case of music tagging,” in *ISMIR*, 2009.
- [32] C. Kereliuk, B. L. Sturm, and J. Larsen, “Deep learning and music adversaries,” *IEEE Transactions on Multimedia*, 2015.
- [33] P. Knees, Á. Faraldo Pérez, H. Boyer, R. Vogl, S. Böck, F. Hörschläger, M. Le Goff *et al.*, “Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections,” in *ISMIR*, 2015.
- [34] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, “1000 songs for emotional analysis of music,” in *ACM International Workshop on Crowdsourcing for Multimedia*, 2013.
- [35] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The Million Song Dataset,” in *ISMIR*, 2011.
- [36] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, “Evaluation of CNN-based automatic music tagging models,” *arXiv:2006.00751*, 2020.
- [37] G. Tzanetakis and P. Cook, “Musical genre classification of audio signals,” *IEEE Transactions on Speech and Audio Processing*, 2002.
- [38] B. L. Sturm, “The GTZAN dataset: its contents, its faults, their effects on evaluation, and its future use,” *arXiv:1306.1461*, 2013.
- [39] F. Korzeniowski and G. Widmer, “End-to-end musical key estimation using a convolutional neural network,” in *European Signal Processing Conference*, 2017.
- [40] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, “mir_eval: A transparent implementation of common mir metrics,” in *ISMIR*, 2014.
- [41] A. Huq, J. P. Bello, and R. Rowe, “Automated music emotion recognition: A systematic evaluation,” *Journal of New Music Research*, 2010.
- [42] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni, “What you can cram into a single vector: Probing sentence embeddings for linguistic properties,” *arXiv:1805.01070*, 2018.
- [43] J. Hewitt and C. D. Manning, “A structural probe for finding syntax in word representations,” in *North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.
- [44] F. Wenginger, F. Eyben, and B. Schuller, “On-line continuous-time music mood regression with deep recurrent neural networks,” in *ICASSP*, 2014.
- [45] E. Koh and S. Dubnov, “Comparison and analysis of deep audio embeddings for music emotion recognition,” *arXiv:2104.06517*, 2021.
- [46] Pioneer, “rekordbox v3.2.2,” 2015. [Online]. Available: <http://www.cp.jku.at/datasets/giantsteps/>
- [47] J. Jiang, G. G. Xia, and D. B. Carlton, “MIREX 2019 submission: Crowd annotation for audio key estimation,” *MIREX*, 2019.
- [48] F. Medhat, D. Chesmore, and J. Robinson, “Masked conditional neural networks for audio classification,” in *International Conference on Artificial Neural Networks*, 2017.
- [49] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Python in Science Conference*, 2015.
- [50] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models,” *arXiv:2001.08361*, 2020.
- [51] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *OpenAI Blog*, 2018.
- [52] P. Hamel and D. Eck, “Learning features from music audio with deep belief networks,” in *ISMIR*, 2010.
- [53] J. Weston, S. Bengio, and P. Hamel, “Multi-tasking with joint semantic spaces for large-scale music annotation and retrieval,” *Journal of New Music Research*, 2011.
- [54] K. Seyerlehner, M. Schedl, R. Sonnleitner, D. Hauger, and B. Ionescu, “From improved auto-taggers to improved music similarity measures,” in *International Workshop on Adaptive Multimedia Retrieval*, 2012.
- [55] X. Favory, K. Drossos, T. Virtanen, and X. Serra, “Learning contextual tag embeddings for cross-modal alignment of audio and tags,” *arXiv:2010.14171*, 2020.

- [56] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, 2021.
- [57] A. Baevski, M. Auli, and A. Mohamed, "Effectiveness of self-supervised pre-training for speech recognition," *arXiv:1911.03912*, 2019.
- [58] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv:2006.11477*, 2020.
- [59] P. Verma and J. Smith, "A framework for contrastive and generative learning of audio representations," *arXiv:2010.11459*, 2020.
- [60] D. Eck and J. Schmidhuber, "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks," in *IEEE Workshop on Neural Networks for Signal Processing*, 2002.
- [61] I. Simon and S. Oore, "Performance RNN: Generating music with expressive timing and dynamics," 2017. [Online]. Available: <https://magenta.tensorflow.org/performance-rnn>
- [62] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *ICLR*, 2019.
- [63] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *ISMIR*, 2019.
- [64] H.-T. Hung, C.-Y. Wang, Y.-H. Yang, and H.-M. Wang, "Improving automatic jazz melody generation by transfer learning techniques," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, 2019.

VARIABLE-LENGTH MUSIC SCORE INFILLING VIA XLNET AND MUSICALLY SPECIALIZED POSITIONAL ENCODING

Chin-Jui Chang

Research Center for IT Innovation,
Academia Sinica
reichang182@gmail.com

Chun-Yi Lee

Department of Computer Science,
National Tsing Hua University
cylee@cs.nthu.edu.tw

Yi-Hsuan Yang

Yating Music Team
Taiwan AI Labs
yhyang@ailabs.tw

ABSTRACT

This paper proposes a new self-attention based model for *music score infilling*, i.e., to generate a polyphonic music sequence that fills in the gap between given past and future contexts. While existing approaches can only fill in a short segment with a fixed number of notes, or a fixed time span between the past and future contexts, our model can infill a variable number of notes (up to 128) for different time spans. We achieve so with three major technical contributions. First, we adapt *XLNet*, an autoregressive model originally proposed for unsupervised model pre-training, to music score infilling. Second, we propose a new, musically specialized positional encoding called *relative bar encoding* that better informs the model of notes' position within the past and future context. Third, to capitalize relative bar encoding, we perform *look-ahead onset prediction* to predict the onset of a note one time step before predicting the other attributes of the note. We compare our proposed model with two strong baselines and show that our model is superior in both objective and subjective analyses.

1. INTRODUCTION

A growing body of research work has adopted deep learning techniques to generate music sequentially, taking only the past context as a condition while generating. This paper deals with a different setting where both the past and future context are given, called *music score infilling* [1–9]. As depicted in Figure 1, the goal of this task is to generate a short piece of symbolic music that fits in the middle gap. The generated piece is expected to meet certain requirements, e.g., being coherent with the contexts, and having an appropriate duration span such that the generated piece will not overflow to the outside of the designated region. Models with such a capability are useful in a few scenarios. For instance, one may have an inspiration to write two segments of melodies but somehow do not know how to connect them [5]. Or, a music piece may have an impaired part in the middle that requires restoration [10].

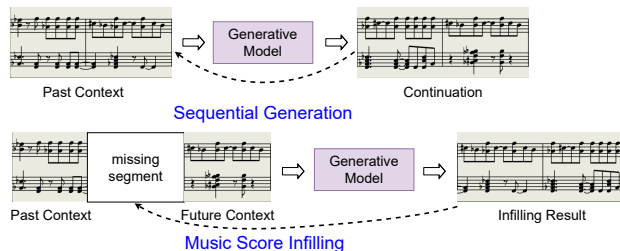


Figure 1: Comparison between sequential generation and music score infilling (a.k.a., music score inpainting).

As reviewed in Section 2, existing models for score infilling can be categorized by their representation of musical scores. Among them, we are interested in the case when music is represented as a sequence of *event tokens* such as note-on and note-off [11], for such a *token-based* representation facilitates the use of the self-attention based Transformers [12] for model building, which has been shown to outperform recurrent neural networks (RNN) in modeling sequences [13, 14].¹ However, Transformers are originally designed for sequential generation: predicting the future given the past tokens. Adapting it to account for both the past and future contexts while generating the missing token sequence in the middle may not be trivial.

The Infilling Piano Performances (IPP) [9] is the only pioneering work adapting Transformers for music score infilling, to our best knowledge. It achieves so with a simple approach of “reordering”: the past and future contexts are concatenated and placed *before* the missing segment as the *prompt*. A standard Transformer decoder is then trained to autoregressively generate a continuation of a given prompt. We note that this approach has a strong limitation: the past and future contexts need to consist of a fixed number of c notes each, and the missing segment a *fixed* number of k notes (c and k can be different). Moreover, as reported in [9], empirically their model works well only for infilling a *short* sequence with $k = 16$ missing notes; for larger k , its performance deteriorates and the infilled sequence cannot connect well with the future context.

In *text infilling* [25–30], the music infilling counterpart in natural language processing (NLP), many approaches have been based on Transformers. For instance, Infilling Language Model (ILM) [29] uses special tokens to in-

¹Token-based representations have also been heavily adopted by recent Transformer models for sequential MIDI music generation [15–24].



form the language models where to infill text. FELIX [30] enables BERT [13] to solve the infilling task by letting BERT predict [PAD] tokens for redundant masked positions. Both models can perform *variable-length text infilling*, but they were both tested only on infilling short sequences with less than 10 consecutive tokens (words).

This paper proposes a new self-attention based model to attain *long* and *variable-length music score infilling*. In our experiment, the model is able to generate a variable-length polyphonic infilled sequence with up to 768 tokens, or 128 notes, given the past and future contexts.² Moreover, our model is able to infill spans of different length (e.g., 2 bars to 4 bars) without re-training for each span length. We achieve so with three technical contributions. First, we employ *XLNet* [31], a Transformer encoder-based model, as the model architecture for the first time for music generation. Unlike other bidirectional models such as BERT [13], XLNet can attend to the past and future contexts while maintaining its autoregressive predicting order. We show that music infilling can be attained by XLNet via a specific factorization order of the token sequence.

Second, we point out that the original XLNet can infill only a fixed-length token sequence, because it relies on the *vanilla* positional encoding³ that requires the length of the missing segment to be known and fixed in advance. We propose a *musically specialized positional encoding* and a modification of the two-stream attention mechanism of XLNet to make it feasible for variable-length infilling. Specifically, our model represents the distance between two tokens in terms of *the number of bars* between them, rather than the exact number of intermediate tokens. Doing so, the positions of notes are also specified in a musically more meaningful way.

Third, with our special positional encoding, we need to know the musical position of the next note to be predicted in the autoregressive generation process. Therefore, we adapt the multi-output methodology of the *Compound Word* (CP) Transformer [24] to perform *look-ahead onset prediction* at each timestep. Specifically, each time, our model predicts the *content*-related tokens of the *current* note (i.e., PITCH, DURATION, VELOCITY, and TEMPO), and the *position*-related tokens (i.e., BAR and SUB-BEAT) of the *next* note to look one note ahead.

For evaluation, we compare our model with the two text infilling state-of-the-arts, ILM [29] and FELIX [30], that we extend to accept the same token representation as ours and to generate variable-length infilled sequences. The results show that our model outperforms these two strong baselines in both objective and subjective analyses.

For reproducibility, we open source our code at GitHub, along with examples of the infilling result.⁴

² As described in Section 3.1, we represent a musical note with 6 tokens (PITCH, DURATION, etc) in total.

³ Unlike RNNs, the Transformers do not have a built-in notion of the sequential order of tokens, and thus need to rely on the so-called *positional encodings* that “assign” positions to each token [32, 33]. This is usually done by using a token’s *absolute* position in the sequence [12], or by its *relative* distance to other tokens [34–36].

⁴ <https://github.com/reichang182/variable-length-piano-infilling>

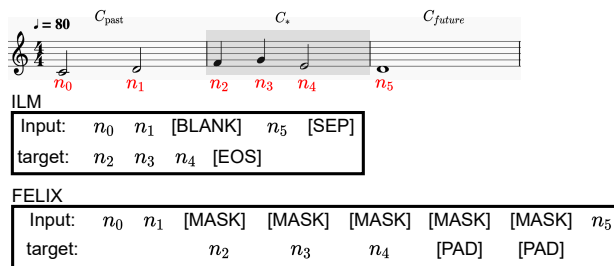


Figure 2: An illustration of how the baselines, ILM [29] and FELIX [30], solve the infilling task. For FELIX, the pre-defined mask length is set to five in this illustration.

2. BACKGROUND

2.1 Related Work on Music Score Infilling

Existing work can be divided by their data representation:

DeepBach-like. DeepBach [1] predicts a missing note based on the information from the notes around. They use two RNNs to aggregate the past and future contexts and a feedforward neural network for the notes occurring at the same temporal position as the current target note. They use pseudo-Gibbs sampling to improve the generated score gradually. Anticipation-RNN [2] introduces a constraint-RNN to the generation model, enforcing the model to consider the user-defined constraints while generating. Music InpaintNet [3] also uses an RNN to integrate the information within the context. However, they use the encoder of a bar-wise variational auto-encoder (VAE) to encode measures into latent vectors first and use the decoder of the VAE to reconstruct measures from the latent vectors. Music SketchNet [4] extends the work of Music InpaintNet to allow their model to consider user preferences.

DeepBach-like representation represents notes in an elegant way. But, it is mainly suitable for scores with a constant number of voices, e.g., four voices for Bach chorale and one voice for Irish and Scottish folk tunes [37], since representing multiple notes played at the same time requires more tracks added to the representation. Prior arts on DeepBach-like representation all use RNNs to deal with contexts, while we use Transformer-based models, which have been shown more powerful [16, 38].

Piano roll. Coconet [6] trains a convolutional neural network (CNN) to complete partial music score and uses blocked Gibbs sampling as an analogue to rewriting. Nakamura *et al.* [7] use CNN with deconvolutional layers at the end. The whole model is trained under the framework of generative adversarial networks. ES-Net [8] represents music as a sequence of *edit events*, each of which denotes either an addition or removal of a note. ES-Net is able to modify a music score while preventing the accumulation of errors that autoregressive models are prone to have.

The piano roll representation typically encodes information concerning pitch, pitch duration, and beat position only, not tempo and velocity, which are important to form an expressive piece. Moreover, as the CNN treats a piano roll as an fixed-size image, all existing piano roll-based models can only infill spans of fixed-length (e.g., 2 bars)

once the CNN was trained. On the contrary, a single model trained with our methodology can be applied to tasks with missing spans of different length (e.g., 2 bars to 4 bars).⁵

Aside from differences in data representation, some existing methods impose additional assumptions on data. For example, the use of bar-wise VAE in Music InpaintNet and Music SketchNet restrict their usage to cases where the missing segments start and end at precisely the start or end position of a bar, while our model is free of this restriction.

Event tokens. IPP [9] is the only work we are aware of that considers music as a sequence of tokens and employs the Transformer as the backbone model. Our work differs from theirs in two aspects. First, their model can only infill a fixed number of tokens, while ours can do variable-length infilling. Second, while we both group tokens related to a note to a tuple (cf. Section 3.1) [15, 24], our representation is based on the beat-based CP tokens [24], which have built-in notion of bars and beats.

Converting our data to be acceptable to the aforementioned models and making them produce variable-length infilling is not trivial. Hence, in our experiment we adopt text infilling models [29, 30] as the baselines, for they are both able to infill variable-length segment by design, not these music score infilling models.

2.2 Related Work on Text Infilling

ILM [29] replaces the missing segment with a single special token [BLANK]. The model learns to predict the original contents at the end of the sequence in an autoregressive manner. ILM only changes the input order of tokens and does not modify the attention mechanism of the vanilla Transformer. **FELIX** [30] uses BERT to derive the capacity of utilizing bidirectional contexts. For variable-length infilling, the missing segment is replaced with a series of [MASK] tokens of a pre-defined length that is sufficiently long. The model learns to predict the tokens to infill and [PAD] tokens to indicate no token here. FELIX is different from our model and ILM in that it predicts all tokens in the missing segment at once, and does not consider previously generated tokens during generation. Both ILM and FELIX were tested on infilling less than 10 words in the original papers, while we consider up to 128 notes in our task. Figure 2 illustrates how these two baselines work.

3. METHODOLOGY

We follow the definition of music score infilling in [3]: given a **past context** C_{past} and a **future context** C_{future} , the task is to generate an **infilled segment** C_* , which connects C_{past} and C_{future} in a musically meaningful way. During training, the model should maximize the likelihood:

$$P(C_* | C_{\text{past}}, C_{\text{future}}). \quad (1)$$

We consider in this paper the case where the model is a Transformer encoder and its input is a token sequence com-

⁵ Moreover, both DeepBach-like and piano roll require a token to hold at each position; e.g., 100 tokens are needed to represent one second of piano performance at a temporal resolution of 10ms, regardless of how many note events there are [15].

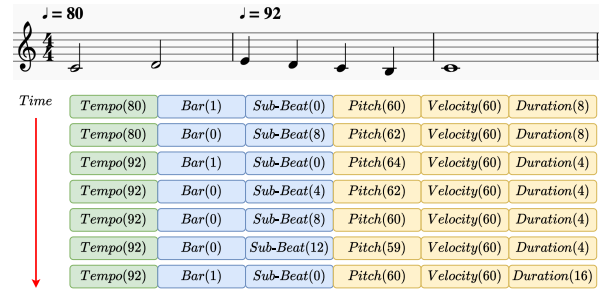


Figure 3: An example of a piece of score encoded using our representation. Note that the BAR and SUB-BEAT tokens are for positioning a note event on the time grid [20].

Token type	Voc. size	Values
Tempo	47	28, 32, ..., 212
Bar	2	0, 1
SUB-BEAT	16	0, 1, ..., 15
Pitch	86	22, 23, ..., 107
Velocity	33	0, 4, ..., 128
Duration	16	1, 2, ..., 16

Table 1: The token vocabulary used in our experiments. Note that all the vocabulary sizes do not count special tokens such as <EOS> and <PAD>, since the use of the special tokens are model-dependent.

posing of $\{C_{\text{past}}, \text{a masked version of } C_*, C_{\text{future}}\}$. The target output is the sequence $\{C_{\text{past}}, C_*, C_{\text{future}}\}$. Model loss is computed over only the middle part related to C_* .

Moreover, we desire our model not to generate C_* all at once, but one token at a time in an autoregressive manner. This way, the model builds C_* progressively by considering its previously generated tokens. The training objective can be factorized to

$$\prod_{0 < i \leq T} P(n_i | C_{\text{past}}, C_{\text{future}}, n_j, 0 < j < i), \quad (2)$$

where n_1, \dots, n_T are the tokens in C_* . Please note that, in our setting, the number of tokens T is variable, so does the number of tokens within C_{past} and C_{future} , respectively.

3.1 Compound Word-based Token Representation

We modify the beat-based token representation REMI [20] and CP [24] to encode our music data. As illustrated in Figure 3,⁶ we describe different attributes of a musical note through six different tokens—three note-related ones, PITCH, DURATION, and VELOCITY, and three metric-related ones, TEMPO, BAR, and SUB-BEAT.⁷ Table 1 shows the vocabulary of the adopted token representation. BAR is encoded to 1 for encountering a new bar, while encoded to 0 for staying at the same bar. SUB-BEAT is the position of a note within one bar, represented in a resolution of the 16-th note. Following the *multi-output* method-

⁶ For simplicity, we use monophonic pieces as examples in all the figures in the paper, though we actually use polyphonic pieces in experiment.

⁷ Even though it is not that reasonable to associate a TEMPO event with each note, we found that the models learn to predict similar tempos for adjacent notes, thus barely lower the quality of music.

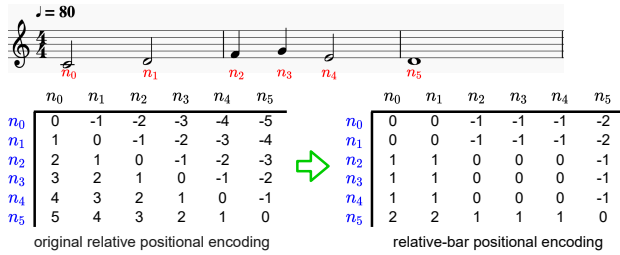


Figure 4: (Left) the widely-used relative positional encoding vs. (right) the proposed relative-bar positional encoding. Each row shows the positional encodings of a token.

ology of the CP Transformer [24], we may consider the six tokens defining a note as a group and predict them altogether *at once* at each timestep. Accordingly, the n_i in Eq. (2) is actually a “super token” (also referred to as a “note” or a CP token hereafter) comprising six tokens. When a CP token is masked, all its six constituent tokens are masked.

Below, we also refer to BAR and SUB-BEAT as the *onset* of a note, and the other four tokens as the note’s *content*.

3.2 Learning Bidirectional Contexts through XLNet

We adapt XLNet [31] to predict the masked (missing) tokens of C_* . Unlike other bidirectional models such as BERT [13], XLNet can effectively address Eq. (2) due to its special *two-stream self-attention* mechanism.

Models such as BERT cannot address Eq. (2) because the missing parts of the input (i.e., n_j in Eq. (2)) is replaced with a series of masked (CP-)tokens and those parts cannot be seen by the notes to be predicted after (i.e., n_i , $j < i$). The idea of two-stream self-attention is to separate the input into two streams—the *content stream* and the *query stream*. Each masked token n_i is inferred through the query stream, which masks the content of the target token at the timestep i . But, in inferring n_i , we can attend to the content of other tokens n_j that are before n_i through the content stream, which does not mask any tokens.

The original XLNet model is general and does not require the masked tokens to be consecutive as the case considered in Eq. (1). It covers music infilling as a special case with the following specific permutation order in its *permutation language modeling*: $C_{\text{past}} \rightarrow C_{\text{future}} \rightarrow C_*$.

3.3 A New Positional Encoding

The adapted XLNet considers both C_{past} and C_{future} and does well in fixed-length infilling, i.e., for scenarios where the number of tokens in C_* is known or pre-defined. However, to extend the model to variable-length infilling, the vanilla positional encodings [34] employed in the original XLNet (and most Transformer-based models) to realize the sequential order of the tokens become a problem. *Without knowing the number of tokens in C_* , we cannot assign proper positional encoding to the notes in C_{future} .*⁸

To address this issue, we propose a novel *relative bar encoding* to replace the original vanilla relative positional

⁸ Specifically, while we know the length of C_* at training time, we do not know its length at inference time.

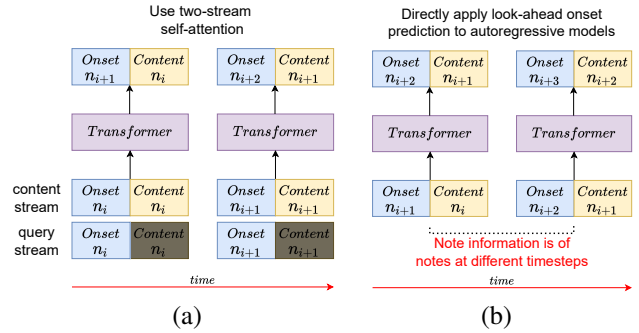


Figure 5: Look-ahead onset prediction with (a) the XLNet (those shaded are masked tokens) and (b) a Transformer decoder; the input tokens in (b) are unsynchronized.

embedding [34] adopted by XLNet. While the original positional encoding represents the relative distance between two notes in terms of the number of intermediate notes, the proposed method represents the distance by the number of bars in between. For instance, tokens within the same bar are 0 bar apart, and thus get 0 for the relative-bar positional encoding. However, for notes in the next bar, the current note is one bar before, and thus will get -1 for the relative-bar positional encoding. In this way, we only need to know how many bars C_{past} and C_{future} are apart to assign the relative bar positional encoding to their and C_* ’s notes.⁹ See Figure 4 for an illustration.

3.4 Look-ahead Onset Prediction through XLNet

While Section 3.1 suggests that we predict the six tokens of a note n_i at the same timestep, this is actually not ideal when it comes to exploiting the relative-bar positional encoding. The problem is that we need to know the onset of n_i beforehand to assign a proper relative-bar positional encoding to its corresponding input (i.e., a masked CP token) to the Transformer. To this end, we propose *look-ahead onset prediction*, where the onset of a note n_i is inferred one timestep ahead. Specifically, we modify the XLNet such that *the onset of n_i* (i.e., BAR or SUB-BEAT) is predicted along with *the content of n_{i-1}* (i.e., four content-related tokens) at timestep $i - 1$. The onset of n_i is then fed to the model at timestep i as the query stream input, with the content part of the query stream input masked, to infer the content of n_i . This is illustrated in Figure 5(a).

There are two design details. First, the onset of the first note to be infilled should be provided by the user during inference phase, which may be desirable as the user can decide where to start infilling. Second, the onset of the next note also serves as a stop signal; once the model predicts a special [EOS] token for either BAR or SUB-BEAT of the next note, the infilling process comes to an end.

The overall architecture is shown in Figure 6. The query stream inputs are the same as the content stream input ex-

⁹ Moreover, relative bar encoding actually provides more musically meaningful information to models than the original positional encoding does. For example, five notes being played at the same time are considered as four notes apart for the farthest two notes by the model with the original positional encoding. However, they are actually notes with the same onset time in a score.

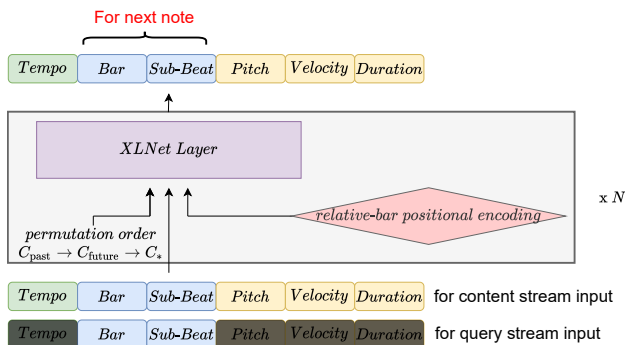


Figure 6: An illustration of the overall architecture proposed in this paper. (1) A specific permutation order of sequence is given to the model. (2) Tokens except for BAR and SUB-BEAT are masked to form the query stream input of XLNet. (3) Relative-bar positional encodings are used instead. (4) BAR and SUB-BEAT from the output of XLNet are the musical position of the next note.

cept that the content of notes are replaced by mask tokens, since those are the parts to be inferred. The onset part of the note (i.e., BAR and SUB-BEAT) is made visible, to allow the model to exploit the onset information. We note that the BAR token fed as input to the query stream only tells the model whether the note is in a new bar or stays in the same bar, but not how many bars apart the current note to the other notes in C_{past} , C_{future} and the rest of C_* . Therefore, the relative-bar positional encoding is still needed.

3.5 The Necessity of Two-Stream Self-Attention

We are now ready to elaborate more why we are in favor of XLNet instead of a Transformer decoder such as the one used by IPP [9]. As depicted in Figure 5(b) and exemplified in Figure 7, to realize the look-ahead onset prediction needed by the proposed relative bar encoding, the onset-related tokens and content-related tokens in the input to a Transformer decoder would be *unsynchronized*. For instance, the onset of the first input is for n_{i+1} , yet the content is for n_i . Such a mismatch impedes the Transformer decoder to attend to proper notes through the dot product of the input embeddings. This is not a problem for XLNet, since the input tokens in either the content stream or query stream remain *synchronized* (e.g., both for n_i).

4. EXPERIMENTAL SETUP

Dataset. We use the AILabs-Pop1k7 dataset shared publicly by Hsiao *et al.* [24], which contains 1,748 MIDI files of polyphonic pop piano performances, all in 4/4 time signature. We further quantize the tempo, beat position, duration, and velocity to reduce the vocabulary size, setting the 16-th note as our minimal temporal resolution for beat position and duration. There are on average 12.6 notes per bar. We crop the music into 16-bar pieces with an 8-bar overlap between successive pieces, yielding in total 19,789 16-bar data for model training.

Detailed Settings. We train all the models on 16-bar data with up to 512 CP tokens and design the experiment

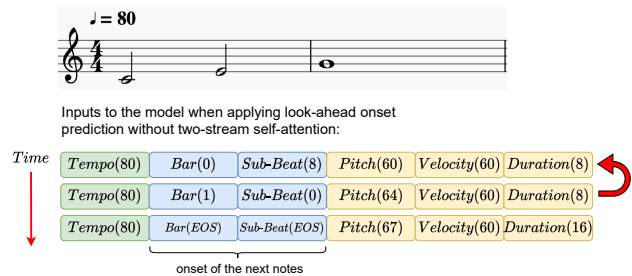


Figure 7: Illustration of the problem of a Transformer decoder such as IPP [9] for realizing look-ahead onset prediction. To infer the third note in the sequence, it may be beneficial if the model can attend to the first note, since both notes are at the same sub-beat position (though in different bars). However, because the onset-related and content-related tokens are unsynchronized (cf. Figure 5(b)), the model may not be able to attend to the proper notes.

with the following conditions in mind. First, C_{past} and C_{future} must be long enough to provide sufficient contextual information. Second, the length and the musical position of C_* should not be fixed, since we do not know where the missing segment starts and how many notes should be infilled for various cases in the real world. Consequently, for each token sequence, a range within the middle four bars, i.e., bar7 to bar10, is randomly selected to be C_* , such that C_{past} and C_{future} are at least 6 bars long, respectively. Note C_* is not required to start right at the beginning of bar7. The minimum number of CP tokens to be infilled is set to half the number of CP tokens within bar7 and bar10.¹⁰

A CP token is transformed before being fed to models. First, each of the six tokens composing the CP token is mapped to an embedding with size 256 using a lookup table. Then, these embeddings are concatenated and merged through a linear layer not shared across models, producing a merged embedding of size 768. All the models accept these merged embeddings as input and each has 8 heads, 12 self-attention layers, and intermediate layers of dimension 3,072. The output from these models is transformed back to probabilities with linear projection followed by softmax. At inference time, the tokens are sampled through *nucleus* [39] with temperature 1.0 and threshold 0.9.

5. EXPERIMENTAL RESULTS

5.1 Objective Evaluation

We evaluate these models with a number of metrics proposed in [22], which are *pitch class histogram entropy* and *grooving pattern similarity*. The former provides us an indicator to the usage distribution of each pitch class within 1 bar and 4 bars, resulting in metrics \mathcal{H}_1 and \mathcal{H}_4 . The latter evaluates the rhythmic pattern similarity between bars (\mathcal{GS}). Since the goal of the task is to connect contexts and generate fluent music, these metrics calculated on C_*

¹⁰ And, note that we do not expect the models to rely on the absolute musical position within an entire music piece to inference, but should only rely on the note’s contexts around. Thus, when providing the 16-bar data to the models, the absolute position of these 16-bar data within an entire music piece is discarded.

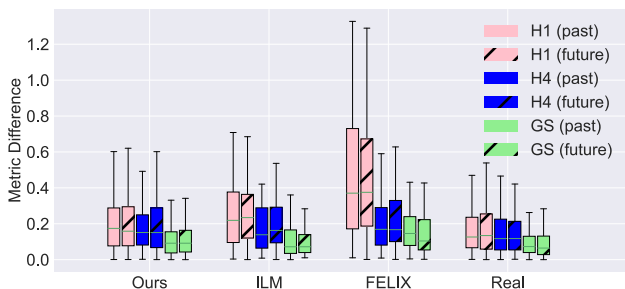


Figure 8: The difference between “ C_* and C_{past} ” and between “ C_* and C_{future} ” in 3 objective metrics: \mathcal{H}_1 , \mathcal{H}_4 , \mathcal{GS} .

should be close to those calculated on C_{past} and C_{future} . Thus we calculate the difference of metrics between C_* and C_{past} , and also between C_* and C_{future} . The lower the values are, the better the model is. Since these are bar-wise metrics, we let the models generate all four middle bars, i.e., bar7 to bar10, instead of a portion of them. From Figure 8, it seems that our model and ILM learn well to generate C_* coherent to its contexts. The results are even close to that of the real data. In contrast, FELIX performs poorly in \mathcal{H}_1 . It is possibly due to the dependency problems between the masked tokens, since each token is unaware of what other tokens predict, leading to uncertain tonality.

We also adapt the *MIREX-like prediction test* [22,40,41] for testing the models’ ability to infer the correct answer from contexts. The test includes 1,000 questions generated from a held-out validation set, with each question comprises 6-bar C_{past} and C_{future} . The goal of this test is to select the right infilling from four choices. A model makes a choice by calculating the average probability of the first few notes belonging to each choice, and selects the one with the highest probability. The number of notes to average from is dependent on the shortest length of all the choices. In addition, the choices could be randomly selected from a *different* song, or somewhere from the *same* song. The latter is harder since the choices are much more similar. We name them the **simple test** (choices come from different music) and the **hard test** (from the same music). The results in Table 2 show that our model outperforms the other two consistently in both tests. It should be noted that stability is an important factor for autoregressive models in this test, since a wrongly predicted note leads to the accumulation of errors in the following prediction. While both our model and ILM could suffer from such a problem, our model still performs slightly better.

5.2 Subjective Evaluation

A user study is conducted with in total 30 subjects, where 7 of them are deemed as professionals according to a question about their musical background. Each subject is presented with 3 sets of music randomly selected from the total 15 sets of music. Within each set, a subject listens to a music piece with a missing segment, and is then presented with 4 music pieces, where 3 of them are generated by the models (Ours, ILM, and FELIX) and 1 of them is the real music without the missing segment. The subject is

	VLI (ours)	ILM [29]	FELIX [30]
simple test	0.940*	0.796	0.919
hard test	0.467**	0.361	0.398

** : leads all others with $p < .01$; * : with $p < .05$

Table 2: Accuracy in the MIREX-like prediction test. **VLI** denotes the proposed variable-length piano infilling model.

		M	R	I	F
all	VLI (ours)	3.27	3.43	2.93	27%
	ILM [29]	2.63	2.63	2.67	17%
	FELIX [30]	2.83	2.77	2.57	16%
	Real	3.83	3.73	2.83	41%
pro	VLI (ours)	3.08	3.38	2.77	24%
	ILM [29]	2.69	2.77	2.62	10%
	FELIX [30]	2.85	2.62	2.62	14%
	Real	3.77	3.92	2.92	52%

Table 3: Results of the user study: mean opinion scores in 1–5 in **M** (melodic fluency), **R** (rhythmic fluency), **I** (impression), and percentage of votes in **F** (favorite), from ‘all’ the participants or only the music ‘pro’-essionals.

asked to rate each of the 4 music pieces according to its 1) **melodic fluency**: how many wrong notes are there? The fewer wrong notes, the higher the score; 2) **rhythmic fluency**: Are there notes played at the wrong time? The less, the higher the score; 3) **impression**: how much the subject is impressed? The more, the higher the score. After listening to the 4 music pieces, they are asked to choose a **favorite** one from them. From the results in Table 3, our model does beat the other baselines by a large margin. Our model even has a higher impression score than the real music when considering all subjects. However, there is still a gap between ours and the real data in the “favorite” score, suggesting that the music generated by our model is still differentiable from the real music.

6. CONCLUSION & DISCUSSION

In this paper, we have proposed a new model adapted from XLNet to address music score infilling. Specifically, to make the model able to perform variable-length infilling, we replace the token-based distance attention mechanism in Transformers with a musically specialized one considering relative bar distance. We have also reported evaluations showing that our model outperforms two strong baselines.

We do not pay attention to whether the past and future contexts are similar in style or theme in this work. It would be interesting to see in a future work whether our model can infill a nice transition between two dissimilar segments. Moreover, by changing the permutation order, our model can be applied to sequential generation in the future, to study whether the relative bar encoding improves the metrical structure of the generated music.

7. ACKNOWLEDGEMENT

The authors are grateful to Dr. Chris Donahue, the first author of the ILM paper [29], for fruitful discussions during the initial phase of the project. We also thank the anonymous reviewers for their constructive suggestions, and the HuggingFace team for releasing their implementation of Transformer models that are used in our work.

8. REFERENCES

- [1] G. Hadjeres, F. Pachet, and F. Nielsen, “DeepBach: a steerable model for Bach chorales generation,” in *Proc. International Conference on Machine Learning (ICML)*, 2017, pp. 1362–1371.
- [2] G. Hadjeres and F. Nielsen, “Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation,” *Neural Computing and Applications*, 2018.
- [3] A. Pati, A. Lerch, and G. Hadjeres, “Learning to traverse latent spaces for musical score inpainting,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [4] K. Chen, C.-I. Wang, T. Berg-Kirkpatrick, and S. Dubnov, “Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [5] T. Bazin and G. Hadjeres, “NONOTO: A model-agnostic web interface for interactive music composition by inpainting,” in *Proc. International Conference on Computational Creativity (ICCC)*, 2019.
- [6] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by convolution,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [7] K. Nakamura, T. Nose, Y. Chiba, and A. Ito, “A symbol-level melody completion based on a convolutional neural network with generative adversarial learning,” *Journal of Information Processing*, vol. 28, pp. 248–257, 2020.
- [8] W. Chi, P. Kumar, S. Yaddanapudi, R. Suresh, and U. Isik, “Generating music with a self-correcting non-chronological autoregressive model,” *ArXiv*, vol. abs/2008.08927, 2020.
- [9] D. Ippolito, A. Huang, C. Hawthorne, and D. Eck, “In-filling piano performances,” in *Proc. NIPS Workshop on Machine Learning for Creativity and Design*, 2018.
- [10] A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley, “Audio inpainting,” *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, pp. 922 – 932, 03 2012.
- [11] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, “This time with feeling: Learning expressive musical performance,” in *Neural Computing and Applications*, vol. abs/1808.03715, 2018.
- [12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conference on North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019, pp. 4171–4186.
- [14] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplín, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, “A comparative study on Transformer vs RNN in speech applications,” in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, 2019.
- [15] C. Hawthorne, A. Huang, D. Ippolito, and D. Eck, “Transformer-NADE for piano performances,” in *Proc. NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.
- [16] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. M. Shazeer, A. M. Dai, M. Hoffman, M. Dinculescu, and D. Eck, “Music Transformer: Generating music with long-term structure,” in *Proc. International Conference on Learning Representations (ICLR)*, 2019.
- [17] C. M. Payne, “MuseNet,” *OpenAI Blog*, 2019.
- [18] C. Donahue, H. H. Mao, Y. E. Li, G. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 685–692.
- [19] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, “Encoding musical style with transformer autoencoders,” in *Proc. International Conference on Machine Learning (ICML)*, 2020, pp. 1899–1908.
- [20] Y.-S. Huang and Y.-H. Yang, “Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions,” in *Proc. ACM Multimedia*, 2020, pp. 1180—1188.
- [21] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, “Automatic composition of guitar tabs by Transformers and groove modeling,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [22] S.-L. Wu and Y.-H. Yang, “The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

- [23] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, “PopMAG: Pop music accompaniment generation,” in *Proc. ACM Multimedia*, 2020, pp. 1198–1206.
- [24] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, “Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs,” in *Proc. AAAI Conference on Artificial Intelligence*, 2021.
- [25] W. Fedus, I. Goodfellow, and A. Dai, “MaskGAN: Better text generation via filling in the _____,” in *Proc. International Conference on Learning Representations (ICLR)*, 2018.
- [26] W. Zhu, Z. Hu, and E. P. Xing, “Text infilling,” *arXiv preprint arXiv:1901.00158*, 2019.
- [27] T. Shen, V. Quach, R. Barzilay, and T. S. Jaakkola, “Blank language models,” in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 5186–5198.
- [28] Y. Huang, Y. Zhang, O. Elachqar, and Y. Cheng, “IN-SET: Sentence infilling with INter-SENTential Transformer,” in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 2502–2515.
- [29] C. Donahue, M. Lee, and P. Liang, “Enabling language models to fill in the blanks,” in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 2492–2501.
- [30] J. Mallinson, A. Severyn, E. Malmi, and G. Garrido, “FELIX: Flexible text editing through tagging and insertion,” in *Proc. Findings of the Association for Computational Linguistics*, 2020, pp. 1244–1255.
- [31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [32] G. Ke, D. He, and T.-Y. Liu, “Rethinking the positional encoding in language pre-training,” in *Proc. International Conference on Learning Representations (ICLR)*, 2021.
- [33] B. Wang, L. Shang, C. Lioma, X. Jiang, H. Yang, Q. Liu, and J. G. Simonsen, “On position embeddings in BERT,” in *Proc. International Conference on Learning Representations (ICLR)*, 2021.
- [34] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, “Transformer-XL: Attentive language models beyond a fixed-length context,” in *Proc. Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019, pp. 2978–2988.
- [35] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2018, pp. 464–468.
- [36] A. Liutkus, O. Cifka, S.-L. Wu, U. Şimşekli, Y.-H. Yang, and G. Richard, “Relative positional encoding for Transformers with linear complexity,” in *Proc. International Conference on Machine Learning (ICML)*, 2021.
- [37] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, “Music transcription modelling and composition using deep learning,” in *Proc. Conference on Computer Simulation of Musical Creativity*, 2016.
- [38] S.-L. Wu and Y.-H. Yang, “MuseMorphose: Full-song and fine-grained music style transfer with just one Transformer VAE,” *arXiv preprint arXiv:2105.04090*, 2021.
- [39] A. Holtzman, J. Buys, M. Forbes, and Y. Choi, “The curious case of neural text degeneration,” in *Proc. International Conference on Learning Representations (ICLR)*, 2020.
- [40] “The “Patterns for Prediction Challenge” of Music Information Retrieval Evaluation eXchange,” [Online] https://www.music-ir.org/mirex/wiki/2019:Patterns_for_Prediction.
- [41] B. Janssen, T. Collins, and I. Ren, “Algorithmic ability to predict the musical future: Datasets and evaluation,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 208–215.

SURPRISENET: MELODY HARMONIZATION CONDITIONING ON USER-CONTROLLED SURPRISE CONTOURS

Yi-Wei Chen

Hung-Shin Lee

Yen-Hsing Chen

Hsin-Min Wang

Institute of Information Science, Academia Sinica, Taiwan

chenvictor@iis.sinica.edu.tw

ABSTRACT

The surprisingness of a song is an essential and seemingly subjective factor in determining whether the listener likes it. With the help of information theory, it can be described as the transition probability of a music sequence modeled as a Markov chain. In this study, we introduce the concept of deriving entropy variations over time, so that the surprise contour of each chord sequence can be extracted. Based on this, we propose a user-controllable framework that uses a conditional variational autoencoder (CVAE) to harmonize the melody based on the given chord surprise indication. Through explicit conditions, the model can randomly generate various and harmonic chord progressions for a melody, and the Spearman’s correlation and p-value significance show that the resulting chord progressions match the given surprise contour quite well. The vanilla CVAE model was evaluated in a basic melody harmonization task (no surprise control) in terms of six objective metrics. The results of experiments on the Hooktheory Lead Sheet Dataset show that our model achieves performance comparable to the state-of-the-art melody harmonization model.

1. INTRODUCTION

In recent years, deep learning has developed rapidly and has become the main technology for automatic music generation. In this study, we focus on the task of automatic melody harmonization, in which a system needs to assign appropriate and harmonic chords to a given melody. From previous studies, we have seen that the models based on the bidirectional long short-term memory (BiLSTM) perform well in this task [1, 2]. Most of them can generate harmonic chords to harmonize a melody. In [3], by introducing blocked Gibbs sampling and class weighting, the model can further generate more reasonable and interesting chords, and is even comparable to human composers.

Based on these previous studies, we hope to further control the model to generate chords according to both

melody conditions and user instructions. Latent representation learning is a powerful method that has been widely used in computer vision [4–6] and speech processing fields [7, 8] to learn semantically meaningful information of attributes. Such techniques have also been applied to music processing in several recent works [9–11]. Motivated by these latent variable models, we modify the variational autoencoder (VAE) [12] so that the chord-to-chord mapping can be trained under the conditions of a melody sequence and a user-defined temporal contour.

However, in the model, what are the most attractive and practical controllable conditions for users? Many high-level subjective emotions, such as happiness, sadness, surprisingness, and interestingness, can describe a musical sequence. These emotions seem to be abstract and difficult to quantify objectively. Fortunately, some of them can be calculated from the information dynamics [13]. Previous music theory studies have shown that perceptual qualities and subjective states, such as uncertainty, surprisingness, complexity, tension, and interestingness, are closely related to the measurement of information theory, such as relative entropy and mutual information. In [13], the authors explored this idea in the context of Markov chains using musical sequences, resulting a structural analysis, which is largely consistent with the views of professional human listeners. Therefore, we use the surprisingness metric, which is defined as the negative log transition probability in a Markov chain, to generate the time-varying surprisingness contour of a chord sequence.

Similar to Mellotron, a text-to-speech system that uses pitch and rhythm contours as conditions to synthesize speech [14], in the training stage, we concatenate a chord sequence with additional conditions, namely its corresponding melody and surprise contour, feed them into an encoder to convert them into latent variables. Then, the latent variables are concatenated with the melody and surprise contour again, and input into a decoder to reconstruct the chord sequence. The model is expected to learn the latent representations of chords when the melody and surprise contour are given. Owing to the sampling mechanism of the VAE-based model, in the inference stage, we can randomly sample latent variables from the standard normal distribution to generate a variety of chords, which cohere the input melody and propagate according to the required surprise contour. In addition, we extend the 96 chords used in [3] to all chord types in the Hooktheory Lead Sheet



Dataset [15] to expand the chord selection of the model.

Our model is named SurpriseNet. It can harmonize a melody through user-controllable conditions. The highlights of SurpriseNet are two-fold. First, it relieves the tension between coherence and surprisingness caused by the melody and user-supplied condition, respectively. In general, it is easy to catch one factor (i.e., surprisingness) and lose another. Second, the results show that the vivid and harmonic chords generated by our model can not only correspond to the melody, but also strictly follow the given surprise contour. Several examples are available at <https://scmv301135.github.io/SurpriseNet>.

2. RELATED WORK

2.1 Automatic Melody Harmonization

Automatic melody harmonization aims to establish a learning model that can generate chord sequences to accompany a given melody [16, 17]. In music, a chord is an arbitrary harmonic set consisting of three or more notes, which sounds as if these notes are sounding simultaneously [18]. Conventionally, methods based on hidden Markov models (HMMs) [19–21] and genetic algorithms (GA) [22] are commonly used to deal with the task.

Recently, some models based on deep learning have been proposed. For example, Lim *et al.* first proposed a model based on the BiLSTM [1]. The melody is input to the model to predict the simplified 24 chords (i.e., the major and minor triads) in the Wikifonia corpus. Based on the same model architecture in [1], Yeh *et al.* proposed a model called MTHarmonizer, in which the chord types were extended to 48 by considering major, minor, augment and diminished chords in a larger corpus (i.e., the Hooktheory Lead Sheet Dataset) [2]. In addition, they integrated information of chord functions [23] into the loss function to help chord label prediction [2]. However, there are several drawbacks in the above methods, such as overusing common chords and incorrect phrasing problems. Sun *et al.* tried to solve these problems and produced interesting but still reasonable chords by introducing the orderless NADE training techniques, class weights, and Blocked Gibbs sampling into their model. They also extended the chord space to 96 types, including major, minor, augment, diminish, suspend, major7, minor7, and dominant7 [3].

2.2 Controllable Music Generation

Music generation can be regarded as a conditional estimation problem defined as $p(\text{music}|\text{condition})$, where both “music” and “condition” are usually time-series features. Related tasks include melody-based chord generation [17] and chord-based melody generation [23, 24].

An alternative way is to learn the joint distribution $p(\text{music}, \text{condition})$, and then set the condition during the generation process. Related tasks include automatic music completion or accompaniment based on the melody or chords [25–29]. However, many abstract music factors, such as music texture, melody contour, or other high-level subjective perception, are difficult to be explicitly encoded.

Latent representation learning is an ideal solution to the above problem, because representation learning embeds discrete music and condition sequences into a continuous latent space, and accurately captures the latent information from the music. Recent research has used disentangled representation learning to achieve controllable music generation models for style transfer, texture variation, and accompaniment arrangement [11]. High-level subjective perception can also be captured in the latent space to generate music following the arousal condition [9]. These studies show that VAEs [30, 31] are an effective framework for learning the representation of discrete music sequences. We incorporated this idea into our research, and expected the model to capture the latent information of chords when conditioned by melody sequences and surprise contours.

3. METHODOLOGY

In this section, we will introduce the calculation of surprise contours and the model architecture in detail. SurpriseNet is based on a conditional VAE, and its goal is to learn the representation of chords when conditioned by the melody sequences and surprise contours. In the inference process, the random latent variables, melody conditions, and surprise contours are provided to the decoder to produce harmonization.

3.1 Surprise Contour

Measures such as entropy and mutual information can be used to characterize random processes. One of the salient effects of listening to music is to create expectations for what is to come next, which may be fulfilled immediately, after a delay, or not at all depending on the situation. An essential aspect of this is that music is experienced as a phenomenon that “unfolds” over time, rather than being apprehended as a static object presented in its entirety [13].

Consider a snapshot of a stationary random process taken at a certain time: we can divide the timeline into the past and present parts, denoted as $t - 1$ and t , respectively. Here we will consider one of the simplest random processes, a first-order Markov chain. Let S be a Markov chain with a finite state space $\{1, \dots, N\}$ such that S_t is the random variable representing the t -th element of the sequence. We can establish a transition matrix $a \in \mathbb{R}^{N \times N}$ encoding the distribution of any element of the chord sequence given the previous element, that is $p(\text{chord}_t|\text{chord}_{t-1}) = a_{t,t-1}$. According to the definition in [13], the surprise contour can be derived as the negative log probability as

$$\text{Surprisingness} = -\log p(\text{chord}_t|\text{chord}_{t-1}). \quad (1)$$

Equation (1) is actually the definition of the information content in information theory. It means that in a chord sequence, the higher the surprise value at a certain time, the greater the amount of information at that time.

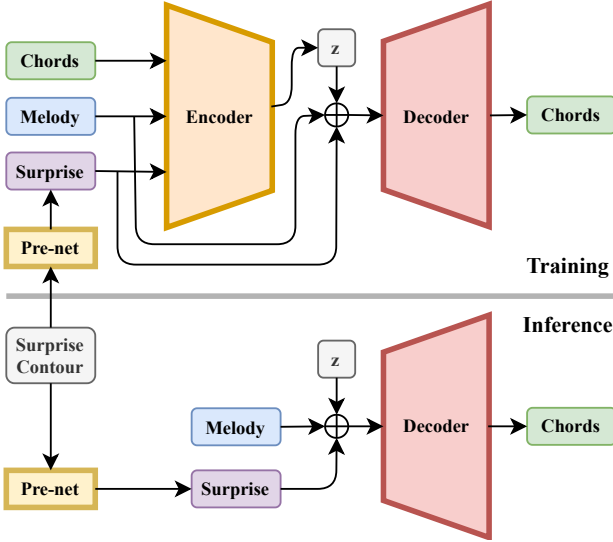


Figure 1. The structure of SurpriseNet. In the inference stage, only Decoder and Pre-net are used.

3.2 VAE and Conditional VAE

As described in [31], a common goal for various kinds of autoencoders is that they compress the salient information within the input sample into a lower-dimensional latent code. Ideally, this will force the model to produce compact representations to capture the important factors of variation in the dataset. In pursuit of this goal, VAE [30, 32] introduces the constraint that the latent code \mathbf{z} is a random variable distributed according to a prior $p(\mathbf{z})$.

The generative process of VAE is described as follows. A latent variable \mathbf{z} is generated from the prior distribution $p(\mathbf{z})$, and the observation \mathbf{x} is generated by the generative distribution $p(\mathbf{x}|\mathbf{z})$ conditioned on \mathbf{z} ; that is, $\mathbf{z} \sim p(\mathbf{z})$ and $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$. A VAE consists of an encoder θ , which approximates the posterior $p(\mathbf{z}|\mathbf{x})$, and a decoder ϕ , which parameterizes the likelihood $p(\mathbf{x}|\mathbf{z})$. Following the framework of variational inference, we do posterior inference by minimizing the KL divergence between the approximate posterior (i.e., the output of the encoder) and the true posterior $p(\mathbf{z}|\mathbf{x})$ by maximizing the evidence lower bound (ELBO). The objective function of VAE with Gaussian latent variables is

$$\tilde{\mathcal{L}}_{VAE} = \mathbb{E}[\log p_{\theta}(\mathbf{x}|\mathbf{z})] - KL(p_{\phi}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \quad (2)$$

In the common case where $p(\mathbf{z})$ is a diagonal-covariance Gaussian distribution, this can be circumvented by replacing $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma\mathbf{I})$ with

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \epsilon, \quad (3)$$

where ϵ is a small random factor.

As for the conditional VAE [12], the conditional generative process of the model is given as follows. For a given condition \mathbf{c} , \mathbf{z} is drawn from the prior distribution $p(\mathbf{z}|\mathbf{c})$ realized by a standard Gaussian distribution, and the output \mathbf{x} is generated from the distribution $p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{c})$. Therefore, the training objective function can be expressed as

$$\tilde{\mathcal{L}}_{CVAE} = \mathbb{E}[\log p_{\theta}(\mathbf{x}|\mathbf{z}, \mathbf{c})] - KL(p_{\phi}(\mathbf{z}|\mathbf{x}, \mathbf{c})||p(\mathbf{z}|\mathbf{c})). \quad (4)$$

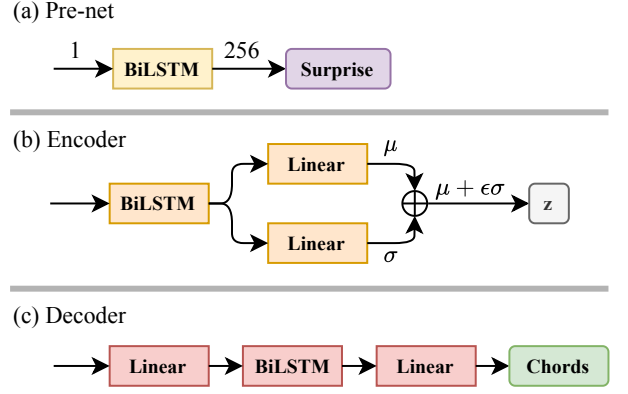


Figure 2. Three main components of SurpriseNet.

Our work is divided into two parts. We first train the conditional VAE models for 96 or 633 types of chords, conditioned only on the melody, to observe the performance in completing the basic harmonization task. Next, we select the better conditional VAE to train the final SurpriseNet in combination with the surprise contours.

The main architecture of SurpriseNet and its key components are shown in Fig. 1 and Fig. 2, respectively. Our components of VAE follow the structures used in [33] and MusicVAE [31], and finally combine with the conditional part [12] to complete the harmonization task. In the training stage, the surprise contour is processed by the Pre-net implemented by a BiLSTM, as shown in Fig. 2(a), to extend the feature from a scalar to a 256-dimensional vector. Afterwards, the features of the chord, melody, and extended surprise contour are concatenated and fed into the encoder to generate a latent code \mathbf{z} subject to a standard normal distribution. The encoder is also implemented by a BiLSTM, as shown in Fig. 2(b), where the size of each layer is 256 or 512. Different from the RNN-based VAE [33] and MusicVAE [31], the frame-wise outputs are further transformed by two linear layers to respectively generate the sequential latent variables, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, with dimensionalities of 16 or 64.

As for the decoder, instead of performing it in an autoregressive manner as in MusicVAE, we concatenate \mathbf{z} , melody, and extended surprise representation frame by frame as inputs of the decoder to reconstruct the chord sequence. The number of layers and hidden size in the decoder are the same as those in the encoder. Dropout was employed with a rate of 0.2 on each BiLSTM to prevent overfitting. The batch size was set to 64. Early stopping for 10 epoch patience was applied.

In the inference stage, given the melody and the surprise contour, the surprise contour is first processed by the Pre-net. Then, the latent variable is sampled from a standard normal distribution. Finally, the latent variable, melody, and extended surprise representation are input to the decoder to complete the harmonization process. The implementation details of the model are available at <https://github.com/scmvp301135/SurpriseNet>.

4. EXPERIMENTS

4.1 Datasets

We performed experiments on the Hooktheory Lead Sheet Dataset (HLSD) [15], which contains high-quality and human-arranged melodies with chord progressions. The dataset is provided in two formats, event-based JSON files and MIDI files. Furthermore, there are many types of labels on chords, such as chord symbols and Roman numerals for reference. The dataset contains a total of 633 chord types, including 9th, 11th, 13th, half diminished, and slash chords. In previous studies conducted on the dataset [1–3], the number of chord types was simplified to 48 (major and minor triads) or 96 (major, minor, augment, diminish, suspend, major7, minor7, and dominant7). In this study, we experimented with two settings, 96 and 633 chord types.

We followed the data split in [3]; the training set contains 17,505 samples, the test set contains 500 samples. For each song, the melody and chords are aligned every two beats in a measure, and the chords are encoded into a one-hot format for training.

4.2 Objective Metrics

For objective evaluation, we used six different objective metrics proposed in [2]. The first three metrics measure the quality of the chord progression, and the others measure the harmonicity between the melody and the chords.

- **Chord histogram entropy (CHE):** The entropy of the chord histogram.
- **Chord coverage (CC):** The number of chord types in a chord sequence.
- **Chord tonal distance (CTD):** The tonal distance between two chords when they are represented by 6-D feature vectors [34].
- **Chord tone to non-chord tone ratio (CTnCTR):** The ratio of the number of chord tones to the number of non-chord tones.
- **Pitch consonance score (PCS):** The sum of the consonance scores between a melody note and each note in a given chord.
- **Melody-chord tonal distance (MCTD):** The tonal distance between a melody note and a chord when they are represented by 6-D feature vectors.

4.3 Surprise Contour Evaluation

The task of user-controlled melody harmonization has never been seen in the literature. Therefore, there is no baseline systems for comparison. We decided to use some statistical methods to evaluate the correlation or causation between a given contour and the generated sample, instead of subjective testing.

Unlike the pitch and rhythm error evaluation in Melotron [14], we used Spearman’s correlation, which is suitable for continuous and discrete ordinal values between two variables. The p-value was used to determine the significance of the results under the assumption that there is no significant correlation between the surprisingness trend

Table 1. Objective evaluation results with respect to various models. For the metrics related to Chord Progression, the higher value in CHE and CC means the higher diversity of the generated chords, and the lower value in CTD implies that the chord progression is smoother. As for the metrics related to Harmonicity, the higher value in CTnCTR and PCS and the lower value in MCTD indicate better harmonization results. The arrow denotes whether the metric is the larger the better or the lower the better.

Chord Progression	CHE↑	CC↑	CTD↓
Humans	1.266	4.344	0.628
Sun <i>et al.</i> , $ S = 96$	1.280	4.900	0.730
CVAE, $ S = 96$	1.210	4.712	0.577
CVAE weight, $ S = 96$	1.670	7.360	0.620
CVAE, $ S = 633$	1.644	7.074	0.730
CVAE weight, $ S = 633$	1.934	9.890	0.649
M/C Harmonicity	CTnCTR↑	PCS↑	MCTD↓
Humans	0.726	0.515	1.276
Sun <i>et al.</i> , $ S = 96$	0.887	0.652	1.052
CVAE, $ S = 96$	0.851	0.611	1.110
CVAE weight, $ S = 96$	0.841	0.523	1.190
CVAE, $ S = 633$	0.767	0.530	1.229
CVAE weight, $ S = 633$	0.705	0.476	1.290

of the predicted chord progression and the given surprise contour. A small p-value indicates strong evidence against the assumption, which means that the results are correlated to some extent.

The reason for not using error evaluation, such as the mean squared error (MSE), is that the harmonization result is jointly decided by the melody and the surprise contour. If the error between the generated trend and the given surprise contour is zero, it means that the model completely follows the surprise contour and ignores the melody conditions. Obviously, this is not acceptable and will lead to discordant harmonization results.

5. RESULTS

In this section, we will first compare the conditional VAE models with Sun *et al.*’s model [3] and human performance in terms of six objective metrics. Then, we will illustrate some harmonization samples generated by SurpriseNet based on different surprise contours. The last part is the correlation analysis.

5.1 Objective Evaluation

The objective evaluation results are shown in Table 1. Compared with the results of humans and Sun *et al.*’s model, we can see that the vanilla CVAE model without using class weights (cf. CVAE, $|S| = 96$) achieved comparable results in all metrics. It performed better in CTD, indicating that the generated chord progression is smoother. After introducing chord balancing (cf. CVAE weight, $|S| = 96$), the CVAE model learned how to sample rare chords, thereby increasing the chord diversity, as

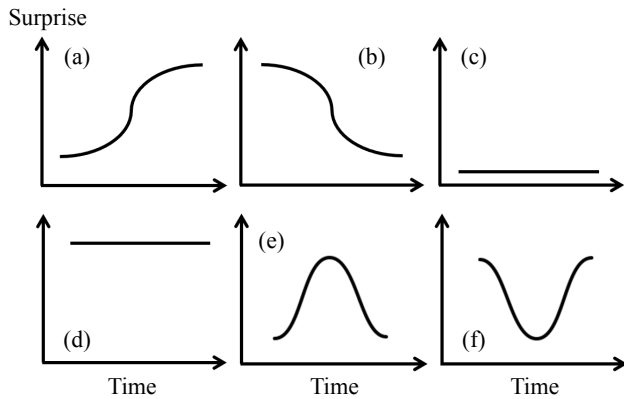


Figure 3. The six given surprise contours.

shown in the results in CC and CHE.

We further expanded the chord space to 633 to maintain the integrity of the chords in the dataset. Due to the extension of the chord dimension, the deeper CVAE without using class weights (cf. CVAE, $|S| = 633$) obtained results comparable to the vanilla CVAE model with chord balancing (cf. CVAE weight, $|S| = 96$). After introducing chord balancing (CVAE weight, $|S| = 633$), the deeper CVAE model achieved the best chord diversity, with an average of nearly 10 chord types in a musical sequence.

Despite the above improvements, trade-offs in other metrics (such as CTnCTR, PCS, and MCTD) can be observed. As pointed out in [3], rare chords, such as 7th, 9th, 11th, and 13th, will cause a degradation in the melody/chord harmonic metrics, but this is mainly due to the definition of CTnCTR, PCS, and MCTD. In order to maintain the diversity of chords, we decided to train SurpriseNet with the conditional VAE architecture considering 633 chord types.

5.2 Generated Samples

We compared the chord generation results of SurpriseNet (based on CVAE, $|S| = 633$) and weighted SurpriseNet (CVAE weight, $|S| = 633$) based on 6 representative surprise contours, as shown in Fig. 3. The 6 contours were generated by the sigmoid function, plain line, and normal distribution, and their reversed profiles, respectively. We intended to check whether the generated chord progression really follows the surprise contour to harmonize the given melody.

To use the sigmoid function to represent the surprise contour, we first normalized it to match the maximum value in the surprise contours of the training data. This contour (cf. Fig. 3(1)) represents a song with lower chord variation in the first half and higher chord variation in the second half. The reverse sigmoid function leads to the opposite trend (cf. Fig. 3(2)). In the case of plain line, we used two sequences consisting of zero (cf. Fig. 3(3)) and the maximum value (cf. Fig. 3(4)) as the surprise contours. A surprise contour with all values being zero indicates that there should be no fluctuation in the chord sequence. In other words, it is expected to see that the given melody will always be harmonized with the same chord. As for the surprise contour with all values being the maximum, it

indicates that there should be a lot of up-and-down changes in the resulting chord sequence. That is, it is expected to see that the given melody will be harmonized with various chords. These two cases are considered the most extreme cases. According to the normal distribution, we expect that the highest arousal will appear in the middle of the harmonization result (cf. Fig. 3(5)). As for its inverse profile (cf. Fig. 3(6)), it is expected to generate a plain and more predictable result in the middle of the chord sequence.

Fig. 4 and Fig. 5 show the harmonization results of SurpriseNet and weighted SurpriseNet for a 4-measure melody, respectively. They are displayed in the same order as the function types in Fig. 3. From Fig. 4(1), as expected, we can see that SurpriseNet generated continuous C chords for the first two measures at the beginning, and then generated varying chords for the last part of the song, according to the given sigmoid-like surprise contour in Fig. 3(1). From Fig. 4(2), we can also see that SurpriseNet generated different chords at the beginning, and then generated more C and G chords that appeared previously, following the given reverse sigmoid-like surprise contour in Fig. 3(2). As for the results of weighted SurpriseNet (see Figs. 5(1) and 5(2)), the chord progressions generated were not exactly as expected, but some surprising and complicated chords were brought in for users' reference. Next, given the all-zero surprise contour in Fig. 3(3), it is obvious that both models followed the condition to generate only one type of chord in the results (see Figs. 4(3) and 5(3)). As for the all-maximum surprise contour in Fig. 3(4), as shown in Figs. 4(4) and 5(4), it is also obvious that the results generated by the two models changed in the chord type almost every two beats, which is the minimum time unit for changing the chord in the training data. For the normal distribution contour in Fig. 3(5), the result generated by SurpriseNet is roughly as expected, with more chord changes in the middle of the song (see Figs. 4(5)). But the result of weighted SurpriseNet is quite different from expectations (see Figs. 5(5)). For the inverse normal distribution contour in Fig. 3(6), the results of both models are not in line with our expectations. But we can see that they are similar to the results generate based on the reverse sigmoid-like surprise contour in the beginning part of the song. When the model is initially assigned a high surprise value, the trend in the first part of the output is similar.

In summary, the above samples generated by SurpriseNet are almost in good agreement with our expectations. The model can indeed generate chords that have a tendency to follow a given surprise contour. However, weighted SurpriseNet seems to over concentrate on using more complicated or rare chords to harmonize the melody due to the class penalty (i.e., weights), so that the trend is not clearly consistent with the given contour.

5.3 Correlation Measurement

Because there is no existing model for this task, we use Spearman's ρ and p-value to evaluate the correlation and significance between the surprisingness values in the given surprise contour and the surprisingness values in the gen-



Figure 4. Samples generated by SurpriseNet.

Table 2. Spearman’s ρ and p-value significance of SurpriseNet and weighted SurpriseNet.

Method	Spearman’s ρ	p-value
SurpriseNet	0.517	< 0.001
Weighted SurpriseNet	0.406	< 0.001

erated chord progression. From Table 2, we can find that there is indeed a certain correlation between the given surprise contour and the generated chord progression. Furthermore, SurpriseNet seems to be more controllable than weighted SurpriseNet, with a higher Spearman’s ρ . The p-value shows that there is no significant difference between the surprisingness trend of the given surprise contour and the surprisingness trend of the generated chord progression for the two models. The result implicitly confirms that given a surprise contour and a melody, these models can generate the corresponding chords as instructed to complete the melody harmonization task, thereby achieving a user-controlled model.

6. FUTURE WORK

The HLSD dataset contains rich intonation data, such as Roman, symbol, secondary, and mode data. We can introduce some approaches from the NLP field, such as modeling these data by referring to various language models. Moreover, in this work, the rhythmic type of chords is simplified and restricted to two beats in a measure. But in fact, there are various rhythmic types in the dataset, such as syncopation and tuplets. Perhaps a disentangled representation learning model can be used to capture this in-



Figure 5. Samples generated by weighted SurpriseNet.

formation, so that we can implement an omni model with more complicated rhythms.

In addition, surprise is still an open for discussion topic. In this work, we only consider the surprise in the chord sequence. In the future, can also consider the surprise in the melody sequence at the same time. Moreover, the surprise can be considered not only as the conditional probability of past chord events but also as the conditional probability of the melody sequence at that time. These different considerations will bring different meanings to the surprise.

7. CONCLUSIONS

In this paper, we proposed SurpriseNet, which is based on a conditional VAE model and combines a surprise contour from the transition probability in a Markov chain, to achieve a user-controlled melody harmonization task. From the generated samples, we observed that the model could accurately generate various harmonic chord progressions according to the given surprise contours. The Spearman’s correlation and p-value significance show that there is a positive correlation between the given surprise contour and the generated chord progression.

The vanilla conditional VAE model was evaluated in the basic melody harmonization task. The objective evaluation results show that the conditional VAE model could achieve performance comparable to the state-of-the-art melody harmonization model [3]. We expanded the chord space from 96 to 633 to broaden the range of chord selection for the model. The conditional VAE model could generate more types of chords, such as 7th, 9th, 11, 13th, and slash chords, resulting in vivid and harmonic results.

8. REFERENCES

- [1] H. Lim, S. Rhyu, and K. Lee, “Chord generation from symbolic melody using BLSTM networks,” in *Proc. ISMIR*, 2017.
- [2] Y. C. Yeh, W. Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H. M. Liu, H. W. Dong, Y. Chen, T. Leong, and Y. H. Yang, “Automatic melody harmonization with triad chords: A comparative study,” 2020. [Online]. Available: <https://arxiv.org/abs/2001.02360>
- [3] C.-E. Sun, Y.-W. Chen, H.-S. Lee, Y.-H. Chen, and H.-M. Wang, “Melody harmonization using orderless NADE, chord balancing, and blocked Gibbs sampling,” in *Proc. ICASSP*, 2020.
- [4] H. Kim and A. Mnih, “Disentangling by factorising,” in *Proc. MLR*, 2018.
- [5] X. Chen, Y. Duan, R. Houthoofd, J. Schulman, I. Sutskever, and P. Abbeel, “InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets,” in *Proc. NIPS*, 2016.
- [6] Y. Li and S. Mandt, “Disentangled sequential autoencoder,” 2018. [Online]. Available: <https://arxiv.org/abs/1709.01620>
- [7] W. N. Hsu, Y. Zhang, and J. Glass, “Unsupervised learning of disentangled and interpretable representations from sequential data,” in *Proc. NIPS*, 2017.
- [8] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, F. Ren, Y. Jia, and R. A. Saurous, “Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis,” in *Proc. ICML*, 2018.
- [9] H. H. Tan and D. Herremans, “Music fadernets: controllable music generation based on high-level features via low-level feature modelling,” in *Proc. ISMIR*, 2020.
- [10] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, “MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer,” in *Proc. ISMIR*, 2018.
- [11] Z. Wang, D. Wang, Y. Zhang, and G. Xia, “Learning interpretable representation for controllable polyphonic music generation,” in *Proc. ISMIR*, 2020.
- [12] K. Sohn, X. Yan, and H. Lee, “Learning structured output representation using deep conditional generative models,” in *Proc. NIPS*, 2015.
- [13] S. Abdallah and M. Plumbley, “Information dynamics: Patterns of expectation and surprise in the perception of music,” *J. Connection Science*, vol. 21, 2009.
- [14] R. Valle, J. Li, R. Prenger, and B. Catanzaro, “Melotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens,” in *Proc. ICASSP*, 2020.
- [15] C. Anderson, D. Carlton, R. Miyakawa, and D. Schwachhofer, “Hooktheory.” [Online]. Available: <https://www.hooktheory.com>
- [16] C.-H. Chuan and E. Chew, “A hybrid system for automatic generation of style-specific accompaniment,” in *Proc. IJWCC*, 2007.
- [17] I. Simon, D. Morris, and S. Basu, “MySong: automatic accompaniment generation for vocal melodies,” in *Proc. CHI*, 2008.
- [18] D. Makris, I. Kayrdis, and S. Sioutas, “Automatic melodic harmonization: An overview, challenges and future directions,” in *Trends in Music Information Seeking, Behavior, and Retrieval for Creativity*. IGI Global, 2016, pp. 146–165.
- [19] J. F. Paiement, D. Eck, and S. Bengio, “Probabilistic melodic harmonization,” in *Proc. Canadian AI*, 2006.
- [20] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii, “Function- and rhythm-aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences,” in *Proc. ISMIR*, 2017.
- [21] —, “Generative statistical models with self-emergent grammar of chord sequences,” *J. New Music Res.*, 2018.
- [22] T. Kitahara, S. Giraldo, and R. Ramírez, “JamSketch: Improvisation support system with GA-based melody creation from user’s drawing,” in *Proc. CMMR*, 2018.
- [23] T.-P. Chen and L. Su, “Functional harmony recognition of symbolic music data with multi-task recurrent neural networks,” in *Proc. ISMIR*, 2018.
- [24] L. C. Yang, S. Y. Chou, and Y. H. Yang, “MidiNet: A convolutional generative adversarial network for symbolic-domain music generation,” in *Proc. ISMIR*, 2017.
- [25] G. Hadjeres, F. Pachet, and F. Nielsen, “DeepBach: A steerable model for bach chorales generation,” in *Proc. ICML*, 2017.
- [26] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, “Xiaoice band: A melody and arrangement generation framework for pop music,” in *Proc. ACM SIGKDD*, 2018.
- [27] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Proc. AAAI*, 2018.
- [28] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *Proc. ISMIR*, 2019.

- [29] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," 2018. [Online]. Available: <https://arxiv.org/abs/1806.00195>
- [30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. ICLR*, 2014.
- [31] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. ICML*, 2018.
- [32] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proc. ICML*, 2014.
- [33] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proc. SIGNLL*, 2016.
- [34] C. Harte, M. Sandler, and M. Gasser, "Detecting harmonic change in musical audio," in *Proc. AMCMM*, 2006.

SEMI-SUPERVISED VIOLIN FINGERING GENERATION USING VARIATIONAL AUTOENCODERS

Vincent K.M. Cheung Hsuan-Kai Kao Li Su
Institute of Information Science, Academia Sinica, Taiwan
{cheung,hsuankai,lisu}@iis.sinica.edu.tw

ABSTRACT

There are many ways to play the same note with the fingerboard hand on string instruments such as the violin. Musicians can flexibly adapt their string choice, hand position, and finger placement to maximise expressivity and playability when sounding each note. Violin fingerings therefore serve as important guides in ensuring effective performance, especially for inexperienced players. However, fingering annotations are often missing or only partially available on violin sheet music. Here, we propose a model based on the variational autoencoder that generates violin fingering patterns using only pitch and timing information found on the score. Our model leverages limited existing fingering data with the possibility to learn in a semi-supervised manner. Results indicate that fingering annotations generated by our model successfully imitate the style and preferences of a human performer. We further show its significantly improved performance with semi-supervised learning, and demonstrate our model’s ability to match the state-of-the-art in violin fingering pattern generation when trained on only half the amount of labelled data.¹

1. INTRODUCTION

Musicians produce different pitches on string instruments such as the violin and guitar by pressing on a particular string with their fingerboard hand (typically the left) to temporarily reduce its length. The string oscillates at a higher frequency and a higher pitch is consequently sounded. However, apart from the lowest and highest notes of the instrument, the mapping between pitch and fingering (i.e., where along the fingerboard and with which finger to press) is not unique [1].

For the violin, musicians are faced with the decision of selecting an appropriate string, hand position, and finger placement for every note they play [2]. Such decisions depend on the trade-off between artistic expression and playability [3]. For example, playing a note on the

¹Example code and supplementary information are available at <https://github.com/vkmcheung/violin-ssvae>

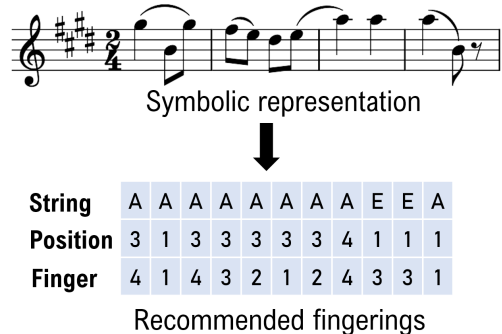


Figure 1. Our proposed learning model generates violin fingerings from pitch and timing information found on the score (Example: Elgar’s *Salut d’amour*, bars 1-4).

open string is the easiest as no finger placements are required [2]. However, its distinctive, brighter timbre is often undesired as it breaks the consistency in sound quality over a musical context [3, 4]. Musicians instead tend to play these notes on a lower string with vibrato to achieve a warmer and richer tone [5]. Likewise, using the same finger for different pitches consecutively is often avoided as this incurs a constant shift in hand position that could lead to poor intonation or unintended glissandi [1]. Selecting an effective string, position, and fingering combination to sound each note is therefore a non-trivial aspect of violin playing that could shape the outcome of a performance.

The importance of violin fingerings is evinced as they often appear on the musical score as performance directions or reminders for the musician [6]. They are also used as a pedagogical aid for inexperienced violinists [7]. However, the majority of violin sheet music does not come with fingering annotations. Sampling from the International Music Score Library Project (imslp.org), one of the largest digital repositories of public domain sheet music, 85% of 23,142 scores featuring the violin do not contain any fingering information². In other words, most annotations are still done by hand by the musician, in a process that requires experience and is often time-consuming [8]. Therefore, there is a crucial need for a model that not only generates fingerings, but also requires few labelled data as prior knowledge.

To this end, we propose a violin fingering generation model based on the variational autoencoder [9–11]. Our

²Determined by inspecting the first available violin part of every 400th entry in the category *Scores featuring the violin* on 3 May 2021.

model relies only on pitch and timing information found on the musical score as inputs, and can be trained in a semi-supervised manner. This allows our model to capitalise on the predominantly unlabelled existing data in generating fingerings that conform to the style and preferences of violinists.

2. RELATED WORK

Work on fingering generation is not exclusive to the violin and has previously been explored in other musical instruments such as guitar [8, 12, 13] and piano [14–16]. Nevertheless, despite the popularity of the instrument, there exist few models on violin fingering generation. Early approaches have focussed on fingering generation through heuristic rules and dynamic programming. For example, Maezawa et al. [1, 3, 17] introduced three ‘consistency rules’ for their model to ensure that generated fingerings were consistent in direction and magnitude during pitch and string changes, as well as during a mordent. Fingering generation was thus achieved by minimising the transition cost within a context of two and three notes using a musical score and an audio recording. However, the multimodal nature of the input and rigidity of these models mean that adapting generated fingerings to match individual preferences or styles is not straightforward.

Later works have remedied this problem with learning models. For example, hidden Markov models have been trained on violin textbooks [2] to complement partially annotated fingerings [18]. A recent deep learning model [7] has also combined a pretrained bidirectional long short-term memory (BLSTM) neural network with heuristic rules to generate fingerings with different options. This enabled musicians to select fingerings according to their preferences in e.g., staying in a lower position, or to minimise hand-position shifting. However, the paucity of violin sheet music with labelled fingerings means that there might not always be sufficient training data. By contrast, our semi-supervised approach enables our proposed model to make use of unlabelled data during training to generate high-quality fingering annotations even in the context of limited labelled data.

3. METHODS

Here, we briefly review the background behind semi-supervised variational autoencoders before introducing our proposed model and metrics for performance evaluation.

3.1 Variational autoencoders (VAEs)

VAEs [9, 10] are a popular class of deep generative models that are prized for their ability in estimating complex probability distributions through variational inference [19]. Let X be some observed data generated by latent variable z . We want to learn parameters θ and ϕ that optimise the (log-)likelihood $p_\theta(x|z)$, parametrised by θ , and approximate posterior $q_\phi(z|x)$, parametrised by ϕ . This is achieved by maximising the evidence lower bound (ELBO). If we further assume that $p(z)=\mathcal{N}(\mathbf{0}, I)$

and $q_\phi(z|x)=\mathcal{N}(z|\mu_\phi, \text{diag}(\sigma_\phi^2))$, then we can use a reparametrisation trick to write samples of z as transformations of a standard Gaussian random variable, i.e.

$$z_i = \mu_i + \sigma_i \epsilon \quad (1)$$

for some $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$. This allows us to compute gradients of the ELBO to optimise θ and ϕ using neural networks, for which $q_\phi(z|x)$ is often referred to as the encoder and $p_\theta(x|z)$ the decoder. In practice, a non-negative hyperparameter β is often added to the ELBO to control the extent to which the approximate posterior $q_\phi(z|x)$ resembles the prior $p(z)$, i.e.

$$\begin{aligned} \log p_\theta(x) &\geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] \\ &\quad - \beta D_{\text{KL}}(q_\phi(z|x) \parallel p(z)). \end{aligned} \quad (2)$$

We have the original VAE formulation when $\beta = 1$, whilst reconstruction is improved at the expense of a more entangled latent representation when $0 < \beta < 1$ [20, 21].

3.2 Semi-supervised VAEs

The intuition behind semi-supervised VAEs [11] is to capitalise on its generative ability and to extend the VAE latent space to include information from a classifier. Reconstruction errors from the unlabelled data can then be explicitly used to update the classifier during backpropagation.

Formally, let (X, Y) be some observed (partially-) labelled data generated by a continuous latent variable z . Suppose $p(z)=\mathcal{N}(z|\mathbf{0}, I)$ and $p(y)=\text{Cat}(y|\pi)$, where the latter is a multinomial distribution with distribution π , and that the likelihood $p_\theta(x|y, z)$ is parametrised using a neural network (the decoder). We can again use variational inference to approximate the intractable posterior $p(y, z|x)$ with $q_\phi(y, z|x)$.

Now assuming that $q_\phi(y, z|x)=q_\phi(y|x)q_\phi(z|x)$, we can construct the approximate posterior using a neural network with two components: a multinomial classifier $q_\phi(y|x)=\text{Cat}(y|\pi(x))$, and a Gaussian encoder with diagonal covariance matrix $q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma_\phi^2(x))$.

As before, finding suitable values for θ and ϕ amounts to optimising the ELBO. For labelled data, that is

$$\begin{aligned} \log p_\theta(x, y) &\geq \mathbb{E}_{q_\phi(z|x, y)}[\log p_\theta(x|y, z)] \\ &\quad - \beta D_{\text{KL}}(q_\phi(z|x) \parallel p(z)) = L(x, y). \end{aligned} \quad (3)$$

For unlabelled data, we can treat the missing label y' as an additional categorical latent variable that generates the observed data and assume that y', z are marginally independent. However, backpropagating through samples from a multinomial distribution is problematic as the operation is not differentiable. Fortunately, we can approximate this sampling operation with the Gumbel-Softmax distribution [22, 23], for which samples can be drawn via the reparametrisation trick

$$y'_i = \frac{\exp((\log(\pi_i) + g)/\tau)}{\sum_{j=1}^L \exp((\log(\pi_j) + g)/\tau)}, \quad (4)$$

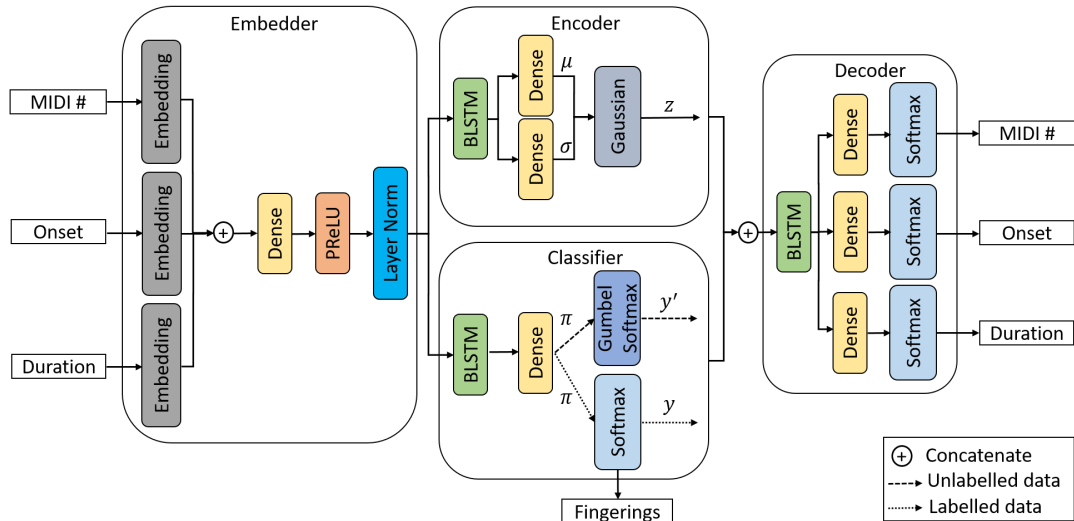


Figure 2. Model architecture. Our VAE-based model supports semi-supervision by treating missing labels from unlabelled data as an additional latent variable for reconstructing pitch and timing information.

where $g \sim \text{Gumbel}(0, 1)$, L denotes the number of classes, and τ controls how strongly the distribution approximates the multinomial distribution.

With the two reparametrisation tricks at hand, the ELBO is now maximised for unlabelled data as follows:

$$\begin{aligned} \log p_{\theta}(x) &\geq \mathbb{E}_{q_{\phi}(y', z|x)}[\log p_{\theta}(x|y', z)] \\ &\quad - \beta D_{\text{KL}}(q_{\phi}(z|x) \| p(z)) \\ &\quad - \beta D_{\text{KL}}(q_{\phi}(y'|x) \| p_{\pi}(y')) \\ &= U(x). \end{aligned} \quad (5)$$

Lastly, a classification loss is introduced to the classifier $q_{\phi}(y|x)$ for labelled data. The overall objective of this model is thus to maximise

$$\mathcal{J} = \mathbb{E}_{D_L} [L(x, y) + \log q_{\phi}(y|x)] + \mathbb{E}_{D_U} U(x), \quad (6)$$

where D_L and D_U denote labelled and unlabelled data, respectively.

3.3 Model architecture

Our proposed model (Figure 2) consists of four modules: embedder, encoder, classifier, and decoder. The embedder accepts a sequence of notes as inputs, where each note is represented by a numeric vector denoting its MIDI number, onset, and duration. The sequence is passed through embedding layers of dimension 16, 8, and 4 for the three respective features, concatenated, and then fed into a dense layer of 64 units with a PReLU activation function before leaving the module through a layer normalisation layer.

Outputs from the embedder are then passed in parallel onto the encoder and classifier. These go through a bidirectional long short-term memory (BLSTM) layer of 64×2 units in the encoder before being mapped onto a Gaussian latent space of 16 dimensions as output via a reparametrisation trick (Equation 1).

The embedder output is likewise passed through a BLSTM layer of 128×2 units in the classifier. This is followed by a dense layer of N_{spf} units, where N_{spf} denotes

the number of possible (string, position, finger) arrangements. By considering fingerings as the joint distribution of string, position, and finger, we can model dependencies between these three labels. Otherwise, different optimal (string, position, fingering) combinations might be predicted for each label separately if only their marginal distributions are considered. For labelled data, a softmax activation function is subsequently applied as output of the classifier. This provides a probability density estimate for each fingering combination given the input. For unlabelled data, logits from the dense layer are mapped onto a latent Gumbel-softmax distribution as outputs via a reparametrisation trick as described in Equation 4.

Finally, outputs from the encoder and classifier are concatenated and passed through a 128×2 -unit BLSTM layer in the decoder. This is followed by three softmax-activated dense layers of N_{MIDI} , N_{onset} , N_{duration} units as outputs, which denote the number of MIDI, onset, and duration classes, respectively.

3.4 Dataset

We use a recently published dataset of symbolic violin performance for the current study [7]. This dataset is a compilation of 217,690 note-by-note annotations of 14 solo violin excerpts as performed by 10 professional violinists. The excerpts are selected from diverse styles, covering Western classical music from the Baroque, Classical, and Romantic period, as well as Eastern folk melodies. Symbolic information from the score include pitch class and height of each note in addition to its onset and duration within the bar. They are accompanied by the corresponding string selection, hand position, and finger placement used by each musician when performing the piece. Additional descriptors include bar numbers and bowing, but were not used in our model as they are not always present in violin sheet music.

3.5 Implementation

Pitch class and height of each note in the dataset was converted into its corresponding MIDI number as numerical input into the model with $N_{\text{MIDI}} = 47$ to include all possible pitches on the violin (with 0 reserved for missing notes). As timing information in the dataset was based on subdividing the crotchet into $2^{10}=1024$ units, we chose to discretise onsets into $N_{\text{onset}} = 2^6+2^5=96$ categories (56 are present in the dataset) and duration into $N_{\text{duration}} = 32$ (26 present) to allow for generalisation beyond excerpts in the dataset.

String selection ($\{G, D, A, E\}$), hand position ($\{1, \dots, 12\}$), and finger placement ($\{0, \dots, 4\}$) were combined into a single label consisting of $N_{\text{spf}} = 241$ classes (with 0 reserved for missing fingerings).

The model was trained on different numbers of excerpts (see Section 4), but always tested on one, and we report results following leave-one-out cross-validation. To maintain stylistic consistency and for comparison with previous work [7], we derived training and test data from one violinist (#2 in the dataset). However, it is important to note that violin fingerings are highly individualised and are dependent on performers’ background and expert ability.

Training was implemented using batch size = 32 and optimised using Adam [24] with a learning rate of 0.01. Each excerpt was divided into sequences of length 32 for training using a hop size of 16 (i.e., half overlap), and sequences were right-padded with zeros to maintain the same length. We trained two separate models for labelled and unlabelled data simultaneously with shared layers, and oversampled the smaller dataset size to match the input sizes. Five percent of the training data was reserved for validation, and training was early-stopped [25] whenever total validation loss did not improve over 10 epochs, for which the best weights were retained.

Dense and BLSTM layers were initialised using a Glorot Uniform initialiser [26], whilst embedding layers were initialised from uniformly distributed samples. L1/L2 regularisation and L2 regularisation were respectively used for kernel and bias regularisation in the embedding and dense layers of the embedder module. In addition to kernel and bias regularisation, L2 recurrent regularisation was also used in all BLSTM layers. Finally, KL losses were weighted with $\beta = 0.001$ to improve reconstruction quality, and we set the Gumbel Softmax temperature $\tau = 0.75$.

3.6 Evaluation

We consider a variety of objective measures from information retrieval to evaluate model performance. The first is the F1 score, which we calculate using the model’s most probable predicted (string, position, finger) combination. Here, we consider the F1 score as a measure for how well our model replicates the fingering style of a performer, since each note can be played with multiple fingerings.

Nevertheless, since our model predicts a probability distribution of fingerings for each note, we can also examine the position to which the true label is ranked. This provides a measure for the quality of predicted fingerings. A

high, if not the highest, ranking should be assigned to the performer’s chosen fingering. One metric that captures this intuition is the mean reciprocal rank (MRR) [27], given by

$$MRR = \frac{1}{N} \sum_{j=1}^N \frac{1}{rank^{(j)}}, \tag{7}$$

where N is the number of notes in the training excerpt and $rank^{(j)}$ denotes the rank of which the true string, position, or finger first appears for note j . Note that because we modelled the joint distribution of these three labels, $rank^{(j)}$ may exceed the number of classes in each label.

Furthermore, given the variation in fingerings used across violinists, it would be interesting to examine the preference or relevance of our model’s predicted fingerings to other performers. We can capture this with a metric known as the normalised discounted cumulative gain (nDCG) [28]. First, we derive the relevance score of note j by calculating the proportion of violinists in the dataset (i.e., 10) who selected a given string, position, and finger to play the note. Next, we calculate the discounted cumulative gain (DCG) of j that is given by the sum of relevance scores for the model’s top K predicted labels weighted by its log rank, i.e.

$$DCG^{(j)} = \sum_{i=1}^K \frac{rel_i^{(j)}}{\log_2(i + 1)}, \tag{8}$$

where $rel_i^{(j)}$ denotes the relevance score of the i^{th} most probable predicted label for note j . The idealised DCG (iDCG) can also be computed by taking the DCG where the K labels are ranked from highest to lowest relevance. We can then obtain the normalised discounted cumulative gain (nDCG) at K of note j by dividing $DCG^{(j)}$ by $iDCG^{(j)}$, for which we take the mean across all notes in the testing excerpt.

4. RESULTS AND DISCUSSION

4.1 Style replication

We first consider the fully-supervised case, where our model was trained on 13 excerpts and tested on one. As shown in Figure 3 and Supplementary Table 1, our model generated violin fingerings with an MRR of 0.873 for string selection, 0.715 for hand position, and 0.721 for finger placement. These indicate that the true fingerings as performed by the violinist were predominantly given by the model’s most probable predictions. Examining the confusion matrix (Figure 4) more closely, we see that the model had a tendency towards predictions in first and third position. This can also be seen when the model predicted open strings played by the performer as to be played with the second or fourth finger 32% of the time. Interestingly, the converse was not true: the model seemed to have learnt that open strings should be stylistically avoided, as second and fourth finger placements by the performer were only respectively predicted as open strings by the model 3.5% and 6.5% of the time. However, in rare cases, our model

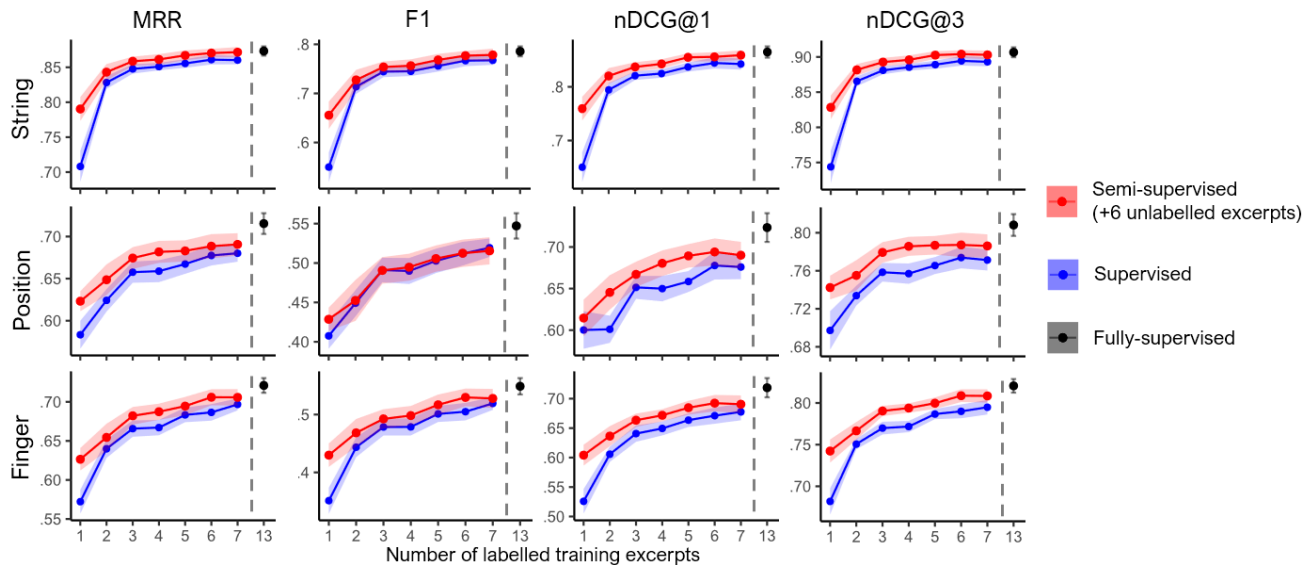


Figure 3. Examining effects of semi-supervision for different labelled training excerpt sizes. Significantly improved performance is observed when our model was trained on both labelled and unlabelled data. The fully supervised case is also shown for comparison. Filled circles and shaded regions denote mean and standard error, respectively.

failed to capture fingerings played in the 11 or 12th position. Upon inspection, we found that these notes were especially high (E7), and our model provided a fingering for the same pitch class but at an octave lower.

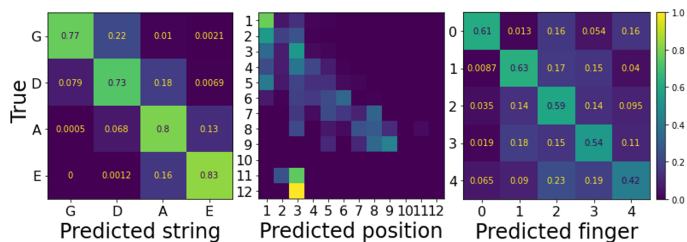


Figure 4. Confusion matrix for our fully supervised model (normalised such that each row sums to 1). Notice the tendency towards predictions in first and third position.

Regarding F1 scores, our model seemed to perform substantially better for string compared to position or finger. This is surprising since violin fingerings were modelled as the joint probability of (string, position, finger) combinations. One possible explanation could be due to the model’s tendency towards predictions in first and third position. An incorrectly predicted position would have led to an incorrect finger placement even if the string was correctly predicted. Nevertheless, compared to previous work [7], our model showed noticeable improvement in F1 scores for string, position, and finger, as well as comparable performance in MRR when tested on Elgar’s *Salut d’amour*³ (see Table 1 and Figure 1).

³ For comparison with previous work [7], we took the simple mean of F1 scores across each class instead of their class-size-weighted mean as in the rest of this paper.

	String		Position		Finger	
	MRR	F1	MRR	F1	MRR	F1
<i>Our model (semi-supervised)</i>						
1L+6U	.563	.291	.448	.120	.481	.147
4L+6U	.816	.714	.528	.128	.606	.277
7L+6U	.903	.834	.716	.214	.758	.500
<i>Our model (fully-supervised)</i>						
13L	.906	.830	.726	.305	.776	.636
<i>Previous work (Jen et al., 2021 [7])</i>						
13L	.913	.667	.729	.241	(-)	.412

Table 1. Comparing generated fingerings to Elgar’s *Salut d’amour*. Our model exceeded previous work when fully supervised, and achieved comparable performance when trained on far fewer labelled data under semi-supervision. *L* and *U* respectively denote number of labelled and unlabelled excerpts used for training.

4.2 Capturing preference across violinists

We next investigated to what extent the generated fingerings were actually performed (and thus regarded as preferred) by violinists in the dataset. The high mean nDCG@1 scores (Figure 3 and Supplementary Table 1) for string, position, and finger indicate that our model’s most probable fingering predictions matched those performed by the professionals, and interestingly, also resembled the MRR scores. This suggests that the fingering patterns learnt by our model corresponded to those that showed the least variation amongst the violinists (even though it was only trained on one). Higher mean nDCG@3 scores further indicate that the stylistic variation across violinists could be adequately captured within the model’s top three fingering predictions. Taken together, our evaluation measures suggest that the fingerings generated by our model matches the style and preferences of human performers.

4.3 Pitch and timing reconstruction

Our model was also able to reconstruct pitch and timing information with near-perfect MRR scores (all >0.945 , see Supplementary Table 2). To test the extent reconstruction depended on the classifier, we replaced its output with zeros during testing. Wilcoxon signed-rank tests revealed significant differences in MRR for pitch and duration ($p = .003$ and $p = .024$, respectively, corrected using Holm’s method) when information flow from the classifier to the decoder was blocked. This was associated with a 2% drop in pitch reconstruction performance, as well as a marginal 0.1% improvement in duration MRR. The former is consistent with the fact that fingerings have a direct impact on pitch, whilst the latter suggests that physical constraints might shape music as performed by humans.

Nevertheless, structured representations for pitch height and pitch class could still be seen in the encoder latent space when visualised using a uniform manifold approximation and projection (UMAP) [29] for dimension reduction (Figure 5). By contrast, the encoder latent space did not seem to separate the different fingerings (here we only show finger placement) into such clear clusters. That is expected as label information was only fed into the classifier and was thus only implicit in the encoder.

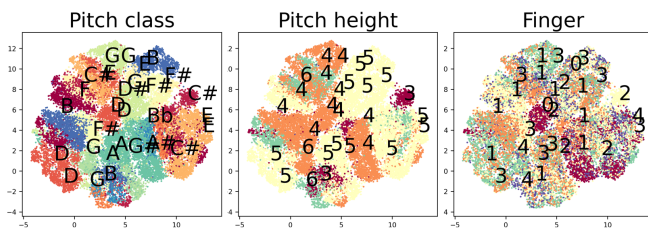


Figure 5. Visualising the encoder latent space with UMAP.

4.4 Semi-supervised learning

To investigate the effects of semi-supervised learning, we trained our model using six randomly selected excerpts as unlabelled data (i.e., without fingerings) and varied the number of labelled excerpts from one to seven. As control, we trained our model on the same labelled excerpts only. If our model could learn from unlabelled data, we would expect better generated fingerings. This is indeed what we found: our model trained on labelled and unlabelled data showed improved performance in all metrics except for the hand position F1 score (Figure 3, Supplementary Table 1). As expected, we also noticed a gradual improvement in performance as the number of labelled excerpts increased.

We further tested for significant effects of semi-supervised learning using random-intercept linear mixed models. The interaction between semi-supervision (labelled only vs. labelled and unlabelled) and number of labelled excerpts, as well as lower order terms were entered as fixed effects. Significant interactions for string MRR, nDCG@1, and nDCG@3 revealed substantial improvements ($\approx 11\%$) under semi-supervision when the ratio between labelled and unlabelled data was 1 : 6. That

is helpful, given our sampling (see Section 1) showed that only 15% of violin sheet music contained fingering information. Echoing the above, significant main effects of unlabelled data (with a mean improvement of around 3-6%) were also detected in MRR, nDCG@1, and nDCG@3 for string, position, and finger, as well as F1 scores for string and position (Table 2).

Finally, we note in Table 1 that our model already exceeded previous work [7] in string F1 performance when trained on four labelled plus six unlabelled excerpts, and achieved comparable performance in other metrics with seven plus six unlabelled excerpts. This demonstrates our model’s ability to make use of unlabelled data to improve fingering generation performance to match the state of the art model with half the amount of labelled data.

Main effect of semi-supervised learning			
		F(1,169)	p
MRR	String	10.05	.00181 **
	Position	11.47	.000879 ***
	Finger	14.06	.000243 ***
F1	String	6.00	.0153 *
	Position	0.28	.597
	Finger	11.32	.000948 ***
nDCG@1	String	12.45	.000537 ***
	Position	7.15	.00825 **
	Finger	12.19	.000614 ***
nDCG@3	String	10.76	.00126 **
	Position	13.57	.000309 ***
	Finger	16.53	7.34×10^{-5} ***

Table 2. Linear mixed model analyses revealed significant performance improvements in all except one metric when our model was trained under semi-supervision. See Supplementary Table 3 for significance of other factors.

5. CONCLUSION AND FUTURE WORK

In this paper, we presented a semi-supervised model that generates violin fingerings from the musical score. Our approach leverages the generative ability of variational autoencoders and reframes fingering generation as an additional latent variable for learning pitch and timing reconstructions from unlabelled data. We demonstrated that our model better replicated the fingering style of a human performer and generated fingerings that were more preferred amongst violinists when trained on both labelled and unlabelled data. Our method can be readily adapted to fingering generation in other instruments such as piano and guitar, which also suffer from the same lack of labelled data [8]. Another possibility is to extend our model with heuristic rules to tailor generated fingerings for different playing styles or groups (e.g., pedagogy for violinists at different skill levels) [2, 3, 7]. Lastly, that pitch reconstruction depended on fingering information also highlights the importance of physical constraints and playability in music performed by humans. Such aspects are often overlooked, but should be explored in future machine-based music generation models if a more human-like quality is desired.

6. ACKNOWLEDGEMENTS

This work was partially supported by MOST Taiwan under the project *Content Generation and Interaction Systems for Virtual Musicians* (Grant No. 109-2221-E-001-018-MY3).

7. REFERENCES

- [1] A. Maezawa, K. Itoyama, T. Takahashi, K. Komatani, T. Ogata, and H. G. Okuno, "Violin fingering estimation based on violin pedagogical fingering model constrained by bowed sequence estimation from audio input," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2010, pp. 249–259.
- [2] W. Nagata, S. Sako, and T. Kitamura, "Violin fingering estimation according to skill level based on hidden markov model." in *ICMC*, 2014.
- [3] A. Maezawa, K. Itoyama, K. Komatani, T. Ogata, and H. G. Okuno, "Automated violin fingering transcription through analysis of an audio recording," *Computer Music Journal*, vol. 36, no. 3, pp. 57–72, 2012.
- [4] P. Barbieri and S. Mangsen, "Violin intonation: a historical survey," *Early music*, vol. 19, no. 1, pp. 69–88, 1991.
- [5] D. Huron and C. Trevor, "Are stopped strings preferred in sad music?" *Empirical Musicology Review*, vol. 11, no. 2, pp. 261–269, 2017.
- [6] M. A. Winget, "Annotations on musical scores by performing musicians: Collaborative models, interactive methods, and music digital library tool development," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 12, pp. 1878–1897, 2008.
- [7] Y.-H. Jen, T.-P. Chen, S.-W. Sun, and L. Su, "Positioning left-hand movement in violin performance: A system and user study of fingering pattern generation," in *26th International Conference on Intelligent User Interfaces*, 2021, pp. 208–212.
- [8] A. Wiggins and Y. Kim, "Guitar tablature estimation with a convolutional neural network." in *ISMIR*, 2019, pp. 284–291.
- [9] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *CoRR*, 2014.
- [10] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*. PMLR, 2014, pp. 1278–1286.
- [11] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," 2014.
- [12] G. Hori and S. Sagayama, "Minimax viterbi algorithm for hmm-based guitar fingering decision." in *ISMIR*, 2016, pp. 448–453.
- [13] S. I. Sayegh, "Fingering for string instruments with the optimum path paradigm," *Computer Music Journal*, vol. 13, no. 3, pp. 76–84, 1989.
- [14] A. Al Kasimi, E. Nichols, and C. Raphael, "A simple algorithm for automatic generation of polyphonic piano fingerings": 8th international conference on music information retrieval," 2007.
- [15] M. Balliauw, D. Herremans, D. Palhazi Cuervo, and K. Sørensen, "A variable neighborhood search algorithm to generate piano fingerings for polyphonic sheet music," *International Transactions in Operational Research*, vol. 24, no. 3, pp. 509–535, 2017.
- [16] "Statistical learning and estimation of piano fingering," *Information Sciences*, vol. 517, pp. 68–85, 2020.
- [17] A. Maezawa, K. Itoyama, T. Takahashi, T. Ogata, and H. G. Okuno, "Bowed string sequence estimation of a violin based on adaptive audio signal classification and context-dependent error correction," in *2009 11th IEEE International Symposium on Multimedia*. IEEE, 2009, pp. 9–16.
- [18] S. Sako, W. Nagata, and T. Kitamura, "Violin fingering estimation according to the performer's skill level based on conditional random field," in *International Conference on Human-Computer Interaction*. Springer, 2015, pp. 485–494.
- [19] C. Doersch, "Tutorial on variational autoencoders," *arXiv preprint arXiv:1606.05908*, 2016.
- [20] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.
- [21] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β -vae," *arXiv preprint arXiv:1804.03599*, 2018.
- [22] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *ICLR*.
- [23] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables."
- [24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.
- [25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

- [26] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.
- [27] N. Craswell, *Mean Reciprocal Rank*. Boston, MA: Springer US, 2009, pp. 1703–1703.
- [28] K. Järvelin and J. Kekäläinen, “Cumulated gain-based evaluation of ir techniques,” *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.
- [29] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.

LISTEN, READ, AND IDENTIFY: MULTIMODAL SINGING LANGUAGE IDENTIFICATION OF MUSIC

Keunwoo Choi and Yuxuan Wang

ByteDance

{keunwoo.choi, wangyuxuan.11}@bytedance.com

ABSTRACT

We propose a multimodal singing language classification model that uses both audio content and textual metadata. LRID-Net, the proposed model, takes an audio signal and a language probability vector estimated from the metadata and outputs the probabilities of the target languages. Optionally, LRID-Net is facilitated with modality dropouts to handle a missing modality. In the experiment, we trained several LRID-Nets with varying modality dropout configuration and tested them with various combinations of input modalities. The experiment results demonstrate that using multimodal input improves performance. The results also suggest that adopting modality dropout does not degrade the performance of the model when there are full modality inputs while enabling the model to handle missing modality cases to some extent.

1. INTRODUCTION

Singing language identification of music (SLID) is a classification task of labeling tracks by the languages used in lyrics. Language information is essential for music discovery and recommendation systems as lyrics play a crucial role in the music listening experience [1]. In reality, despite its importance, language information is not always available or accurate, even for established, large-scale music streaming services. Such a situation has motivated researchers to develop language identification models using the most fundamental music data – the audio content itself.

There have been various audio-based approaches in SLID. Most of them utilize traditional machine learning classifiers and audio features, similar to early methods for spoken language identification [2, 3]. The models in [4] and [5] consist of vocal/non-vocal segmentation, feature vector quantization, and a simple codebook-based language model. More recently, *i*-vector [6], a popular feature vector for speech-related tasks, was used in [7, 8], combined with a support vector machine classifier. A vocal source separation technique that is based on a spatial property of stereo music signals was added in a model [9], although it did not improve its performance in the experi-

ment. More recently, the model in [10] uses a modern vocal separation technique [11] and a one-dimensional deep convolutional neural network. In these approaches, a vocal source separation module is applied to input signals to extract relevant features with a cleaner signal, i.e., with a less amount of accompaniments.

There also have been models that are based on non-audio modalities. The model in [12] is designed to classify music videos into language categories by taking visual features such as histograms of oriented gradients along with basic audio features such as MFCCs. In the experiment, adding video features improved the accuracy of the model from 44.7% to 47.8% in a 25-language classification task. Another model in [13] uses language estimation of track title and album name and showed a comparable performance to their in-house audio-based classifier. Notably, an internal music representation called track vector, which is estimated using music listening history data, showed the highest feature relevance – 0.97 – in their experiment. It re-emphasizes that there is a strong connection between music listening preference and singing language of music.

It is noteworthy that unfortunately, none of the mentioned works is reproducible [4, 5, 7–10, 12, 13] and there has not been any benchmark in SLID. All the previous works rely on private datasets and only little details such as target languages and the number of tracks are known. This is because of a lack of datasets – there has not been a publicly available music language classification dataset until very recently [14]. Some music tagging datasets may be considered as alternatives if they include language tags; for example, the million song dataset has language labels [15]. But their tag popularity is merely at 116th (‘german’), 135th (‘english’), or below, which are often excluded in a prevalent problem formulation such as top-50 classification [16] to suppress noise during training and evaluation [17].

In this paper, we introduce LRID-Net – **Listen, Read, and Identify-Network**, a model that takes audio and textual metadata (track title, album name, and artist name) to identify singing language. LRID-Net is based on a combination of a deep convolutional neural network, MLPs, and modality dropouts as explained in Section 4. We provide brief information about Music4All [14], the dataset that we use, in Section 3. In Section 5, we present our experiment results including the performance of LRID-Net, a comparison of modalities, and verification of modality dropout.



There are three main contributions of our work.

Contribution 1: LRID-Net is the first work in SLID that takes advantage of audio and text inputs, presumably the two most accessible forms of music data.

Contribution 2: We present the first reproducible work in SLID by using a public dataset. This enables a rigorous benchmark to be followed, which is necessary for progress in modern machine learning research.

Contribution 3: LRID-Net has flexibility in its input data form – it is designed to work with missing modalities by adopting *modality dropouts*. This flexibility makes LRID-Net a highly pragmatic solution in a real-world scenario.

2. PROBLEM FORMULATION

We define our problem as a *singing language identification using audio content and metadata*. It is reasonable to expect that in many practical use-cases, (some of) three selected types of metadata – track title, album name, and artist name – would be accessible. For example, music tracks shared on online streaming services usually include all of them. We exclude other types of data such as visual features and pre-computed track vectors despite their benefits shown in [12] and [13], respectively, because their availability is limited for a subset of commercial music tracks even for those in industry who have an access to a large-scale proprietary catalog.

To make our model even more practical, we consider a missing data scenario. Some or all of the metadata can be easily missing. For example, indie music tracks shared online (e.g., SoundCloud or Jamendo) usually do not include any album name, or they can exist but should be considered missing since titles can be blank or consists of numbers and/or special characters only, not providing any linguistic information. The audio could be also missing or not helpful at all, for example, a segment that is input to the model may not contain any vocal part.

From a machine learning point of view, our SLID problem is a single-label multi-class classification with a multi-modal, potentially partially missing input. In fact, some lyrics are multilingual. But we assume that those cases are negligible, following the dataset we use (see Section 3 for more details).¹

3. DATASET

We use Music4All dataset [14] which includes 30-second audio clips (44,100 kHz and stereo), lyrics, and 16 other metadata such as title, album name, artist name, and Spotify identifier of 109,269 tracks. The dataset also includes language labels covering 46 languages. They are estimated from the lyrics using Langdetect,² a Python implementation of Language-Detection [18].³

¹ English words such as “Yeah”, “Hey”, and “Baby” are often used as musical expressions or interjections in non-English songs and we do not assume their existence makes a lyric multilingual.

² <https://pypi.org/project/langdetect/>

³ We noticed there are errors in the ground truth. But Langdetect is known to perform at 99% accuracy with documents, which is significant higher than the performance of the proposed audio-based models.

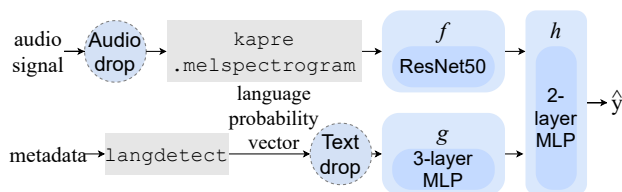


Figure 1: Block diagram of LRID-Net. Gray rectangular boxes indicate non-trainable modules and dotted circles are modality dropouts (Section 4.4), which are optionally applied in some of the experiments (Section 5.4).

The distribution of language labels in Music4ALL is heavily skewed from 84,103 items (English) to 1 item (Hindi, Slovak, Bulgarian, and Hebrew). We consider 11 labels – top-10 popular languages and an “others” category that includes all the other languages. In detail, there are 84,103 English tracks followed by Portuguese (7,020), Spanish (3,225), Korean (1,145), Others (1,059), French (994), Japanese (615), German (577), Polish (446), Italian (437), and Slovakian (231). There are also 9,417 instrumental tracks, but we exclude them because otherwise, the model would need to learn to perform language identification as well as instrumental track classification, which would distract our analysis.

There does not exist an official training-validation split of tracks of M4A dataset. We use an 80:20 stratified split with allocating every artist in only one of the split sets.⁴ This prevents artist-dependent information from being a confounding factor and leaking the information between training and validation sessions.

4. LRID-NET

4.1 Input Audio Preprocessing

The 44.1 kHz sampled 30-second stereo audio signals are downmixed and resampled to 22,050 Hz and converted into 128-bin log-magnitude mel-spectrograms (128 × 2580) with 1024-point FFT and 256 hop size using Kapre [19]. We call this spectrogram x_{audio} .

4.2 Input Text Preprocessing

The metadata strings are joined in an order of artist name, album name, and track title and then input to Langdetect to estimate a language probability vector. Originally, `langdetect.detect_langs()` outputs a probability distribution of 55 supported languages. However, there are some cases where the function fails to estimate probability, e.g., if the text is blank or a numeric value only. We add another dimension to indicate those exceptions. This 56-dimensional vector is called x_{lang} .

4.3 Overall model structure

LRID-Net consists of two input branches that are concatenated at a late stage of the network.

⁴ https://github.com/keunwoochoi/music4all_contrib

The audio branch, f , is a ResNet-50 with 64 base channels and outputs a 2048-channel feature map that is a size of (4×81) [20]. We choose ResNet-50 for its simplicity yet strong performance as shown in [21] for music tagging. Then, a global average pooling is applied to output a one-dimensional vector length of 2048. This procedure is represented as $s_{\text{audio}} = f(x_{\text{audio}})$.

The text branch, g , is a 3-layer MLP where each layer consists of a 128-unit fully-connected layer, a batch normalization layer, and a ReLU activation [22], i.e., $s_{\text{text}} = g(x_{\text{lang}})$.

In the output branch, h , the two outputs of the input branches are concatenated, i.e., $s_{\text{cat}} = [s_{\text{audio}}; s_{\text{text}}]$ and input to an MLP. This MLP consists of a 256-unit fully-connected hidden layer, a batch normalization layer, a ReLU activation, and an 11-unit fully-connected layer with a Softmax activation to output the language probability, i.e., $\hat{y} = h(s_{\text{cat}})$.

4.4 Modality Dropout

We use *modality dropout* which was originally introduced in [23] as *ModDrop* and is illustrated in the block diagram in Figure 1. Similar to the original dropout [23], during training time, a modality dropout module replaces its input with zeros with a probability of r , the dropout rate. There are two main differences between the original dropout and modality dropout. 1) The original dropout is applied to a part (e.g., a single node or a channel) of an input but a modality dropout module drops the whole input of a modality, i.e., an audio signal or a language probability vector. By doing so, it effectively simulates a missing modality input and lets the model learn to perform the task without the dropped input. 2) There is no $1/(1-r)$ scaling in a modality dropout when the input is not dropped. During test time, a system with LRID-Net inputs a zero vector to the model if a modality is missing.

5. EXPERIMENT AND DISCUSSION

We performed a series of experiments to demonstrate the performance and properties of LRID-Net. We use Music4All dataset [14] and process audio and text data using Kapre [19] and Langdetect [18] as detailed in Section 3 and Section 4, respectively. During training, we use Adam optimizer [24] and early stopping with a patience of 20 epochs. We do not adopt any balancing during batching and loss computation.

On the metric, we use F1-score, precision, and recall. As described in Section 3, there is a high imbalance of the number of data points of each language in the dataset. In this case, from a user perspective, a macro average can be used to represent the class-balanced performance to avoid the bias towards popular languages, too. However, *because* they are biased in the *same way* in the training and validation sets, weighted (or micro) averaging can be considered to be more suitable than macro averaging on representing how successfully the model was trained to minimize the empirical loss. Acknowledging this issue, we use both of

Text input	Precision	Recall	F1-Score
Artist Name	.323	.221	.149
Album Name	.399	.378	.284
Track Title	.450	.444	.317
Joining All	.510	.569	.429

Table 1: The macro averaged performance of Langdetect prediction with various text inputs. ‘Joining All’ represents the performance of langdetect baseline.

Text input	Precision	Recall	F1-Score
Artist Name	.600	.368	.456
Album Name	.750	.576	.652
Track Title	.766	.573	.656
Joining All	.922	.819	.857

Table 2: The weighted averaged performance of Langdetect prediction with various text inputs. ‘Joining All’ represents the performance of Langdetect baseline.

the averaging methods and focus on the performance with a suitable one depending on the context.

We present the performance of each language sorted by its occurrence count, as known as Support, to help understanding of any related trend.

5.1 Langdetect baseline

We present the result of a simple solution that is to directly use the top prediction of text-based language identification using Langdetect. This baseline approach is called *Langdetect baseline*. We also present a detailed analysis of the performance and behavior of Langdetect baseline model to deepen our understanding of the problem and the following experiment results.

Table 1 and 2 summarize the performance of Langdetect baseline based on various text inputs, showing that using all the metadata (‘Joining All’) performs the best by achieving an F1-score of 0.429 (macro average) or 0.857 (weighted average). Details of the performance with ‘Joining All’ are presented in Table 3 and its confusion matrix is illustrated in Figure 2. Note that Langdetect baseline model is not trained with our dataset, hence support (number of true items) does not affect the performance.

The F1-scores seem under-performing since Langdetect is reported to show 99% accuracy. We conjecture two reasons for this result. First, music metadata is significantly shorter than typical documents and news article, with which Langdetect was originally trained and tested, respectively. Second, there is a prevailing usage of English for artist name, album name, and track title even if the lyrics are not written in English, especially for those songs that are internationally consumed.

In detail, the precision and recall shows interesting patterns that are partly related to the second reason; There are languages where precision is significantly higher than recall (Group 1: Korean, Japanese) while precision is significantly lower than recall for some other languages (Group 2: French, German, Italian, Slovakian). We note that

Language	Precision	Recall	F1-Score	Support
English	.969	.859	.911	84,103
Portuguese	.897	.695	.783	7,020
Spanish	.676	.620	.647	3,225
Korean	.444	.003	.007	1,145
Others	.092	.700	.162	1,059
French	.367	.585	.452	994
Japanese	.895	.153	.261	615
German	.093	.773	.166	577
Polish	.695	.572	.627	446
Italian	.229	.748	.350	437
Slovakian	.252	.554	.347	231

Table 3: The performance of Langdetect prediction when all the metadata are concatenated ('Joining all'). Support column indicates the number of items in the training set, of which distribution is preserved in the validation set.

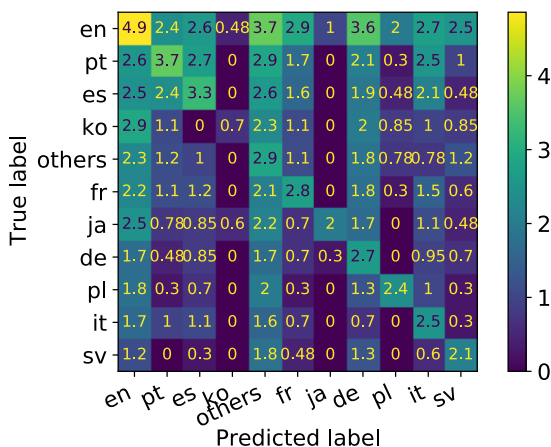


Figure 2: The confusion matrix of Langdetect baseline model ('Joining all'). Item counts are converted by $\log_{10}(x + 1)$.

the languages in Group 1 are only two non-European languages and suggest two explanations for their patterns. First, on high precision scores, since Korean and Japanese use completely different letter systems compared to other languages, it is very easy for Langdetect to recognize them if the metadata is written in Korean or Japanese. Second, the low recall may come from the common usage of English as mentioned earlier. In such cases, metadata-based Langdetect would never be able to correctly predict that the lyrics are written in Korean or Japanese. This is revealed in a deeper dataset analysis. Among the 1,145 Korean songs in the training set, there are 244 unique artists, out of which 241 artist names are English. Conversely, the model almost never misclassifies non-Korean or non-Japanese songs to Korean or Japanese, respectively. Unlike Group 1, we did not find any convincing explanation for the patterns of Group 2.

5.2 Single modality baselines

As additional baseline models, we show experiment results of single modality models. They are AO (audio-only)

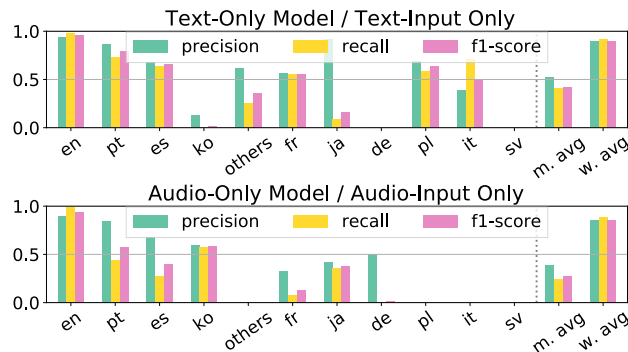


Figure 3: Top - The performance of an text-only model (TO model). Its precision, recall, and F1-score are .526 / .415 / .422 (macro averaging) and .896 / .914 / .900 (weighted averaging). Bottom - The performance of an audio-only model (AO model). The precision, recall, and F1-score are respectively .387 / .248 / .275 (macro averaging) and .852 / .884 / .857 (weighted averaging).

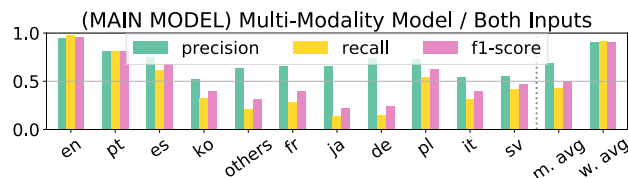


Figure 4: The performance of Main model with both modality inputs. The precision, recall, and F1-score are respectively .688 / .435 / .504 (macro averaging) and .911 / .922 / .911 (weighted averaging).

model and TO (text-only) model.

First of all, we compare the TO model against Langdetect baseline. TO model achieves a comparable macro averaged F1-score (0.422, a degradation of 0.007) and a higher weighted averaged F1-score (0.900, an improvement of 0.043) as in Figure 3. Again, this result seems heavily affected by the class imbalance of the training set.

Second, as illustrated in Figure 3, a lack of a modality leads to negative effects, often critically to some languages. AO model completely fails at identifying Others, Polish, Italian, and Slovakian while TO model fails at Korean, Japanese, German, and Slovakian.

Third, as summarized in Figure 3, in every metric and averaging strategy, TO model outperformed AO model, showing the importance of using metadata. However, this does not mean audio is less useful than textual data. Acknowledging the limit of the information in the metadata discussed with Langdetect baseline in Section 5.1, the result may indicate the opposite that currently, the information from text input is almost saturated and more improvement should be based on a better audio understanding.

5.3 LRID-Net: Main Model

In this section, we introduce the experiment result of our multimodal SLID model, LRID-Net. This LRID-Net net was trained without any modality dropouts and we call this model 'Main model'.

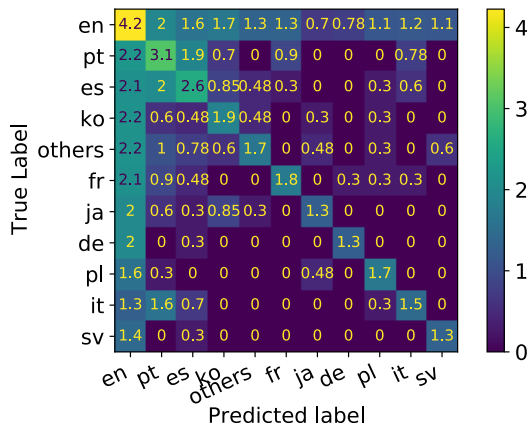


Figure 5: The confusion matrix of Main model. Item counts are converted by $\log_{10}(x + 1)$.

Main model shows an improvement over all the aforementioned baseline models. Compared to Langdetect baseline, it achieves a higher F1-score (+0.048) with a higher precision (+0.134) and a lower recall (-0.118) on weighted average. It also achieves a higher F1-score (+0.075) and a higher precision (+0.158), but a lower recall (-0.134) on macro average. Main model also outperforms AO model and TO model, the single modality models, in every metric and averaging strategy, emphasizing the benefit of using multimodal information.

Among languages, Main model shows low recall rates for Korean, French, Japanese, German, and Italian as shown in Figure 4. This pattern is similar to that in the results of Langdetect baseline as discussed in Section 5.1. We conjecture that a similar type of confusion may have happened in Main model, especially if the 56-dimensional text-based language probability input computed with Langdetect plays an important role (which seems true given that TO model outperforms AO model in Section 5.2). The class imbalance of the training set seems to penalize recall rates of those languages because a classification of unconfident items is likely to be biased towards the mode of the distribution of training items, i.e., English. This is shown in Figure 5.

The overall improvement of F1-score did not benefit all the languages equally. The performance of Main model is rather polarized than Langdetect baseline model. The F1-scores of 3/4 popular languages (English, Spanish, and Korean) and Others category are improved in Main model compared to Langdetect baseline. In the meantime, out of the six less popular languages, only two languages (Italian and Slovakian) show an improvement. This might mean a correlation between the performance and the number of training data. However, there is a clear exception. Main model achieves an F1-score of 0.347 for Slovakian, which both AO model and TO model completely failed.

For some languages, Main model achieved a lower performance than a single modality model, leaving room for further improvement. For example, Main model is outperformed by AO model for Korean and TO model for French and Italian.

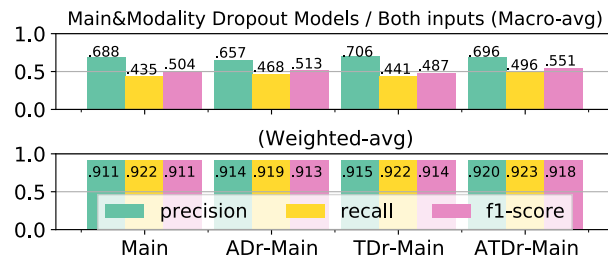


Figure 6: The comparison of metrics of Main model and models with various modality dropout strategies when there is no missing modalities in the data. ADr-Main, TDr-Main, and ATDr-Main indicate models with the same architecture but with modality dropouts applied on audio input, text input, and both of the inputs, respectively.

5.4 Modality Dropout

In this section, we discuss the benefits and effects of applying modality dropout to Main model. We test three modality dropout strategies: applying modality dropout to audio input only (ADr-Main model), text input only (TDr-Main model), and both of the inputs (ATDr-Main model). The dropout rate is fixed to 0.2 for every model.⁵ In ATDr-Main model, the two dropouts work independently. This means that during training, stochastically, 4% of training items would have zero values for both of the inputs where backpropagation is still applied to update the model. This leads to strengthen the model to predict the distribution of the training data and may not be ideal, but we did not observe any critical issue in practice.

Since there is no language-specific pattern, we present the averaged metrics only in this section.

5.4.1 Case 1: Complete Modality – Do modality dropouts have any negative affects?

It is unusual to use modality dropout in music information retrieval. We first investigate to ensure that adopting it does not harm the normal use-cases, i.e., when there is no missing modality.

As presented in Figure 6, all the models with modality dropouts - ADr-Main model, TDr-Main model, and ATDr-Main model achieve comparable or even outperforming performances over Main model although they might not be statistically significant. The effects of Modality dropout are better reflected on weighted-average scores than macro-average ones. That is because weighted-averaged scores are more linearly related to the empirical loss that our models are trained to minimize. As shown, modality dropouts only improved those scores. To summarize, adopting modality dropouts does not harm the normal use-cases.

5.4.2 Case 2: Missing Audio Input

Figure 7 presents the performances of Main model, ADr-Main model, and ATDr-Main model when audio inputs are missing, i.e., there is only text input.

⁵ 0.2 was chosen assuming a small portion (for example, 20%) of tracks would have missing modality in the real use-cases.

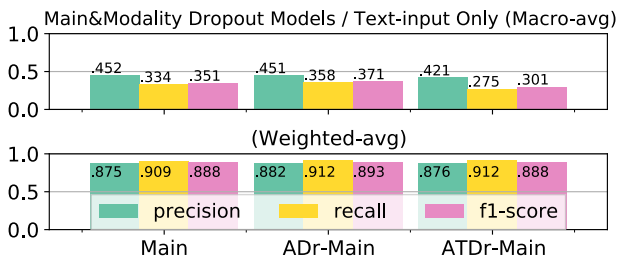


Figure 7: The comparison of metrics of Main model and models with various modality dropout strategies (see 6 for details) when there is missing audio modality. Note that as in Figure 3 (top), TO model, a dedicated text-only model achieved precision / recall / F1-score of .526 / .415 / .422 (macro averaging) and .896 / .914 / .900 (weighted averaging), respectively.

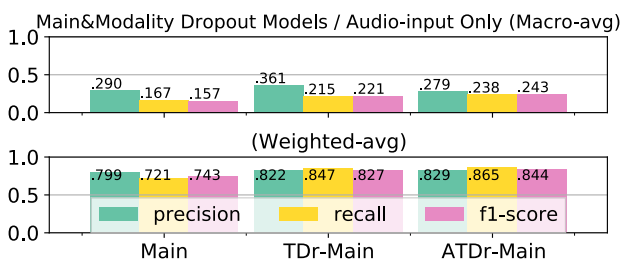


Figure 8: The comparison of metrics of Main model and models with various modality dropout strategies (see 6 for details) when there is missing text modality. Note that as in Figure 3 (bottom), AO model, a dedicated audio-only model achieved precision / recall / F1-score of .387 / .248 / .275 (macro averaging) and .852 / .884 / .857 (weighted averaging), respectively.

First, surprisingly, Main model - one that is trained without any modality dropout - performs comparably with ADr-Main model and ATDr-Main model, which are trained explicitly to be able to deal with missing audio input. This is an unexpected behavior since there is no reason the model should learn to ignore a zero audio input (silence) and make a correct prediction solely based on the text information – but, seems like it does. We find it difficult to explain it and leave it as a future work.

Second, it is worth comparing these results with that of TO model (Figure 3, top), a model that is designed and trained to work with text input only. In all the metrics and averaging strategies, TO model outperforms all the three models. For F1-score, even with the best model (ADr-Main), there is a difference of 0.051 (macro averaging) or 0.007 (weighted averaging). This indicates that despite versatility, a model with an audio modality dropout may not completely replace a dedicated text-only model, especially if missing audio inputs are highly likely.

5.4.3 Case 3: Missing Text Input

Figure 8 presents the performance of Main model, TDr-Main model, and ATDr-Main model when text input is missing, i.e., there is only audio input.

First, there are noticeable improvements by apply-

ing modality dropouts - TDr-Main model and ATDr-Main model outperformed Main model for the most of the metrics. ATDr-Main model achieved +0.086 and +0.101 higher F1-scores than Main model does. This result supports a potential real-world use-case of serving a single SLID model where metadata may or may not be available.

Second, when compared to AO model (Figure 3, top), all the three models in Figure 8 are outperformed. This means, similar to the conclusion of Section 5.4.2, a model with a text modality dropout may not serve as a perfect alternative and whether to apply text modality dropout (as opposed to train two different models, AO model and Main model) would be a practical choice: the decision would be based on the ratio of missing-text inputs and the costs of training and maintaining one vs. two models.

6. CONCLUSION

In this paper, we presented LRID-Net, a deep learning model for singing language identification (SLID) that takes advantage of multimodal data. LRID-Net takes an audio input as well as a text input that combines track title, album name, and artist name. We also propose modality dropout in MIR task, which is designed to let a single model be used with varying input availability. In the experiment, we showed that i) multimodal input improves the performance, ii) a language probability vector of metadata is an effective representation for SLID, iii) modality dropouts do not harm the performance when both of the input modalities exist, and iv) modality dropouts make a model robust with missing input to some extent.

Our research has several limitations. There are some behaviors that we could not provide a satisfying explanation about. Although being useful to some extent, the modality dropout did not completely fulfill the need of building multiple models to cope with missing modalities. Due to the already complicated experiment configuration, we did not opt for balancing the languages, which would be a necessary step to build a more practical language classifier.

There are many research questions to be answered in SLID. A data-driven SLID model might learn some non-linguistic features that are correlated to language labels, and identifying those mechanisms would lead to building more robust SLID models. One approach to demystifying their behavior is to use source separation techniques and observe how much a model is relying on vocal parts vs accompaniments. Source separation techniques also could lead to a better performing model because separated signals would provide a disentangled input representation that may be useful for the task. Finally, another highly related and interesting task is lyric transcription, which can be combined with SLID, where a mutual benefit is anticipated.

7. ACKNOWLEDGEMENT

We thank Jeong Choi, Minz Won, Jordan Smith, Janne Spijkervet, Gianluca Micchi, and Zhihao Ouyang for their helpful reviews and discussion on this paper.

8. REFERENCES

- [1] J. H. Lee and J. S. Downie, "Survey of music information needs, uses, and seeking behaviours: preliminary findings." in *The 5th International Society of Music Information Retrieval Conference (ISMIR)*, vol. 2004. Citeseer, 2004, p. 5th.
- [2] M. Sugiyama, "Automatic language recognition using acoustic features," in *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 1991, pp. 813–816.
- [3] L. F. Lamel and J.-L. Gauvain, "Language identification using phone-based acoustic likelihoods," in *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 1. IEEE, 1994, pp. 1–293.
- [4] W.-H. Tsai, H.-M. Wang *et al.*, "Towards automatic identification of singing language in popular music recordings." in *International Society of Music Information Retrieval (ISMIR)*. Citeseer, 2004.
- [5] W.-H. Tsai and H.-M. Wang, "Automatic identification of the sung language in popular music recordings," *Journal of New Music Research*, vol. 36, no. 2, pp. 105–114, 2007.
- [6] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2010.
- [7] A. M. Kruspe and I. Fraunhofer, "Improving singing language identification through i-vector extraction." in *DAFx*, 2014, pp. 227–233.
- [8] A. M. Kruspe, J. Abesser, and C. Dittmar, "A gmm approach to singing language identification," in *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.
- [9] J. Schwenninger, R. Brueckner, D. Willett, and M. E. Hennecke, "Language identification in vocal music." in *International Society of Music Information Retrieval (ISMIR)*, 2006, pp. 377–379.
- [10] D. DEL CASTILLO, "End-to-end learning for singing-language identification," 2020, master's thesis, KTH, School of Electrical Engineering and Computer Science (EECS).
- [11] K. Drossos, S. I. Mimilakis, D. Serdyuk, G. Schuller, T. Virtanen, and Y. Bengio, "Mad twinnet: Masker-denoiser architecture with twin networks for monaural sound source separation," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.
- [12] V. Chandrasekhar, M. E. Sargin, and D. A. Ross, "Automatic language identification in music videos with low level audio and visual features," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5724–5727.
- [13] L. Roxbergh, "Language classification of music using metadata," 2019, master's thesis, Uppsala Universitet.
- [14] I. A. P. Santana, F. Pinhelli, J. Donini, L. Catharin, R. B. Mangolin, V. D. Feltrim, M. A. Domingues *et al.*, "Music4all: A new music database and its applications," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. IEEE, 2020, pp. 399–404.
- [15] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.
- [16] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," *The 17th International Society of Music Information Retrieval Conference, New York, USA*, 2016.
- [17] K. Choi, G. Fazekas, K. Cho, and M. Sandler, "The effects of noisy labels on deep convolutional neural networks for music tagging," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 139–149, 2018.
- [18] N. Shuyo, "Language detection library for java," 2010. [Online]. Available: <http://code.google.com/p/language-detection/>
- [19] K. Choi, D. Joo, and J. Kim, "Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras," in *Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning*. ICML, 2017.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [21] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," *Proceedings of 17th Sound and Music Computing (SMC)*, 2020.
- [22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
- [23] N. Neverova, C. Wolf, G. Taylor, and F. Nebout, "Mod-drop: adaptive multi-modal gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 8, pp. 1692–1706, 2015.
- [24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *3rd International Conference for Learning Representations, San Diego*, 2015.

ON PERCEIVED EMOTION IN EXPRESSIVE PIANO PERFORMANCE: FURTHER EXPERIMENTAL EVIDENCE FOR THE RELEVANCE OF MID-LEVEL PERCEPTUAL FEATURES

Shreyan Chowdhury¹ Gerhard Widmer^{1,2}

¹Institute of Computational Perception, Johannes Kepler University Linz, Austria

²LIT AI Lab, Linz Institute of Technology, Austria

firstname.lastname@jku.at

ABSTRACT

Despite recent advances in audio content-based music emotion recognition, a question that remains to be explored is whether an algorithm can reliably discern emotional or expressive qualities between different performances of the same piece. In the present work, we analyze several sets of features on their effectiveness in predicting arousal and valence of six different performances (by six famous pianists) of Bach’s Well-Tempered Clavier Book 1. These features include low-level acoustic features, score-based features, features extracted using a pre-trained emotion model, and Mid-level perceptual features. We compare their predictive power by evaluating them on several experiments designed to test performance-wise or piece-wise variations of emotion. We find that Mid-level features show significant contribution in performance-wise variation of both arousal and valence – even better than the pre-trained emotion model. Our findings add to the evidence of Mid-level perceptual features being an important representation of musical attributes for several tasks – specifically, in this case, for capturing the expressive aspects of music that manifest as perceived emotion of a musical performance.

1. INTRODUCTION

A musical performance, particularly in the Western music tradition, is not merely a literal acoustic rendering of a notated piece or composition. Rather, the piece is transformed by the performer’s own expressive performance choices, relating to such dimensions as the choice of tempo, expressive tempo and timing variations, dynamics, articulation, and so on. The emotional effect of a performance on a listener can be a consequence both of the composition itself, with its musical properties and structures, and of the performance, the way the piece was played. In fact, it has been convincingly demonstrated [1, 2] that performers are capable of communicating, with high accuracy,

intended emotional qualities by their playing.

The analysis of emotion in music recordings has a long history in Music Information Retrieval, with many works addressing content-based emotion regression and classification typically using low-level or hand-crafted audio and musical features [3–6] or using deep learning based methods [7–9]. However, there has been little research on the more subtle problem of identifying emotional aspects that are due to the actual *performance*, and even less on models that can automatically recognize this from *audio* recordings. On the latter problem – the one to be addressed in this paper – the most directly relevant prior work we are aware of is [10], where 324 6-second audio snippets of different genres (classical, jazz, blues, metal, etc.) were annotated in terms of perceived emotion (valence and arousal), and various regressors were trained to predict these two dimensions from a set of standard audio features. The regression models were then used to predict valence-arousal trajectories over 5 different recordings of 4 Chopin pieces, but no ground truth in terms of human emotion annotations was collected. The relevance of the model predictions was evaluated only indirectly, by comparing similarity scores between predicted profiles with overall performance similarity ratings by three human listeners, which showed some non-negligible correlations.

In a recent focused study [11], Battcock & Schutz (referred to as “B&S” henceforth) investigate how three specific score-based cues (Mode, Pitch Height, and Attack Rate¹) work together to convey emotion in J.S.Bach’s preludes and fugues collected in his *Well-tempered Clavier (WTC)*. They used recordings of the complete WTC Book 1 (48 pieces) of one famous pianist (Friedrich Gulda) as stimuli for human listeners to rate each performance on perceived arousal and valence. Their findings suggest that within this set of performances, arousal is significantly correlated with attack rate and valence is affected by both the attack rate and the mode. However, that study was based on only one set of performances, making it impossible to decide whether the human emotion ratings used as ground truth really reflect aspects of the compositions themselves, or whether they were also (or even predominantly) affected by the specific (and, in some cases, rather unconventional)



© S. Chowdhury, and G. Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** S. Chowdhury, and G. Widmer, “On Perceived Emotion in Expressive Piano Performance: Further Experimental Evidence for the Relevance of Mid-level Perceptual Features”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ Actually, attack rate as computed by B&S is also informed by the average tempo of the performance; thus, it is not strictly a score-only feature.

Pianist	Recording	Year
Glenn Gould	Sony 88725412692	1962-1965
Friedrich Gulda	MPS 0300650MSW	1972
Angela Hewitt	Hyperion 44291/4	1997-1999
Sviatoslav Richter	RCA 82876623152	1970
Andr�as Schiff	ECM 4764827	2011
Rosalyn Tureck	DG 4633052	1952-1953

Table 1: Pianists and recordings.

way in Friedrich Gulda plays the pieces – that is, whether the emotion ratings reflect piece or performance aspects.

The purpose of the present paper is to try to disentangle the possible contributions and roles of different features in capturing composer-(piece-)specific and performer-(recording-)specific aspects. To this end, we collected human ratings of perceived valence and arousal in six complete sets of recordings of WTC Book 1, and then performed a systematic study with feature sets derived from various levels of musical abstraction, including some extracted by pre-trained deep neural networks.

2. DATA COLLECTION

2.1 Pieces and Recordings

J.S.Bach’s *Well-tempered Clavier (WTC)* is ideally suited for systematic and controlled studies of this kind, as it comprises a stylistically coherent set of keyboard pieces from a particular period, evenly distributed over all keys and major/minor modes, with a pair of two pieces (a prelude, followed by a fugue) in each of the 24 possible keys, for a total of 48 pieces. Each piece has its own distinctive musical character, and despite being written in a rather strict style and not meant to be played in ‘romantic’ ways, the music offers pianists (or pianists take) lots of liberties in ornamentation, but also overall performance parameters (e.g., tempo and articulation). For example, there are pieces in our set of recordings that one pianist takes more than twice (!) as fast as another.

For a broad set of diverse performances, we selected six recordings of the complete WTC Book 1, by six famous and highly respected pianists, all of whom can be considered Bach specialists to various degrees. The recordings are listed in Table 1.

2.2 Emotion Annotations and Pre-processing

In accordance with B&S, we will only use the first 8 bars of each recording for the annotation process and our experiments. These were cut out manually. The participants of our annotation exercise were students of a course at a university, without a specifically musical background. Each participant heard a subset of the recordings (all 48 pieces as played by one pianist) and was asked to rate the valence on a scale of -5 to +5 (11 levels) and the arousal on a scale of 0 to 100 (increments of 10; a total of 11 levels). They could listen to a recording as many times as they liked. Each recording was rated by 29 participants. In total, we collected 8,352 valence-arousal annotation pairs.

For the purposes of this paper, we take the mean arousal and mean valence ratings for each recording, and these values serve as our ground-truth values for all following experiments. The distributions (over the 6 performances) of these mean ratings for each piece are summarised in Figure 1.

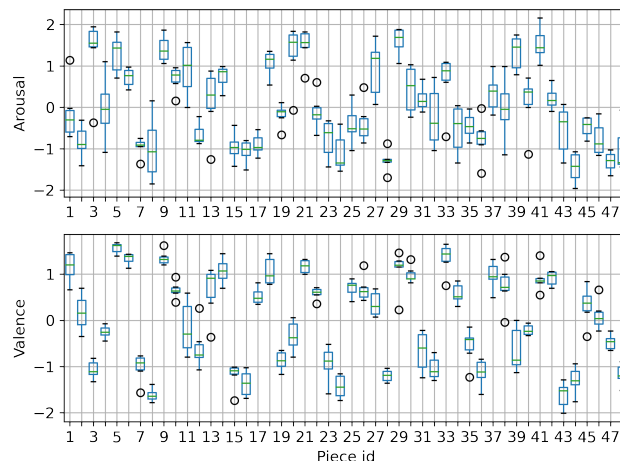


Figure 1: Distribution of emotion ratings across pianists for every piece.

3. FEATURE SETS

In this section, we briefly describe the four feature sets that we use to model arousal and valence.

3.1 Low-level Features

These consist of hand-crafted musical features (such as onset rate, tempo, pitch salience) as well as generic audio descriptors (such as spectral centroid, loudness). Taken together, they reflect several musical characteristics such as tone colour, dynamics, and rhythm. A brief description of all low-level features that we use is given in Table 2. We use Essentia [12] and Librosa [13] for extracting these. The audio is sampled at 44.1kHz and the spectra computed (when required) with a frame size of 1024 samples and a hop size of 512 samples. Each feature is aggregated over the entire duration of an audio clip by computing the mean and standard deviation over all the frames of the clip (a ‘clip’ being an 8 bar initial segment from a recording).

3.2 Score Features

The following set of features was computed directly from the musical score (i.e., sheet music) of the pieces instead of the audio files. The unit of score time, “beat”, is defined by the time signature of the piece (e.g., 4/4 means that there are 4 beats of duration 1 quarter in a bar). The score information and the audio files were linked using automatic score-to-performance alignment. Table 3 describes the score features in detail.

Dissonance	Total harmonic dissonance computed from pairwise dissonance of all spectral peaks.
Dynamic Complexity	The average absolute deviation from the global loudness level estimate in dB.
Loudness	Mean loudness of the signal computed from the signal amplitude.
Onset Rate	Number of onsets (note beginnings or transients) per second.
Pitch Saliency	A measure of tone sensation, computed from the harmonic content of the signal.
Spectral Centroid	The weighted mean frequency in the signal, with frequency magnitudes as the weights.
Spectral Flatness	A measure to quantify how much noise-like a sound is, as opposed to being tone-like.
Spectral Bandwidth	The second order bandwidth of the spectrum.
Spectral Rolloff	The frequency under which 85% of the total energy of the spectrum is contained.
Spectral Complexity	The number of peaks in the input spectrum.
Tempo (BPM)	Tempo estimate from audio in beats per minute.

Table 2: Low-level Features

Inter Onset Interval	The time interval between consecutive notes per beat.
Duration	Two features describing the empirical mean and standard deviation of the notated duration per beat in the snippet.
Onset Density	The number of note onsets per beat. A chord constitutes a single onset.
Pitch Density	The number of unique notes per beat.
Mode	Binary feature denoting major/minor modality, computed using the Krumhansl-Schmuckler key finding algorithm [14] (to reflect the fact that the dominant key over the segment may be different from the given key signature).
Key Strength	This feature represents how much does the tonality by the "Mode" feature fit the snippet.

Table 3: Score Features

3.3 Mid-level Features

Mid-level features, described in [15], are perceptual musical features that are intuitively understandable to the average listener. They seem well-suited to bridge the “semantic gap” between low-level audio features and high-level descriptors such as emotion and have been shown to be useful in explainable music emotion recognition [16]. We learn these features from the Mid-level Dataset [15] using a receptive-field regularised residual neural network (RF-ResNet) model [17]. Since we intend to use this model to extract features from solo piano recordings (a genre that is not covered by the original training data), we use a domain-adaptive training approach as described in [18]. We use an input audio length of 30 seconds, padded or cropped as required. As these features cannot be strictly defined, Table 4 lists a rough description adapted from the questions in [15] that were shown to the annotators of the dataset to help them rate the audio clips.

3.4 DEAMResNet Emotion Features

To compare the mid-level features with another deep neural network based feature extractor, we train a model with

Melodiousness	How singable is this music?
Articulation	Overall impression of articulation in terms of staccato or legato playing style. Higher means more staccato.
Rhythmic Stability	How easy is it to march-along with the music?
Rhythmic Complexity	How difficult is it to follow the music by tapping? Rhythmic layers and different meters correlate with higher complexity.
Dissonance	Noisier timbre or presence of dissonant intervals (tritones, seconds, etc.)
Tonal Stability	How clear or apparent the tonic and key are.
Minorness	Relates to the perceived tonality. More “minor-sounding” music will have higher minorness.

Table 4: Mid-level Features

the same architecture (RF-ResNet) and training strategy on the DEAM dataset [19] to predict arousal and valence from spectrogram inputs. Since this model is trained to predict arousal and valence, it is expected to learn representations suitable for this task. As with the mid-level model, we perform domain adaptation while training this model also.

Features are extracted from the penultimate layer of the model, which gives us 512 features. Since these are too many features to use for our dataset containing only 288 data points, we perform dimensionality reduction using PCA (Principal Component Analysis), to obtain 9 components explaining at least 98% of the variance. These 9 features are named as `pca_x` with `x` being the principal component number.

4. FEATURE EVALUATION EXPERIMENTS

In this section, we evaluate the four feature sets. The aim of this section is to answer the following questions:

1. How well can each feature set fit the arousal and valence ratings? How do these feature sets compare to the ones used by B&S? (Sections 4.1 and 4.2)
2. In each feature set, which features are the most important? (Section 4.3)
3. Which feature set best explains variation of arousal and valence *between pieces*? (Section 4.4)
4. Which feature set best explains variation of arousal and valence *between different performances of the same piece*? (Section 4.5)

We use ordinary least squares fitting on the dataset in question and calculate the regression metrics. The metrics we report are adjusted coefficient of determination (\tilde{R}^2), root mean squared error between true and predicted values (RMSE), and Pearson’s correlation coefficient between true and predicted values (Corr).

4.1 Evaluation on B&S Data

As a starting point, we take the data used by B&S in Experiment 3 of their paper – Gulda’s performances rated on valence and arousal. We perform regression with our feature sets and compare with the values obtained by B&S

using their features Attack Rate, Pitch Height, and Mode. The results are summarised in Table 5.

	Arousal			Valence		
	\tilde{R}^2	RMSE	Corr	\tilde{R}^2	RMSE	Corr
Mid-level	0.84	0.36	0.93	0.79	0.42	0.91
DEAMResNet	0.91	0.27	0.96	0.69	0.50	0.86
Low-level	0.86	0.29	0.96	0.67	0.45	0.89
Score	0.31	0.74	0.67	0.61	0.55	0.83
B&S (exp 3)	0.48	-	-	0.75	-	-

Table 5: Regression on Gulda data from B&S [11].

We can see that all three audio-based features perform considerably well for both arousal and valence to motivate further analysis.

4.2 Evaluation on Our Dataset

Next, we perform regression on our complete dataset (comprising of 288 unique recordings – 48 pieces \times 6 pianists). The results summary can be seen in Table 6a. Here again, we observe that while DEAMResNet Emotion features perform best on arousal and Score features perform best on valence, Mid-level features show a balanced performance across both the emotion dimensions.

To evaluate generalizability, we perform cross-validation with three different kinds of splits – piece-wise (all 6 performances of a piece are test samples in a fold, for a total of 48 folds), pianist-wise (all 48 pieces of a pianist are test samples in a fold, for a total of 6 folds), and leave-one-out (one recording is the test sample per fold, for a total of 288 folds). This is summarized in Table 6b.

Feature Set	Arousal			Valence		
	\tilde{R}^2	RMSE	Corr	\tilde{R}^2	RMSE	Corr
Mid-level	0.68	0.56	0.83	0.63	0.60	0.80
DEAMResNet	0.70	0.54	0.84	0.42	0.72	0.69
Low-level	0.62	0.59	0.81	0.41	0.74	0.67
Score	0.41	0.75	0.65	0.75	0.49	0.87

(a) Regression metrics with our data

Feature Set	Piece-wise		Pianist-wise		LOO	
	A	V	A	V	A	V
Mid-level	0.68	0.63	0.68	0.64	0.69	0.65
DEAMResNet	0.67	0.37	0.61	0.41	0.68	0.43
Low-level	0.54	0.20	-0.11	-0.05	0.57	0.30
Score	0.08	0.67	0.39	0.75	0.37	0.74

(b) \tilde{R}^2 for different cross-validation splits. A: Arousal, V: Valence, LOO: Leave-One-Out

Table 6: Evaluation results on our data

We see that Mid-level features show good generalization for arousal and are robust to different kinds of splits. They also show balanced performance between arousal and valence for all splits. The good performance of the Score features on the valence dimension (V), here and in the previous experiment, is mostly due to the *Mode* feature; there is a substantial correlation in the annotations between major/minor mode and positive/negative valence.

4.3 Feature Importance within Feature Sets

We use the absolute value of the t-statistic of a feature as the importance measure. T-statistic is defined as the estimated weight scaled with its standard error. We focus on the audio-based feature sets here, as in most realistic applications scenarios, the score information will not be available (and, being constant across different performances, will not be able to distinguish performance aspects). We perform a regression using all audio-based features (numbering 39 in total) and compare the t-values in Figure 2.

We see that the top-4 and top-2 features in arousal and valence, respectively, are Mid-level features. These features also make obvious musical sense – modality is often correlated with valence (positive or negative emotional quality), and tempo, rhythm, and articulation with arousal (intensity or energy of the emotion).

4.4 Explaining Piece-wise Variation

We observe from the annotation data (see Figure 1) that the distribution of emotion ratings of each piece is distinct (here, the 6 performances for each piece form the “distribution” of the piece). In essence, the mean value of arousal or valence depends on the piece in question. Therefore, to take into account this factor of variation, we use linear mixed models [20] to model arousal and valence.

In this linear mixed effect model, the piece id is considered as a “random effect” intercept, which models part of the residual remaining unexplained by the features we are evaluating (“fixed effects”). A feature set that models piece-wise variation better than another set would naturally have a lesser residual variation to be explained by the random effect. We therefore look at which feature set has the least fraction of residual variance explained by the random effect of piece id, defined as:

$$E_{\text{random}} = \frac{\text{Var}_{\text{random}}}{\text{Var}_{\text{random}} + \text{Var}_{\text{residual}}} \quad (1)$$

where $\text{Var}_{\text{random}}$ is the variance of the random effect intercept and $\text{Var}_{\text{residual}}$ is the variance of the residual that remains after mixed effects modeling.

Feature Set	Arousal	Valence
Mid-level	0.50	0.86
DEAMResNet	0.47	0.89
Low-level	0.66	0.90
Score	0.63	0.68

Table 7: Fraction of residual variance explained by the random effect of “piece id”.

We see from Table 7 that the DEAMResNet emotion features best explain piece-wise variation in arousal, followed closely by Mid-level features. For valence, the performance of all three audio-based features are close, with Mid-level features performing the best, however, score features outperform them with a large margin. This is again due to the relationship between mode and valence, and mode covarying tightly with the piece ids.

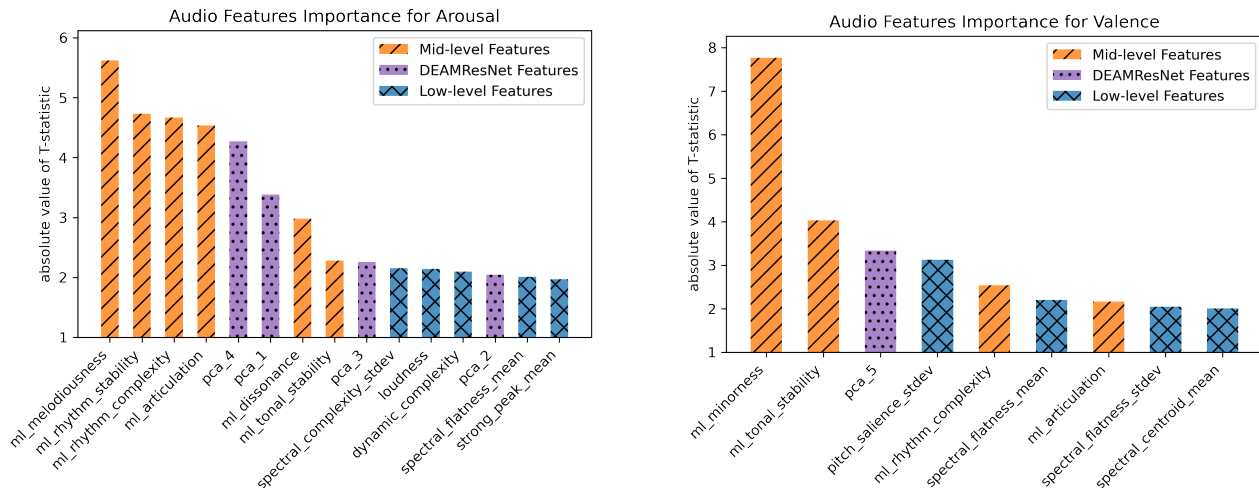


Figure 2: Feature importance for audio features using T-statistic. Only features with $p < 0.05$ are shown.

4.5 Explaining Performance-wise Variation

Evaluation of performance-wise variation modelling cannot be done with the mixed effects approach as in the previous section because the means (of arousal or valence across all pieces) are nearly identical for each pianist.

Therefore, we look at one piece at a time and compute the fraction of variance unexplained (FVU) and Pearson’s correlation coefficient (Corr) between predicted and true values across performances for each such test piece. This is done as leave-one-piece-out cross-validation, and aggregated by taking the means. The p -values of the correlation coefficients are counted and we report the percentage of pieces for which $p < 0.1$. With only 6 performances per piece, a significance level of $p < 0.05$ is obtained for only a handful of pieces. Since score-features based predictions are exactly equal for all performances of a piece, these metrics are not meaningful, and hence the Score feature set is not included here.

Feature Set	Arousal		Valence	
	FVU	Corr ($p < 0.1$)	FVU	Corr ($p < 0.1$)
Mid-level	0.31	0.58 (47.9%)	0.36	0.42 (27.0%)
DEAMResNet	0.32	0.54 (43.8%)	0.61	0.47 (37.5%)
Low-level	0.43	0.56 (54.2%)	0.75	0.38 (22.9%)

Table 8: Evaluation metrics for performance-wise variation. FVU: Fraction of Variance Unexplained. Corr: Pearson’s correlation coefficient.

Again, Mid-level features come out at the top in most measures. To illustrate the modelling of performance-wise variation, we select a few example pieces that have a high variation of emotion between performances and plot them together with the predicted values using mid-level features in Figure 3. The predicted emotion dimensions follow the ratings closely, even for performances that deviate from the average (e.g. the arousals of Gulda’s performance of Prelude in A major and Tureck’s performance of Fugue in E minor.)

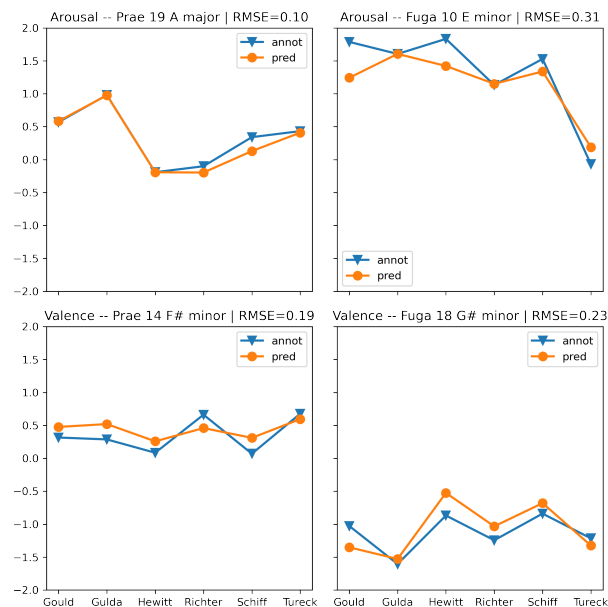


Figure 3: Some example pieces with high emotion variability between performances which are modeled particularly well using mid-level features.

5. PROBING FURTHER

We now describe two additional experiments designed to further probe the predictive power of our feature sets.

5.1 Predicting Emotion of Outlier Performances

Figure 4 shows two examples of pieces where one performance has a vastly different emotional character than the others – in the first example, Gould even produces a negative valence effect (mostly through tempo and articulation) in the E-flat major prelude, which the others play in a much more flowing fashion. A challenge for any model would thus be to predict the emotion of such idiosyncratic performances, not having seen them during training.

We therefore create a test set by picking out the outlier performance for each piece in arousal-valence space

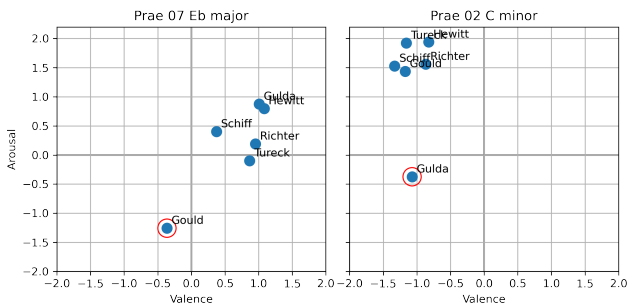


Figure 4: Two examples of outlier performances: Prelude # 7 in Db major, outlier is Gould (left); Prelude #2 in C minor (Gulda, right).

using the elliptic envelope method [21]. This gives us a split of 240 training and 48 test samples (the outliers). We train a linear regression model using each of our feature sets and report the performance on the outlier test set in Figure 5. We see again that Mid-level features outperform the others, for both emotion dimensions. We take this as another piece of evidence for the ability of the mid-level features to capture performance-specific aspects. The surprisingly good performance of score features for valence can be attributed to the fact that for most pieces, the outlier points are separated mostly in the arousal dimension – the spread of valence is rather small (though not always: see the Gould case in Fig. 4) – and the score feature “mode” is an important predictor of valence (see earlier sections).

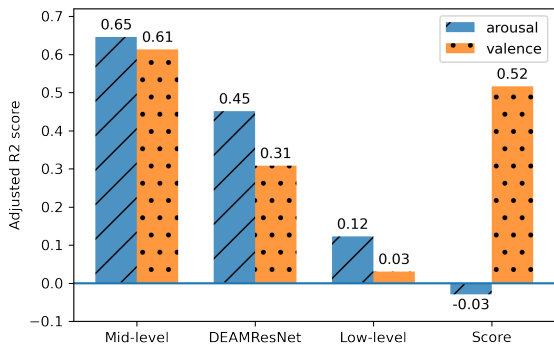


Figure 5: \tilde{R}^2 scores on outlier performances. The outliers were selected using elliptic envelope on the rated arousal-valence space. Out of 48 pieces, the number of times each pianist was an outlier are Gould: 13, Gulda: 10, Tureck: 10, Schiff: 5, Hewitt: 5, Richter: 5.

5.2 Predicting Discrete Emotions

Finally, we evaluate how the feature sets perform in a discrete emotion classification task, which might be relevant in a music recommendation setting, for instance. The emotion ratings are converted to classes simply by reducing them to quadrants in the arousal-valence space. In the literature, these are often associated with the basic emotion labels *happy*, *relaxed*, *sad*, and *angry* (in clockwise fashion, starting at upper right). We then train logistic regres-

sion models using our feature sets and report the leave-one-out cross-validation accuracy in Figure 6. We observe that in this task, all feature sets perform more-or-less equally well, again with a slight advantage for the Mid-level features. Note that the random choice baseline accuracy is 0.25. An emotion classification model based on mid-level perceptual features might be attractive for performance-emotion-aware music recommendation, being able to offer the mid-level concepts not only as explanations but also as ‘handles’ to search for performances with certain characteristics.

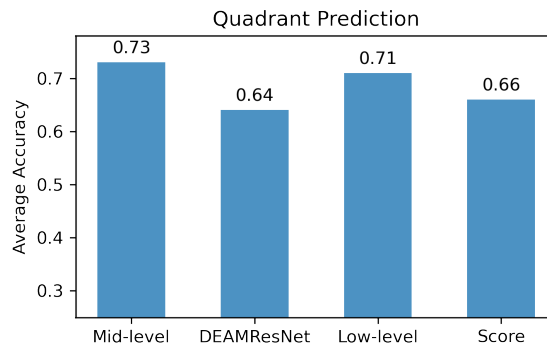


Figure 6: Discrete emotion classification.

6. CONCLUSION

In this work, we evaluated four feature sets – mid-level perceptual features, pre-trained emotion features, low-level audio features, and score-based features on their ability to model and predict emotion in terms of arousal and valence. Specific focus was given on the three audio-based features and their modelling power over performance-wise variation of emotion. Mid-level features emerge as the most robust and important among these.

The search for good features to model music emotion is a worthwhile objective since emotional effect is a very fundamental human response to music. Features that provide a better handle on content-based emotion recognition can have a significant impact on applications such as search and recommendation. Modelling emotion is also becoming increasingly relevant in generative music, allowing possibilities such as expressivity- or emotion-based snippet continuation and emotion-aware human-computer collaborative music.

From the experiments presented in this paper, it is clear that deep-learning-based feature extractors are strong competition to the audio features typically used for emotion recognition [3]. Here the importance of Mid-level features is even more pronounced – in addition to being able to model both arousal and valence well under different conditions, they also provide intuitive musical meaning to each feature, and have been previously used as the basis for explainable emotion recognition in [16].

7. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU’s Horizon 2020 research & innovation programme, grant agreement No. 670035 (“Con Espressione”), and the Federal State of Upper Austria (LIT AI Lab). The authors would like to thank Carlos Cancino Chacón for helpful discussions and calculating score-performance alignments and score features, and Jan Schlüter and Rainer Kelz for data collection and preparation. Thanks to Aimee Battcock, Cameron Anderson, and Michael Schutz for sharing their annotation data.

8. REFERENCES

- [1] A. Gabrielsson and P. N. Juslin, “Emotional expression in music performance: Between the performer’s intention and the listener’s experience,” *Psychology of Music*, vol. 24, no. 1, pp. 68–91, 1996.
- [2] J. Akkermans, R. Schapiro, D. Müllensiefen, K. Jakubowski, D. Shanahan, D. Baker, V. Busch, K. Lothwesen, P. Elvers, T. Fischinger *et al.*, “Decoding emotions in expressive music performances: A multi-lab replication and extension study,” *Cognition and Emotion*, vol. 33, no. 6, pp. 1099–1118, 2019.
- [3] R. Panda, R. M. Malheiro, and R. P. Paiva, “Audio features for music emotion recognition: a survey,” *IEEE Transactions on Affective Computing*, pp. 1–1, 2020.
- [4] M. Soleymani, A. Aljanaki, Y.-H. Yang, M. N. Caro, F. Eyben, K. Markov, B. W. Schuller, R. Veltkamp, F. Weninger, and F. Wiering, “Emotional analysis of music: A comparison of methods,” in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1161–1164.
- [5] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, “Music emotion recognition: A state of the art review,” in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, vol. 86, 2010, pp. 937–952.
- [6] Y.-H. Yang and H. H. Chen, “Machine recognition of music emotion: A review,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, pp. 1–30, 2012.
- [7] R. Orjeseck, R. Jarina, M. Chmulik, and M. Kuba, “DNN based music emotion recognition from raw audio signal,” in *29th International Conference Radioelektronika 2019 (RADIOELEKTRONIKA)*. IEEE, 2019, pp. 1–4.
- [8] M. B. Er and I. B. Aydilek, “Music emotion recognition by using chroma spectrogram and deep visual features,” *International Journal of Computational Intelligence Systems*, vol. 12, no. 2, pp. 1622–1634, 2019.
- [9] N. HE and S. Ferguson, “Multi-view neural networks for raw audio-based music emotion recognition,” in *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2020, pp. 168–172.
- [10] J. Grekow, “Musical performance analysis in terms of emotions it evokes,” *Journal of Intelligent Information Systems*, vol. 51, no. 2, pp. 415–437, 2018.
- [11] A. Battcock and M. Schutz, “Acoustically expressing affect,” *Music Perception: An Interdisciplinary Journal*, vol. 37, no. 1, pp. 66–91, 2019.
- [12] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor *et al.*, “Essentia: An audio analysis library for music information retrieval.” International Society for Music Information Retrieval (ISMIR), 2013, pp. 493–498.
- [13] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Proceedings of the 14th python in science conference*, vol. 8, 2015, pp. 18–25.
- [14] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 2001.
- [15] A. Aljanaki and M. Soleymani, “A Data-driven Approach to Mid-level Perceptual Musical Feature Modeling,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France*, 2018, pp. 615–621.
- [16] S. Chowdhury, A. Vall, V. Haunschmid, and G. Widmer, “Towards Explainable Music Emotion Recognition: The Route via Mid-level Features,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, 2019.
- [17] K. Koutini, H. Eghbal-Zadeh, M. Dorfer, and G. Widmer, “The Receptive Field as a Regularizer in Deep Convolutional Neural Networks for Acoustic Scene Classification,” in *2019 27th European signal processing conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.
- [18] S. Chowdhury and G. Widmer, “Towards explaining expressive qualities in piano recordings: Transfer of explanatory features via acoustic domain adaptation,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 561–565.
- [19] A. Aljanaki, Y.-H. Yang, and M. Soleymani, “Developing a Benchmark for Emotional Analysis of Music,” *PLoS one*, vol. 12, no. 3, p. e0173392, 2017.
- [20] J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team, *nlme: Linear and Nonlinear Mixed Effects Models*, 2021, r package version 3.1-152. [Online]. Available: <https://CRAN.R-project.org/package=nlme>
- [21] P. J. Rousseeuw and K. V. Driessen, “A fast algorithm for the minimum covariance determinant estimator,” *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

COSINE CONTOURS: A MULTIPURPOSE REPRESENTATION FOR MELODIES

Bas Cornelissen Willem Zuidema John Ashley Burgoyne

Institute for Logic, Language and Computation, University of Amsterdam

b.j.m.cornelissen@uva.nl, zuidema@uva.nl, j.a.burgoyne@uva.nl

ABSTRACT

Melodic contour is central to our ability to perceive and produce music. We propose to represent melodic contours as a combination of cosine functions, using the discrete cosine transform. The motivation for this approach is twofold: (1) it approximates a maximally informative contour representation (capturing most of the variation in as few dimensions as possible), but (2) it is nevertheless independent of the specifics of the data sets for which it is used. We consider the relation with principal component analysis, which only meets the first of these requirements. Theoretically, the principal components of a repertoire of random walks are known to be cosines. We find, empirically, that the principal components of melodies also closely approximate cosines in multiple musical traditions. We demonstrate the usefulness of the proposed representation by analyzing contours at three levels (complete songs, melodic phrases and melodic motifs) across multiple traditions in three small case studies.

1. INTRODUCTION

Humans are born with a remarkable sensitivity to melodic contour. This is dramatically illustrated when newborns cry: the cries of German babies tend to go down in pitch, but those of French babies go up, even if falling contours are physiologically easier to produce [1]. By imitating the intonation patterns of their mothers' language, babies take the first steps towards a spoken language—helped by exaggerated pitch contours of infant directed speech [2]. Contour perception remains central to speech, for intonation or even word distinctions, but is also a key ingredient of human musicality [3]. Dowling famously argued that melodies are remembered as two independent parts, a scale and a contour [4]. A scale then functions as a ladder “on which the ups and downs of the contour where hung.” Indeed, when listening to novel melodies, contours appear to stand out more than the exact intervals and influence the perceived similarity of melodies [5]. That has also motivated studies of contour in MIR, in particular for measuring melodic sim-

ilarity [6]. As we briefly review below, many representations of contour have been proposed in answer to the recurring question: how can one best describe melodic contour?

We propose representing melodies as combinations of cosine functions. This is motivated by the need for a concise, maximally informative representation: how can we capture as much of the variability in contour data in as few dimensions as possible? The easiest solution would be to use a *principal component analysis* (PCA). In section 4, we show empirically that the principal components of melodies do not take arbitrary shapes, but in fact closely approximate cosines. We then relate this observation to theoretical results showing that the principal components of certain random walks are sinusoidal, as a result of a particular covariance structure. The proposed ‘cosine contour’ space thus closely approximates the optimal solution provided by PCA, but offers several benefits. The key argument for this representation is theoretical and we leave a systematic comparison of contour representations for future work. Instead we discuss three case studies that demonstrate the usefulness of cosine contours.

Cosine contours meet several desiderata for contour representations. First, a good representation respects the linear structure of melody and is *invariant to transposition and tempo changes*. Second, the representation should be *interpretable* and *intuitive* (and, in particular, avoid some of the shortcomings of polynomial coefficients). Third, the representation should support *variable levels of abstraction*, so that one can interpolate between a broad summary of the shape, and the exact pitch curve. Fourth, we look for a *broadly applicable* and *culturally neutral* representation: it should be able to describe contours from different cultures, or even from different domains (e.g., speech). It should also be able to handle both audio and symbolic data, although we only analyze symbolic data here.

2. WHAT IS MELODIC CONTOUR?

Melodic contour is a general description of a melody's shape that abstracts away from the particular pitches and precise rhythms. It has been characterised in many different ways. Ethnomusicologists (and composers) have used *contour typologies*: small sets of contour types [7]. David Huron, for example, distinguished nine types of contours by comparing the initial and final pitches to the average pitch on the middle part of a melody [8]. When, say, the initial is above the middle, which in turn equals the final,



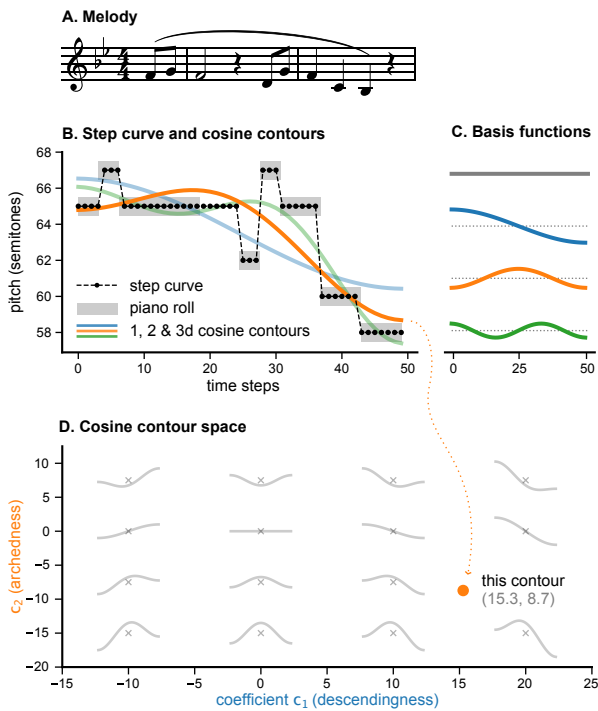


Figure 1. Cosine contours represent a melodic contour as a combination of cosine functions. (A) This is illustrated for a short melodic phrase. (B) A piano roll is interpolated to obtain fixed-length vector of MIDI pitches (black curve). This vector is approximated using a discrete cosine transform (coloured curves). Increasing the dimensionality, from, e.g., 1 (blue) to 3 (green) improves the approximation. (C) The basis functions correspond to simple shapes. This makes the cosine contour space interpretable, as illustrated in (D) for the first two dimensions. Every point in this space defines a contour shape, varying in what we call the *descendingness* and *archedness*. The orange dot represents the orange contour from (B).

the melody has a ‘descending-horizontal’ contour. Such a formal typology can be used in MIR [9], but typologies have also been defined using verbal descriptions or even drawings [7, 10]. CantoCore, for example, instructs an annotator to look for six types: ascending, descending, arched, U-shaped, undulating and horizontal [11]. Even though the types are less sharply defined, such typologies have inspired cross-cultural generalizations such as the *melodic arch hypothesis*: the claim that melodic phrases tend to be arch-shaped or descending [8, 12–14].

In melody extraction from audio, contours are usually represented by sequences of pitches ordered in time. Various contour features derived from this, such as the range or pitch deviation, have been used in classification tasks [15–18]. Contours in symbolic data can be similarly represented as *step curves* (figure 1B, black line) [19,20]. *Parsons code* drastically simplifies a step curve [21]. It describes the direction of movement from one note to the next (up, down, or level) and discards interval size and note durations. Variants between these two extremes have also been used, by distinguishing various classes of jump sizes [6]. Another

strategy is to focus on salient notes, typically turning points (maxima and minima), and to discard other notes [7, 18, 19]. This often requires special handling of ornaments [20], possibly tailored to the repertoire. Yet another approach considers the relative ordering of all pairs of notes in a melody, summarized in a matrix. Such combinatorial models in way expand rather than reduce the representation, break the linearity of the melody and are sensitive to local changes [20].

Finally, one can describe melodies using continuous functions. Müllensiefen and Wiggins fit a polynomial function to a step curve and use the coefficients to represent the contour [20]. The degree of the polynomial is chosen per phrase, using the Bayesian information criterion (BIC) to avoid overfitting. Polynomial coefficients are quite difficult to interpret, however: they change drastically when the degree changes, and can also be sensitive to changes in the data, especially when the polynomials are not orthogonal and introduce correlations between the coefficients (collinearity). Instead of fitting a function to the contour, one can also *decompose* the contour and express it as a sum of (orthogonal) basis functions. Velarde and colleagues have for example used *Haar wavelets* as basis functions in musical pattern discovery [22]. The step-like shapes of those wavelets are well suited to describe particular melodic patterns, but make them less suited for describing the overall contour. An alternative basis of sinusoidal functions is implicit in Schmuckler’s use of a Fourier analyses to represent melodic contour [23]. This has been interpreted as measuring the ‘periodic information’ in a melody, and was reported to correlate with perceived similarity.

3. DATA

With the broad applicability in mind, we analyze music from several independent traditions. The choice of traditions was partly motivated by our aim to analyze contours at multiple levels of description: we expect (different) regularities at different levels. At the highest level, complete *songs* can have characteristic shapes, and those shapes may differ between traditions. At the smaller level *phrases* may be subject to the melodic arch hypothesis cited above. Finally, at the smallest level, *melodic motifs* could exhibit sequential structure, for example when melodies in a repertoire are formed by stringing together melodic motifs (sometimes called *centonization* [24]). We also analyze *random segments* obtained by slicing a melody at random in approximately phrase-length segments, so that their boundaries usually do not overlap with actual phrase boundaries [25].

One tradition for which all of these levels are directly available is Gregorian chant, thanks to two recently released corpora: the CantusCorpus and the GregoBaseCorpus [25]. Gregorian chant has been sung in Roman Catholic churches for well over a thousand years. The close connection between music and text in chant suggests a natural subdivision of the music into motifs corresponding to words or syllables. The notation suggests even smaller motifs: it is based on small figures, called *neumes*, that represent short groups of notes [26]. To analyse motif contours, we use chants from the CantusCorpus (v0.2) with transcriptions

of medieval manuscripts, which include neume boundaries. We focus on the two largest chant genres: *antiphons* and *responsories*. Phrase boundaries are not available in the CantusCorpus, however, and so for that, we turn to the GregoBaseCorpus (v0.3) of modern chant transcriptions. Modern chant notation includes explicit breathing marks (*pausas*), which have been used to extract phrases [25].

Phrase markings are also included in the Essen Folksong Collection [27], from which we analyse phrases from German and Chinese folksongs. We focus on the two largest subsets, ‘Erk’ [28] (9782 contours) and ‘Han’ (7601 contours).¹ At the level of complete songs, we also add music from the Sioux people made available in the *Densmore Collection* [29, 30]. In the supplementary material, we include some further analyses of several other traditions from the *Essen* and *Densmore* collections.

We convert all melodies (be it songs, phrases or motifs) to step contours by extracting note onsets (in quarter notes) and pitches (in MIDI semitones). We then interpolate a step function through these points, from which we sample $N = 100$ equally spaced pitches. Those pitches are collected in vectors $\mathbf{x} = (x_0, \dots, x_{N-1})$ (black curve in figure 1A), which are the basic data analysed in this paper.²

Our starting representation makes several assumptions that seem reasonable (and common: [13, 14, 22]) when only interested in contour. First, we ignored all rests. Second, we normalize the duration of all contours. Both 3-note motifs and 30-note songs are represented by vectors of 100 pitches. The relative durations within that melody are of course retained, so we would still see that contours of short motives are probably simpler than those of long melodies. Third, we assume Euclidean distances between melodies. This is usually problematic, but less so when we are only interested in contour similarity. Our analyses require that all contours are embedded in a vector space. Using more sophisticated measures such as dynamic time warping distance, would require us to reconstruct a space (e.g., using multidimensional scaling), and make the analyses less transparent. Finally, note that we do *not* center the contours to have mean pitch 0. This is sometimes done to make contours transposition invariant and more directly comparable [14, 22, 25]. We will soon see that our proposed representation elegantly resolves this problem without requiring centring.

4. PRINCIPAL COMPONENTS OF CONTOURS

In this section, we explore principal component analysis applied to contours. The goal of PCA is to find a set of orthogonal axes, the *principal components*, that contain most of the variance in the dataset. Note that the principal components, like the original contours from our data, are N -dimensional vectors, such that the contours and components can be interpreted and plotted in the same space.

In figure 2A, we show results from applying PCA on a large dataset of Gregorian chant (similar results with Ger-

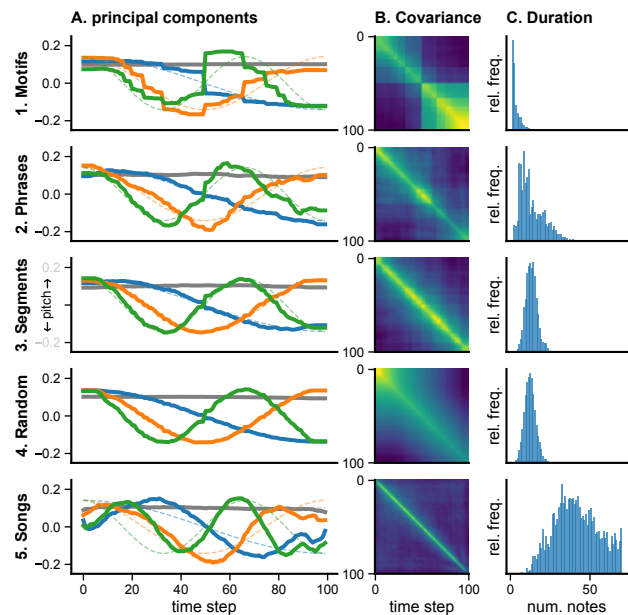


Figure 2. Principal components of contours (solid lines) are roughly cosine shaped (dashed) across different levels (A). This is a result of the particular structure of the covariance matrix (B): matrices of this type have Fourier basis functions as their eigenvectors. This is clearest for phrases (2) or random segments from melodies (3), here of similar length as phrases. Crucially, we see the same effect for simulated, contour-like random walks (4). For complete songs (5) the effect is less clear, probably due to differences in typical length (C) and data size. Contours in 1–4 are from Gregorian chant.

man and Chinese folksongs can be found in supplement S2). We plot the first four principal components of several types of melodies: short motifs (syllables), phrases, random segments of melodies, and complete songs. We show responsory syllables from CantusCorpus for the motifs, antiphon phrases from the GregoBaseCorpus and finally all song contours from GregoBaseCorpus.

Surprisingly, we find that the principal components are highly similar across most of those data sets, and correspond to well-known contour shapes: descending, convex, and—perhaps—undulating. This is clearest for the phrases and random segments. For complete songs the effect is weaker, especially for even smaller datasets (see the supplement S2). Besides small data sizes, the fact that songs are much longer also plays a role (see fig. 2C). We also applied the analysis on simulated random walks approximating phrases: we draw the number of notes from a similar length distribution, normalize the duration and then sample $N = 100$ pitches as before (see supplement S1 for details). Interestingly, the pattern is now even clearer, suggesting there must be a mathematical explanation.

To give that explanation, we need to first describe PCA more formally. We consider a collection of M contour vectors \mathbf{x}_m of length N . Denote the sample mean by $\bar{\mathbf{x}} = \frac{1}{M} \sum_m \mathbf{x}_m$ and the centered data by $\hat{\mathbf{x}}_m = \mathbf{x}_m - \bar{\mathbf{x}}$. The first principal component of the dataset is then defined as

¹ Much is unclear about the exact (bibliographic) origins of the Chinese subset of *Essen*. This is problematic given its wide use in computational musicology and deserves further attention from the community.

² See github.com/bacor/cosine-contours for data, code and supplements.

a normalized vector $\mathbf{u}_1 \in \mathbb{R}^D$ for which the projected data $\{\mathbf{u}_1^T \mathbf{x}_m : 1 \leq m \leq M\}$ has maximal variance. It can be shown (e.g., [31]) that this is the case when \mathbf{u}_1 is an eigenvector corresponding to the largest eigenvalue λ_1 of the covariance matrix

$$\mathbf{S} = \frac{1}{M} \sum_{m=1}^M (\mathbf{x}_m - \bar{\mathbf{x}})(\mathbf{x}_m - \bar{\mathbf{x}})^T, \quad (1)$$

so that $\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$. It follows that the projected variance is given by λ_1 , the largest eigenvalue. The other principal components similarly emerge as the other eigenvectors of the covariance matrix.

The covariance matrices (figure 2B) for both random walks and our empirical data have a particular structure: they *roughly* resemble *Toeplitz matrices*, which have fixed values along each of their diagonals. Such covariance structures are frequently encountered in spatial or temporal data, when the covariance decreases with the distance between the points [32–34]. With the empirical contours that appears to be the case (and for random walks it is there by design): there is higher correlation between successive pitches and lower correlation between distant pitches. As a result, the higher covariances are concentrated along the diagonal. Again, this clearest for the phrases and random segments. For motifs we see some deviations: two ‘blocks’ in the covariance matrix, and corresponding jumps half way through the principal components. This is easily explained by the fact that motifs often span only two notes. In that case, all pitches in the first half of the contour are then perfectly correlated, as are pitches in the final half. Crucially, despite such deviations from a perfect Toeplitz structure, the principal components are still well-approximated by cosines.

If you let a Toeplitz matrix grow in size, it asymptotically tends towards a *circulant* matrix, preserving properties such as eigenvalues and eigenvectors along the way [32]. Circulant matrices have exactly the same values in every row, but rotated one step to the right with respect to the previous row. This has the surprising result that all circulant matrices have the same eigenvectors: basis vectors of the discrete Fourier transform. For a real and symmetric matrices, like covariance matrices, this results in cosine-shaped eigenvectors of increasing frequency—exactly what we see in figure 2. We discuss all of this in more detail in the supplement S2. In sum, because of a Toeplitz-like covariance structure, the principal components of melodic contours will tend to look like cosine functions.

5. COSINE CONTOURS

Next we turn this observation, and its explanation, into a proposal for a new contour representation. The idea is to approximate the principal components by cosine functions and then project the contours on those first few cosines to obtain a low-dimensional representation. This is exactly equivalent to taking a *discrete cosine transform* (DCT) of the contour [35].

Formally, consider a collection of contours of length N as before. We approximate the k -th principal component

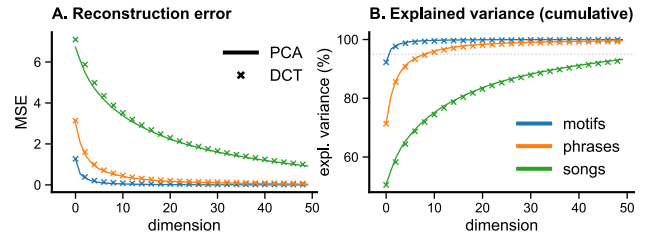


Figure 3. DCT approximates PCA, the optimal transform, in terms of the reconstruction error (A) and the explained variance ratio (B). The reconstruction error is the mean squared error between an contour and a lower dimensional reconstruction. Note that data corresponds to figure 2, and that we did *not* discard the first component c_0 of the DCT in this figure.

\mathbf{u}_k by a vector $\mathbf{v}_k = (v_k(0), \dots, v_k(N-1))$ whose entries are given by the cosine function³

$$v_k(n) = \alpha_k \cdot \cos \frac{\pi(2n+1)k}{2N}. \quad (2)$$

Here $\alpha_0 = 1/\sqrt{N}$ and $\alpha_k = \sqrt{2/N}$ for $k \geq 1$ are normalizing constants ensuring that \mathbf{v}_k has unit norm. The projection of a contour $\mathbf{x} = (x_0, \dots, x_{N-1})$ on \mathbf{v}_k is then given by the inner product $c_k = \mathbf{v}_k^T \mathbf{x}$. Expanding this gives the usual definition of the discrete cosine transform (DCT-II):

$$c_k = \sum_{n=0}^{N-1} x_n \alpha_k \cos \frac{\pi(2n+1)k}{2N}. \quad (3)$$

Conversely, the contour can be reconstructed from the coefficients c_k using the inverse transform $x_n = \sum_{k=0}^{N-1} c_k v_k(n)$. Using only $D < N$ coefficients, we define our low-dimensional *cosine contour representation* as $C_D(\mathbf{x}) = (c_1, \dots, c_D)$. Note that we deliberately discard c_0 . This coefficient corresponds to a flat line and describes the overall pitch height of a contour: exactly what we need to get rid of to make the contour transposition invariant. In this way we resolve the centering of contours discussed above.

Why use this representation instead of principal components? Indeed, a principal component projection (also known, in this context, as the *Karhunen-Loève transform*), is optimal in several ways [35,37]. Not only does it decorrelate the data, it also packs most variance in the first few transform coefficients (sometimes called *energy compaction*), and minimizes the reconstruction error when using only a few coefficients. However, the transformation depends on the data. Concretely, the principal components of German phrase contours differ from Chinese ones. Any choice for using one of the two is arbitrary. In contrast, the DCT is a principled, neutral solution—that approximates the optimal transform. In fact, the DCT was originally introduced for similar reasons [35], and was then found to empirically approximate PCA well in domains ranging from image to audio [37]. The current results suggest that the same applies for melodies.

³ These basis functions correspond to the most popular version of the discrete cosine transform, DCT-II, for which fast implementations are widely available; others would have been possible [36].

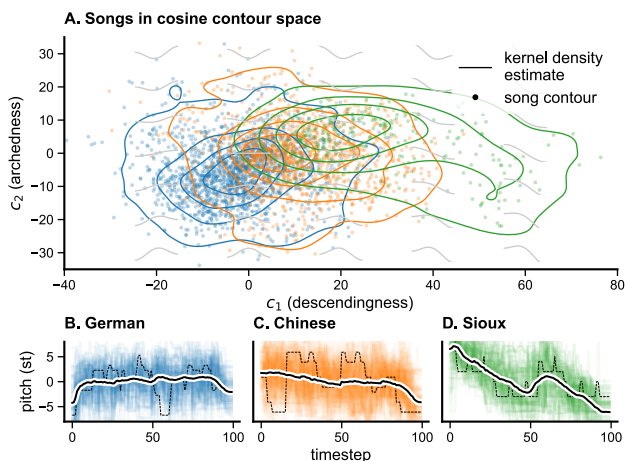


Figure 4. Songs of three cultures represented in the cosine contour space (A) show substantial variability. The average of all contours in a tradition (B–D) also illustrates this (thick black lines; dashed lines highlight one contour).

6. EVALUATION AND CASE STUDIES

We evaluate proposed contour representation by comparing it to a principal component transformation, to demonstrate that representation is close to the optimum. We further designed three case studies to illustrate its usefulness at the levels of (1) song, (2) phrases and (3) motifs. The case studies show that the representation is musicologically meaningful, as it allows visualization of variation (1), a quantitative evaluation of constraints on variation (2), and accurate classification into traditional categories (3). For simplicity, we only look at two dimensional representations in these case studies, but higher dimensions may be useful in practice.

6.1 Optimality

To empirically verify the claim that the DCT approximates the optimal PCA transform, we compute the reconstruction error and the explained variance ratio using the same data as before. The reconstruction error is measured as the mean square error between a contour and its D -dimensional reconstruction, using either the principal components (PCA) or cosines (DCT) as basis functions (so for $D = N$, the reconstruction is guaranteed to be perfect). Figure 3A shows that the reconstruction errors of DCT closely approximate that of PCA. For the shorter contours (motifs and phrases), the error very rapidly decreases, indicating that low-dimensional representations are already effective. Indeed, to explain 95% of the variance using cosine contours, you need 1 dimension for motifs, 9 for phrases and 61 for songs (this is sometimes called the *effective dimensionality* [38]).⁴

6.2 Case Study 1: Visualizing different traditions

Low dimensional representations of song contours are not likely to be very informative, yet we find that some traditions can be somewhat distinguished in just two dimensions.

⁴ However, note that Moore et al [38] show that high-dimensional random walks can falsely appear to have a low effective dimensionality.

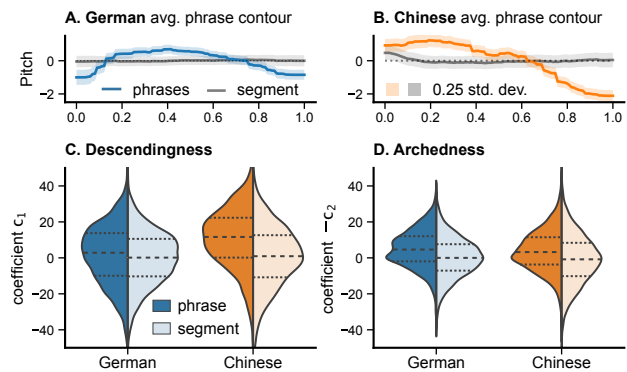


Figure 5. Phrases of German (A) and Chinese (B) songs tend to be more descending and arched compared to random segments from the same melodies, as visible from their average contours. This can be quantified by comparing the first (C) and second (D) coefficients of their cosine representations.

Figure 4 shows song contours from German, Chinese and Sioux songs. Sioux songs have a striking overall shape (subplot D), often strongly descending, which is reflected in the distribution of contour shapes. Similarly, German songs appear to be more arch-like than songs from the other traditions.

6.3 Case Study 2: The melodic arch hypothesis

In a second case study, we look at the melodic arch hypothesis, which states that *phrases* tend to be arch-shaped or descending [8] (see figure 5A, B) in a way that it becomes much easier to test (cf. [14]). We observe that the first component c_1 of a cosine representation roughly measures the *descendingness* of the contour, and, similarly, that $-1 \cdot c_2$ measures the *archedness*. The melodic arch hypothesis can thus be reformulated as stating that c_1 and $-c_2$ are larger for phrases than for random segments of the melodies (cf. [25]). Comparing Chinese and German phrases, we find that all are significantly ($p \ll 0.001$) more descending and arched than the corresponding random segments (see figure 5C, D). This demonstrates that the coefficients of the cosine contour representation are musicologically meaningful.

6.4 Case Study 3: Mode classification

In the final case study, we evaluate the performance of this contour representation on a task: mode classification in plainchant. Gregorian chant uses a system of eight *modes*: Dorian, Phrygian, Lydian and Mixolydian, each in the two flavours plagal and authentic. Modes differ not only in their scales, but also in their melodic movement. Plagal melodies tend to move lower than authentic ones, closer around the tonal center. In a recent paper we suggest that the mode of Gregorian chant can be predicted from contours alone, in that case using a Parsons code contour representation [39]. We sliced up chants in sequences of motifs corresponding to the notational units (so called *neumes*) or textual units: all notes set to one *syllable* of the text would form a unit, and similarly for *words*. Next, we represented chants as

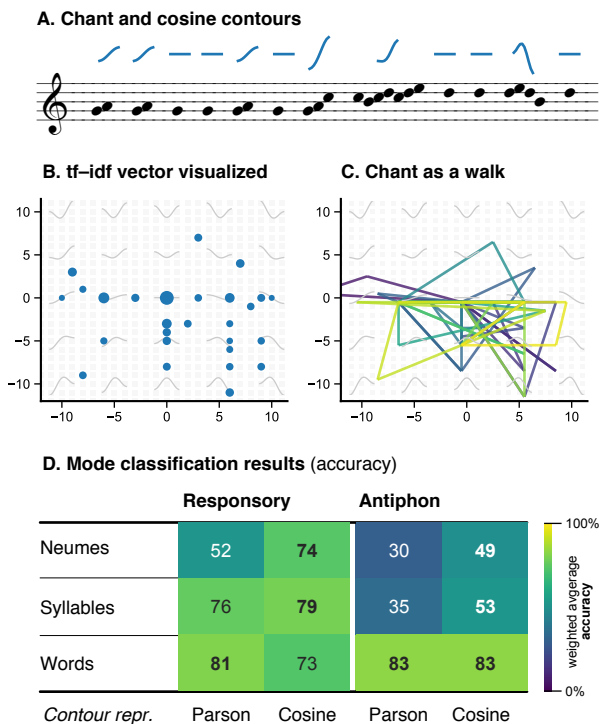


Figure 6. Motifs used for mode classification in Gregorian chant. (A) A chant is segmented into motifs derived from the notation (neumes) or lyrics (syllables, words). The blue curves show the two-dimensional cosine contours for those motifs. (B) We discretize the contour space and represent the chant as a vector of *tf-idf* weighed motif frequencies (‘grid cell frequencies’). Dots illustrate the nonzero entries of this vector for the chant shown above. (C) The chant is now a walk through contour space, but our ‘bag of motifs’ ignores order. (D) Using these vectors to classify mode, we outperform a previous study using a Parsons code for the smaller motifs neumes and syllables.

vectors of motif or *term frequencies* (*tf*), where each entry was weighted by the *inverse document frequency* (*df*; the number of chants or documents containing that motif). A linear support vector machine was then trained on these *tf-idf* vectors to predict the mode.

We repeat these experiments using a two-dimensional cosine representation for the motifs rather than a Parsons code. There is one technical problem: whereas cosine contours are continuous, the *tf-idf* model requires a discrete vocabulary of motifs. We therefore discretize the cosine contour space to a grid, and effectively treat every chant as a sequence of grid-cells (fig. 6C). All in all, this introduces two new parameters to the experiment: the dimensionality of the cosine contour and the resolution of the grid. In this case study, we do not tune these parameters and focus on two dimensional contours, discretized to a grid between -20 and 20 with a grid size of 1. For ease of reading, the figure 6B shows the grid only from -10 to 10.

The results are summarized in figure 6D. We see an interesting pattern: the cosine contours outperform the original results for small motifs such as neumes and syllables,

but not for words, which are much longer motifs. This seems to make sense: two dimensional cosine contours are a fairly crude approximation of those longer contours, but may reasonably approximate short motifs.

7. DISCUSSION AND CONCLUSIONS

This paper proposed a novel representation for melodies using the discrete cosine transform. Observing that the principal components of melodies tend to be shaped like cosines, this representation approximates the optimal representation in the sense that it packs most variance in a few dimensions. First, the cosine representation is easily interpretable, since it presents contours as a linear combination of cosine functions with intuitive shapes. Second, by changing the dimensionality, the level of abstraction of the contour can be varied, allowing arbitrary small reconstruction error by including more and more dimensions. Third, this representation allows one to map contours at multiple levels, from motifs to songs, to one common space. The cosine representation thus creates a common ground for comparing contours across traditions and levels. That is possible as, fourth, the representation is independent of the data, and in that sense culturally neutral.

The observation that principal components of spatial and temporal data can have sinusoidal shapes is not novel, but does not appear to be widely known. Indeed, the sinusoidal shapes have been interpreted as genuine effects, rather than mathematical artefacts. For example, one study interpreted gradients in the principal components of human genetic variation across the world as evidence for certain migration events in human history [40]. Closer inspection revealed that those gradients were sinusoidal ‘artefacts’ analogous to those reported in the present paper [33]. Closer to MIR, it has been observed that the training trajectories of deep neural networks have sinusoidal principal components [41], for the same reason. Again, a detailed analysis [34] revealed these were artefacts, but accurately reflecting the behaviour of high-dimensional random walks [34, 38]. We hope this paper helps increasing the awareness of this phenomenon.

The present work only begins to explore this new contour representation and raises many further questions. One particularly promising possibility is the application to audio data. We only explored symbolic data, but the proposed representation lends itself well for applications on acoustic data. One application we hope to explore further is the analysis of speech intonation using the cosine contour representation. A possible other avenue would be the analysis of folk song recordings, of which vast collections have been collected. Folk song researchers have often used contour in some way to organize repertoires [7], and this representation may contribute to that. Contour typologies have also been used in cross-cultural comparisons (see e.g. [12]). Many typologies have been proposed [7, 8, 10, 11], but they have not been systematically evaluated, and we think the proposed representation will be valuable there.

8. ACKNOWLEDGEMENTS

We would like to thank Henkjan Honing and Marianne de Heer Kloots for their feedback on the manuscript. We would also like to thank the four anonymous reviewers for their careful reviews; we have tried to address all your comments.

9. REFERENCES

- [1] B. Mampe, A. D. Friederici, A. Christophe, and K. Wermke, “Newborns’ cry melody is shaped by their native language,” *Current Biology*, vol. 19, no. 23, pp. 1994–1997, 2009.
- [2] K. Wermke, M. P. Robb, and P. J. Schluter, “Melody complexity of infants’ cry and non-cry vocalisations increases across the first six months,” *Scientific Reports*, vol. 11, no. 1, p. 4137, Dec. 2021.
- [3] H. Honing, C. ten Cate, I. Peretz, and S. E. Trehub, “Without it no music: Cognition, biology and evolution of musicality,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 370, no. 1664, p. 20140088, Mar. 2015.
- [4] W. J. Dowling, “Scale and contour: Two components of a theory of memory for melodies,” *Psychological Review*, vol. 85, no. 4, pp. 341–354, 1978.
- [5] M. A. Schmuckler, “Tonality and contour in melodic processing,” in *The Oxford Handbook of Music Psychology*, 2nd ed., S. Hallam, I. Cross, and M. Thaut, Eds. Oxford University Press, 2016.
- [6] D. Müllensiefen and K. Frieler, “Cognitive adequacy in the measurement of melodic similarity: Algorithmic vs. human judgments,” in *Music Query: Methods, Models, and User Studies*, ser. Computing in Musicology. MIT Press, 2004, no. 13, p. 30.
- [7] C. R. Adams, “Melodic contour typology,” *Ethnomusicology*, vol. 20, no. 2, p. 179, May 1976.
- [8] D. Huron, “The melodic arch in Western folksongs,” *Computing in musicology*, vol. 10, pp. 3–23, 1996.
- [9] D. Müllensiefen, “Fantastic: Feature ANalysis Technology Accessing STatistics (In a Corpus): Technical Report v1.5,” Tech. Rep., 2009.
- [10] T. Kelkar, U. Roy, and A. R. Jensenius, “Evaluating a collection of sound-tracing data of melodic phrases,” in *19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 74–81.
- [11] P. E. Savage, E. Merritt, T. Rzeszutek, and S. Brown, “CantoCore: A new cross-cultural song classification scheme,” in *Analytical Approaches to World Music*, vol. 2, 2012, pp. 87–137.
- [12] P. E. Savage, S. Brown, E. Sakai, and T. E. Currie, “Statistical universals reveal the structures and functions of human music,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 29, pp. 8987–8992, 2015.
- [13] A. T. Tierney, F. A. Russo, and A. D. Patel, “The motor origins of human and avian song structure,” *Proceedings of the National Academy of Sciences*, vol. 108, no. 37, pp. 15 510–15 515, Sep. 2011.
- [14] P. E. Savage, A. T. Tierney, and A. D. Patel, “Global music recordings support the motor constraint hypothesis for human and avian song contour,” *Music Perception: An Interdisciplinary Journal*, vol. 34, no. 3, pp. 327–334, Feb. 2017.
- [15] R. M. Bittner, J. Salamon, J. J. Bosch, and J. P. Bello, “Pitch contours as a mid-level representation for music informatics,” in *Audio Engineering Society Conference on Semantic Audio*, Erlangen, Germany, 2017.
- [16] M. Panteli, R. Bittner, J. P. Bello, and S. Dixon, “Towards the characterization of singing styles in world music,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 636–640.
- [17] R. M. Bittner, J. Salamon, S. Essid, and J. P. Bello, “Melody extraction by contour classification,” in *Proceedings of the 16th International Conference on Music Information Retrieval (ISMIR 2015)*, Malaga, Spain, 2015, pp. 500–506.
- [18] J. Salamon, B. Rocha, and E. Gomez, “Musical genre classification using melody features extracted from polyphonic music signals,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Kyoto, Japan: IEEE, Mar. 2012, pp. 81–84.
- [19] W. Steinbeck, *Struktur und Ähnlichkeit: Methoden automatisierter Melodieanalyse*. Bärenreiter, 1982.
- [20] D. Müllensiefen and G. A. Wiggins, *Polynomial Functions as a Representation of Melodic Phrase Contour*, ser. Hamburger Jahrbuch Für Musikwissenschaft. Peter Lang, Jan. 2012.
- [21] D. Parsons, *Directory of tunes and musical themes*. Spencer Brown, 1975.
- [22] G. Velarde, D. Meredith, and T. Weyde, “A wavelet-based approach to pattern discovery in melodies,” in *Computational Music Analysis*, D. Meredith, Ed. Cham: Springer International Publishing, 2016, pp. 303–333.
- [23] M. A. Schmuckler, “Testing models of melodic contour similarity,” *Music Perception*, vol. 16, no. 3, pp. 295–326, Apr. 1999.
- [24] T. Nuttall, M. G. Casado, V. N. Tarifa, R. C. Repetto, and X. Serra, “Contributing to new musicological theories with computational methods: The case of centonization in Arab-Andalusian music,” in *Proceedings of the*

- 20th International Conference on Music Information Retrieval (ISMIR 2019)*, Delft, The Netherlands, 2019, pp. 223–228.
- [25] B. Cornelissen, W. Zuidema, and J. A. Burgoyne, “Studying large plainchant corpora using Chant21,” in *7th International Conference on Digital Libraries for Musicology*. Montréal QC Canada: ACM, Oct. 2020, pp. 40–44.
- [26] T. F. Kelly, “Notation I,” in *The Cambridge History of Medieval Music*. Cambridge University Press, 2018, vol. 1, pp. 236–262.
- [27] H. Schaffrath, “The Essen folksong collection in the humdrum kern format,” Center for Computer Assisted Research in the Humanities, Tech. Rep., 1995.
- [28] L. Erk and F. M. Böhme, *Deutscher Liederhort: Auswahl der vorzüglicheren deutschen Volkslieder, nach Wort und Weise aus der Vorzeit und Gegenwart*. Leipzig: Breitkopf und Härtel, 1893, vol. 1.
- [29] F. Densmore, *Teton Sioux Music*, ser. Bulletin 61 of the Bureau of American Ethnology, Smithsonian Institution. Washington, U.S.A.: Government Printing Office, 1918, no. 61.
- [30] D. Shanahan and E. Shanahan, “The Densmore collection of Native American songs: A new corpus for studies of effects of geography, language and social function on folk song,” in *Proceedings for the 13th International Conference for Music Perception and Cognition*, Seoul, 2014, pp. 206–208.
- [31] I. Jolliffe, *Principal Component Analysis*, 2nd ed., ser. Springer Series in Statistics. Springer, 2002.
- [32] R. M. Gray, “Toeplitz and circulant matrices: A review,” *Foundations and Trends in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.
- [33] J. Novembre and M. Stephens, “Interpreting principal component analyses of spatial population genetic variation,” *Nature Genetics*, vol. 40, no. 5, pp. 646–649, May 2008.
- [34] J. M. Antognini and J. Sohl-Dickstein, “PCA of high dimensional random walks with comparison to neural network training,” in *32nd Conference on Neural Information Processing Systems*, Montréal, Canada, Jun. 2018.
- [35] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” *IEEE Transactions on Computers*, p. 4, 1974.
- [36] G. Strang, “The discrete cosine transform,” *SIAM Review*, vol. 41, no. 1, pp. 135–147, 1999.
- [37] K. R. Rao and P. C. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Boston: Academic Press, 1990.
- [38] J. Moore, H. Ahmed, and R. Antia, “High dimensional random walks can appear low dimensional: Application to influenza H3N2 evolution,” *Journal of Theoretical Biology*, vol. 447, pp. 56–64, Jun. 2018.
- [39] B. Cornelissen, W. Zuidema, and J. A. Burgoyne, “Mode classification and natural units in plainchant,” in *Proceedings of the 21st International Conference on Music Information Retrieval (ISMIR 2020)*, Montréal, Canada, 2020, p. 869–875.
- [40] L. Cavalli-Sforza, P. Menozzi, and A. Piazza, “Demic expansions and human evolution,” *Science*, vol. 259, no. 5095, pp. 639–646, Jan. 1993.
- [41] E. Lorch, “Visualizing deep network training trajectories with PCA,” in *Proceedings of the 33rd International Conference on Machine Learning*, New York, U.S.A., 2016, p. 5.

CONTROLLABLE DEEP MELODY GENERATION VIA HIERARCHICAL MUSIC STRUCTURE REPRESENTATION

Shuqi Dai **Zeyu Jin** **Celso Gomes** **Roger B. Dannenberg**
Carnegie Mellon university Adobe Inc. Adobe Inc. Carnegie Mellon University
shuqid@cs.cmu.edu zej@adobe.com cegomes@adobe.com rbd@cs.cmu.edu

ABSTRACT

Recent advances in deep learning have expanded possibilities to generate music, but generating a customizable full piece of music with consistent long-term structure remains a challenge. This paper introduces *MusicFrameworks*, a hierarchical music structure representation and a multi-step generative process to create a full-length melody guided by long-term repetitive structure, chord, melodic contour, and rhythm constraints. We first organize the full melody with section and phrase-level structure. To generate melody in each phrase, we generate rhythm and basic melody using two separate transformer-based networks, and then generate the melody conditioned on the basic melody, rhythm and chords in an auto-regressive manner. By factoring music generation into sub-problems, our approach allows simpler models and requires less data. To customize or add variety, one can alter chords, basic melody, and rhythm structure in the music frameworks, letting our networks generate the melody accordingly. Additionally, we introduce new features to encode musical positional information, rhythm patterns, and melodic contours based on musical domain knowledge. A listening test reveals that melodies generated by our method are rated as good as or better than human-composed music in the POP909 dataset about half the time.

1. INTRODUCTION

Music generation is an important component of computational and AI creativity, leading to many potential applications including automatic background music generation for video, music improvisation in human-computer music performance and customizing stylistic music for individual music therapy, to name a few. While works such as MelNet [1] and JukeBox [2] have demonstrated a degree of success in generating music in the audio domain, the majority of the work is in the symbolic domain, i.e., the score, as this is the most fundamental representation of music composition. Research has tackled this question from many angles, including monophonic melody genera-

tion [3], polyphonic performance generation [4] and drum pattern generation [5]. This paper focuses on melody generation, a crucial component in music writing practice.

Recently, deep learning has demonstrated success in capturing implicit rules about music from the data, compared to conventional rule-based and statistical methods [4, 6, 7]. However, there are three problems that are difficult to address: (1) Modeling larger scale music structure and multiple levels of repetition as seen in popular songs, (2) Controllability to match music to video or create desired tempo, styles, and mood, and (3) Scarcity of training data due to limited curated and machine-readable compositions, especially in a given style. Since humans can imitate music styles with just a few samples, there is reason to believe there exists a solution that enables music generation with few samples as well.

We aim to explore automatic melody generation with multiple levels of structure awareness and controllability. Our focus is on (1) addressing structural consistency inside a phrase and on the global scale, and (2) giving explicit control to users to manipulate melody contour and rhythm structure directly. Our solution, *MusicFrameworks*, is based on the design of hierarchical music representations we call *music frameworks* inspired by Hiller and Ames [8]. A music framework is an abstract hierarchical description of a song, including high-level music structure such as repeated sections and phrases, and lower-level representations such as rhythm structure and melodic contour. The idea is to represent a piece of music by music frameworks, and then learn to generate melodies from music frameworks. Controllability is achieved by editing the music frameworks at any level (song, section and phrase); we also present methods that generate these representations from scratch. *MusicFrameworks* can create long-term music structures, including repetition, by factoring music generation into sub-problems, allowing simpler models and requiring less data.

Evaluations of the *MusicFrameworks* approach include objective measures to show expected behavior and subjective assessments. We compare human-composed melodies and melodies generated under various conditions to study the effectiveness of music frameworks. We summarize our contributions as follows: (1) devising a hierarchical music structure representation and approach called *MusicFrameworks* capable of capturing repetitive structure at multiple levels, (2) enabling controllability at multiple levels of abstraction through music frameworks, (3) a set of methods



that analyze a song to derive music frameworks that can be used in music imitation and subsequent deep learning processes, (4) a set of neural networks that generate a song using the *MusicFrameworks* approach, (5) useful musical features and encodings to introduce musical inductive biases into deep learning, (6) comparison of different deep learning architectures for relatively small amounts of training data and a sizable listening test evaluating the musicality of our method against human-composed music.

2. RELATED WORK

Automation of music composition with computers can be traced back to 1957 [9]. Long before representation learning, musicians looked for models that explain the generative process of music [8]. Early music generation systems often relied on generative rules or constraint satisfaction [8, 10–12]. Subsequent approaches replaced human learning of rules with machine learning, such as statistical models [13] and connectionist approaches [14]. Now, deep learning has emerged as one of the most powerful tools to encode implicit rules from data [4, 15–18].

One challenge of music modelling is capturing repetitive patterns and long-term dependencies. There are a few models using rule-based and statistical methods to construct long-term repetitive structure in classical music [19] and pop music [20, 21]. Machine learning models with memory and the ability to associate context have also been popular in this area and include LSTMs and Transformers [4, 6, 22, 23], which operate by generating music one or a few notes at a time, based on information from previously generated notes. These models enable free generation and motif continuation, but it is difficult to control the generated content. StructureNet [3], PopMNet [24] and Racchmaninof [19] are more closely related to our work in that they introduce explicit models for music structure.

Another thread of work enables a degree of controllability by modeling the distribution of music via an intermediate representation (embedding). One such approach is to use Generative Adversarial Networks (GANs) to model the distribution of music [25–27]. GANs learn a mapping from a point z sampled from a prior distribution to an instance of generated music x and hence represents the distribution of music with z . Another method is the Autoencoder, consisting of an encoder transforming music x into embedding z and a decoder that reconstructs music x again from embedding z . The most popular models are Variational Auto-Encoders (VAE) and their variants [28–33]. These models can be controlled by manipulating the embedding, for example, mix-and-matching embeddings of different pitch contours and rhythms [29, 30, 34]. However, a high-dimensional continuous vector has limited interpretability and thus is difficult for a user to control; it is also difficult to model full-length music with a simple fixed-length representation. In contrast, our approach uses a hierarchical music representation (i.e., *music framework*) as an “embedding” of music that encodes long-term dependency in a form that is both interpretable and controllable.

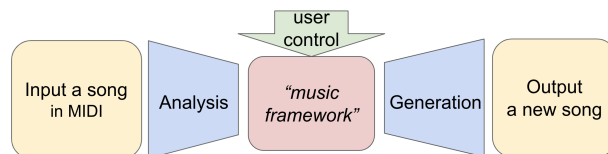


Figure 1. Architecture of *MusicFrameworks*.

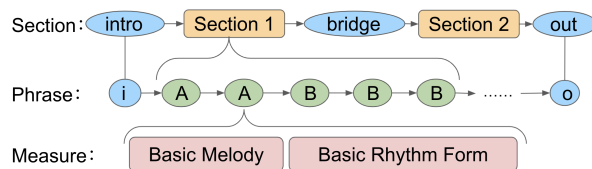


Figure 2. An example music framework.

3. METHOD

We describe a controllable melody generation system that uses hierarchical music representation to generate full-length pop song melodies with multi-level repetition and structure. As shown in Figure 1, a song is input in MIDI format. We analyze its *music framework*, which is an abstracted description of the music ideas of the input song. Then we use the music framework to generate a new song with deep learning models.

Our work is with pop music because structures are relatively simple and listeners are generally familiar with the style and thus able to evaluate compositions. We use a Chinese pop song dataset, POP909 [35], and use its cleaned version [36] (with more labeling and corrections) for training and testing in this paper. We further transpose all the major songs’ key signatures into C major. We use integers 1–15 to represent scale degrees in C3–C5, and 0 to represent a rest. For rhythm, we use the 16th note as the minimum unit. A note in the melody is represented as (p, d) , where p is the pitch number from 0 to 15, and d is duration in sixteenths. For chord progressions, we use integers 1–7 to represent seven scale degree chords in the major mode. (We currently work only with triads, and convert seventh chords into corresponding triads).

3.1 Music Frameworks Analysis

As shown in Figure 2, a music framework contains two parts: (1) section and phrase-level structure analysis re-

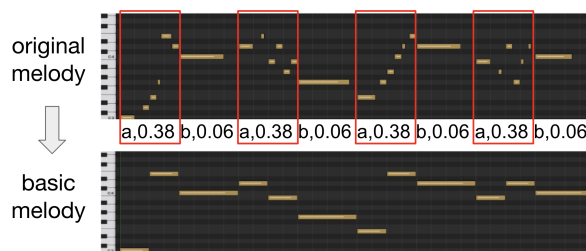


Figure 3. An example melody and its basic melody. The basic rhythm form also appears below the original melody as “a,0.38 b,0.06, ...” indicating similarity and complexity.

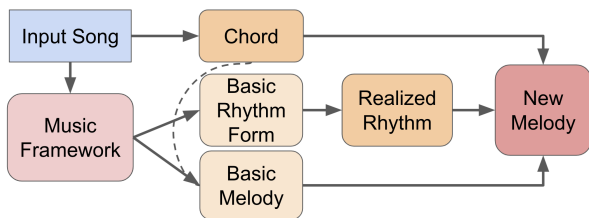


Figure 4. Generation process from music frameworks within each phrase.

sults; (2) basic melody and basic rhythm form within each phrase. A phrase is a small-scale segment that usually ranges from 4 to 16 measures. Phrases can be repeated as in AABBB shown in Figure 2. Sections contain multiple phrases, e.g., the illustrated song has an intro, a main theme section (phrase A as verse and phrase B as chorus), a bridge section followed by a repeat of the theme, and an outro section, which is a typical pop song structure. We extract the section and phrase structure based on finding approximate repetitions following the work of [36].

Within each phrase, the *basic melody* is an abstraction of melody and contour. Basic melody is a sequence of half notes representing the most common pitch in each 2-beat segment of the original phrase (see Figure 3). The *basic rhythm form* consists of a per-measure descriptor with two components: a pattern label based on a rhythm similarity measure [36] (measures with matching labels are similar) and a numerical rhythmic complexity, which is simply the number of notes divided by 16.

With the analysis algorithm, we can process a music dataset such as POP909 for subsequent machine learning and music generation. The music frameworks enable controllability via examples in which a user can also mix and match different music frameworks from multiple songs. For example, a new song can be generated using the structure from song A, basic melody from song B, and basic rhythm form from song C. Users can also edit or directly create a music framework for even greater control. Alternatively, we also created generative algorithms to create new music frameworks without any user intervention as described in subsequent sections.

3.2 Generation Using Music Frameworks

At the top level, section and phrase structure can be provided by a user or simply selected from a library of already analyzed data. We considered several options for imitation at this top level: (1) copy the first several measures of melody from the previously generated phrase (teacher forcing mode) and then complete the current phrase; (2) use the same or similar basic melody from the previous phrase to generate an altered melody with a similar melodic contour; (3) use the same or similar basic rhythm form of the previous phrase to generate a similar rhythm. These options leave room for users to customize their personal preferences. In this study, we forgo all human control by randomly choosing between the first and second option.

At the phrase level, as shown in Figure 4, we first generate a basic melody (or a human provides one). Next, we

generate rhythm using the basic rhythm form. Finally, we generate a new melody given the basic melody, generated rhythm, and chords copied from the input song.

3.2.1 Basic Melody Generation

We use an auto-regressive approach to generate basic melodies. The input $x_i = (\text{pos}_i, c_i, \dots)$ ($i \in 1, \dots, n$) is a set of features that guides basic melody generation where pos_i is the positional feature of the i^{th} note and c_i represents contextual chord information (neighboring chords). We denote p_i of the i^{th} note. Here we fix the duration of each note in the basic melody to the half-note as in the analysis algorithm described in Section 3.1. c_i contains the previous, current and next chords and their lengths for the i^{th} note. pos_i includes the position of the i^{th} note in the current phrase and a long-term positional feature indicating whether the current phrase is at the end of a section or not.

3.2.2 Network Architecture

We use an auto-regressive model based on Transformer and LSTM. The architecture (Figure 5) consists of an encoder and a decoder. The encoder has two layers of transformers that learn a feature representation of the inputs (e.g. positional encodings and chords). The decoder concatenates the encoded representation and the last predicted note as input and passes them through one unidirectional LSTM followed by two layers of 1D convolutions of kernel size 1. Both the input and the last predicted notes to the decoder are passed through a projection layer (aka. a dense layer) respectively before they are processed by the network. The final output is the next note predicted by the decoder via categorical distribution $Pr(p_i|X, p_1, \dots, p_{i-1})$. We also tried using other deep neural network architectures such as a pre-trained full Transformer with random masking (described in Section 4.1) for comparison.

3.2.3 Sampling with Dynamic Time Warping Control

In the sampling stage, we tried three ways to autoregressively generate the basic melody sequence: (1) randomly sample from the estimated posterior distribution of p_i at each step; (2) randomly sample 100 generated sequences and pick the one with highest overall estimated probability; (3) beam search sampling according to the estimated probability. Apart from the above three sampling methods, we also want to control the basic melody contour shape in order to generate similar or repeated phrases. We use a melody contour rating function (based on Dynamic Time Warping) [21] to estimate the contour similarity between two basic melodies. When we want to generate a repetition phrase that has a similar basic melody compared to a previous phrase, we estimate the contour similarity rating between the generated basic melody and the reference basic melody. We only accept basic melodies with a similarity above a threshold of 0.7.

3.2.4 Realized Rhythm Generation

We now turn to the lower level of music generation that transforms music frameworks into realized songs. The first

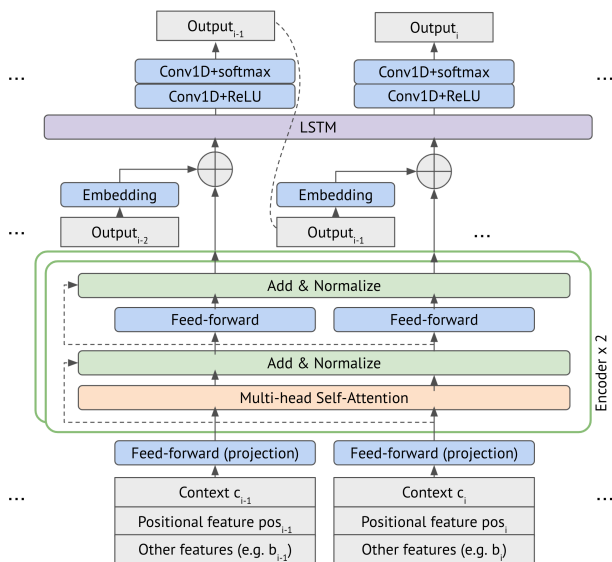


Figure 5. Transformer-LSTM architecture for melody, basic melody and rhythmic pattern generation.

step is to determine the note onsets, namely the rhythm. Instead of generating note onset time one by one, we generate 2-beat rhythm patterns, which more readily encode rhythmic patterns and metrical structure. It is also easier to model (and apparently to learn) similarity using rhythm patterns than with sequences of individual durations.

We generate 2-beat patterns sequentially under the control of a basic rhythm form. There are 256 possible rhythm patterns with a 2-beat length using our smallest subdivision of sixteenthths. For each rhythm pattern r_i , the input of the rhythm generation model is $x_i = (r_{i-1}, \text{brf}_i, \text{pos}_i)$, where r_{i-1} is the previously generated rhythm pattern, brf_i is the index of the first measure similar to it (or the current measure if there is no previous reference) and the current measure complexity; pos_i contains three positional components: (1) the position of the i^{th} pattern in the current phrase; (2) a long-term positional feature indicating whether the current phrase is at the end of a section or not; (3) whether the i^{th} rhythm pattern starts at the barline or not. We also use a Transformer-LSTM architecture (Figure 5), but with different model settings (size). In the sampling stage, we use beam search.

3.2.5 Realized Melody Generation

We generate melody from a basic melody, a rhythm and a chord progression using another Transformer-LSTM architecture similar to generating basic melody in Figure 5. In this case, the index i represents the i^{th} note determined by the rhythm. The input feature x_i also includes the current note’s duration, the current chord, the basic melody pitch, and three positional features for multiple-level structure guidance: the two positional features for basic melody generation (Section 3.2.1) and the offset of the current note within the current measure. We also experimented with other deep neural network architectures described in Section 4.1 for comparison. To sample a good sounding melody, we randomly generate 100 sequences by sampling

	Basic Melody	Rhythm	Melody
Trans-LSTM	38.7%	50.1%	55.2%
LSTM	39.8%	43.6%	51.2%
Transformer	30.9%	25.8%	39.3%
No M.F.	NA	33.1%	37.4%

Table 1. Validation Accuracy of different model architectures. “No M.F.” means no music frameworks used here.

the autoregressive model. We pick the one with the highest overall estimated probability. More details about the network are in Section 4.1.

4. EXPERIMENT AND EVALUATION

4.1 Model Evaluation and Comparison

As a model-selection study, we compared the ability of different deep neural network architectures implementing *MusicFrameworks* to predict the next element in the sequence. *Basic Melody Accuracy* is the percent of correct predictions of the next pitch of the basic melody (half notes). *Rhythm Accuracy* is the percent of correctly predicted 2-beat rhythm patterns. *Melody Accuracy* is the accuracy of next pitch prediction.

We used 4188 phrases from 528 songs in major mode from the POP909 dataset, using 90% of them as training data and the other 10% for validation. The first line in Table 1 represents the Transformer-LSTM models introduced in Section 3. In all three networks, the projection size and feed forward channels are 128; there are 8 heads in the multi-head encoder attention layer; LSTM hidden size is 64; dropout rate for basic melody and realized melody generation is 0.2, dropout rate for rhythm generation is 0.1; decoder input projection size is 8 for rhythm generation and 17 for others. For learning rate, we used the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-6}$, and the same formula in [22] to vary the learning rate over the course of training, with 2000 warmup steps.

We compared this model with several alternatives: the second model is a bi-directional LSTM followed by a unidirectional LSTM (model size is 64 in both). The third model is a Transformer with two layers of encoder and two layers of decoder (with same parameter settings as Transformer-LSTM), and we first pre-trained the encoder with 10% of random masking of input (similar to training in BERT [37]), and then trained the encoder and decoder together. No music frameworks (the fourth line) means generate without basic melody or basic rhythm form, using a Transformer-LSTM model. The results in Table 1 show that the Transformer-LSTM achieved the best accuracy. The full Transformer model performed poorly on this relatively small dataset due to overfitting. Also, in both rhythm and melody generation, the *MusicFrameworks* approach significantly improves the model accuracy.

4.2 Objective Evaluation

We use Transformer-LSTM model for all further evaluations. First, we examine whether music frameworks pro-



Figure 6. This is a generated melody (yellow piano roll) from our system following the input basic melody (blue frame piano roll).

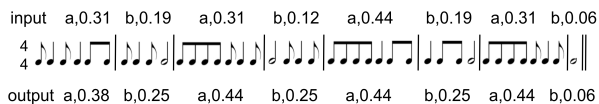


Figure 7. A generated rhythm from our system given the input basic rhythm form. The analyzed basic rhythm form of the output is very similar to the input.

mote controllability. We aim to show that given a basic melody and rhythm form as guidance, the model can generate a new melody that follows the contour of the basic melody, and has a similar rhythm form (Figure 6 and 7).

For this “sanity check,” we randomly picked 20 test songs and generated 500 alternative basic melodies and rhythm forms. After generating and analyzing 500 phrases, we found the analyzed basic melodies match the target (input) basic melodies with an accuracy of 92.27%; the accuracy of rhythm similarity labels is 94.63%; the rhythmic complexity matches the target (within 0.2) 81.79% of the time. Thus, these aspects of melody are easily controllable.

Previous work [38] has shown that pitch and rhythm distributions are related to different levels of long-term structure. We confirmed that our generation exhibits similar structure-related distributions to that of the POP909 dataset. For example, the probability of a generated tonic at end of a phrase is 48.28%, and at the end of a section is 87.63%, while in the training data the probabilities are 49.01% (phrase-end) and 86.57% (section-end).

4.3 Subjective Evaluation

4.3.1 Design of the listening evaluation

We conducted a listening test to evaluate the generated songs. To avoid listening fatigue, we presented sections lasting about 1 minute and containing at least 3 phrases. We randomly selected 6 sections from different songs in the validation set as seeds and then generated melodies based on conditions 1–6 presented in Table 2. To render audio, each melody is mixed with the original chords played as simple block triads via a piano synthesizer. For each section and each condition, we generated at least 2 versions, with 105 generated sections in total.

In each rating session, a listener first enters information about their music background and then provides ratings for six pairs of songs. Each pair is generated from the same input seed song using different generation conditions (see Table 2). For each pair, the listener answers: (1) whether they heard the songs before the survey (yes or no); (2) how much they like the melody of the two songs (integer from 1 to 5); and (3) how similar are the two songs’ melodies (integer from 1 to 5). We also embedded one validation test in which a human-composed song and a randomized

song are provided to help filter out careless ratings.

4.3.2 Results and discussion

We distributed the survey on Chinese social media and collected 274 listener reports. We removed invalid answers following the validation test and a few other criteria. We ended up with 1212 complete pairs of ratings from 196 unique listeners. The demographics information about the listeners are as follows:

Gender male: 120, female: 75, other: 1;

Age distribution 0-10: 0, 11-20: 17, 21-30: 149, 31-40: 28, 41-50: 0, 51-60: 2, >60: 0;

Music proficiency levels lowest (listen to music < 1 hour/week): 16, low (listen to music 1–15 hours/week): 62, medium (listen to music > 15 hours/week): 21, high (studied music for 1–5 years): 52, expert (> 5 years of music practice): 44;

Nationality Chinese: 180, Others: 16 (note that the POP909 dataset is primarily Chinese pop songs, and listeners who are more familiar with this style are likely to be more reliable and discriminating raters.)

Figure 8 shows a distribution of ratings for the seven paired conditions in Table 2. In each pair, we show two bar plots with mean and standard deviation overlaid: the left half shows the distribution of ratings in the first condition and the right half shows those in the second condition. The first three pairs compare music generation with and without a music framework as an intermediate representation. The first two pairs at the bottom compare music with an automatically generated basic melody and rhythm to music using the basic melody and rhythm from a human-composed song. The last two pairs show the ratings of our method compared to music in the POP909 dataset. We also conducted a paired T-test to check the significance against the hypothesis that the first condition is not preferred over the second condition, shown under the distribution plot.

In addition, we derived listener preference based on the relative ratings, summarized in Figure 9. This visualization provides a different view from ratings as it shows how frequently one condition is preferred over the other or there is no preference (equal ratings). Based on these two plots, we point out the following observations:

- Basic melody and basic rhythm form improve the quality of generated melody. Indicated by low p-values and strong preference in “1 vs 3”, “2 vs 3” and “4 vs 5,” generating basic melody and basic rhythm before melody generation has higher ratings than generating melody without these music framework representations.
- Melody generation conditioned on generated basic melody and basic rhythm has similar ratings to melody generated from human-composed music’s basic melody and basic rhythm form. This observation can be derived from similar distribution and near random preference distribution in “1 vs 2” and “1 vs 4,” indicating that preference for the generated basic melody and rhythm form are close to those of music in our dataset.
- Although both distribution tests suggest that human-composed music has higher ratings than generated music

	R.Melody	Basic Melody	Rhythm
0	copy	copy	copy
1	gen	copy	copy
2	gen	gen	copy
3	gen	without	copy
4	gen	copy	gen with BRF
5	gen	copy	gen without BRF
6	gen	gen	gen with BRF

Table 2. Seven evaluation conditions. Group 0 is human-composed. R.Melody: realized melody; gen: generated from our system; BRF: Basic Rhythm Form; copy: directly copying that part from the human-composed song; without: not using music frameworks.

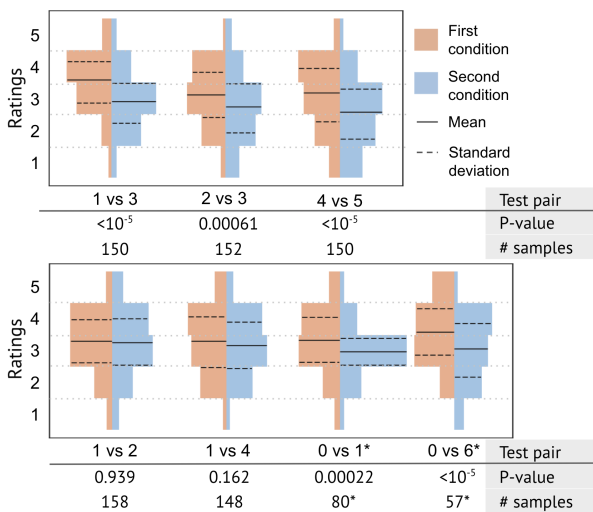


Figure 8. Rating distribution comparison for each paired groups. *For conditions 0 vs 1 and 0 vs 6, we removed the cases where the listeners indicated having heard the song before.

in test pairs “0 vs 1” and “0 vs 6” (and this is statistically significant), the preference test suggests that around half of the time the generated music is as good as or better than human-composed music, indicating the usability of the *MusicFrameworks* approach.

To understand the gap between our generated music and human-composed music, we look into the comments written by listeners and summarize our findings below:

- Since sampling is used in the generative process, there is a non-zero chance that a poor note choice may be generated. Though this does not affect the posterior probability significantly, it degrades the subjective rating. Repeated notes also have an adverse effect on musicality with a lesser influence on posterior probability.
- *MusicFrameworks* uses basic melody and rhythm form to control long-term dependency, i.e., phrases that are repetitions or imitations share the same or similar music framework; however, the generated melody has a chance to sound different due to the sampling process. A human listener can distinguish a human-composed song from a machine-generated song by listening for exact repetition.
- Basic melody provides more benefit for longer phrases. For short phrases (4-6 bars), generating melodies from

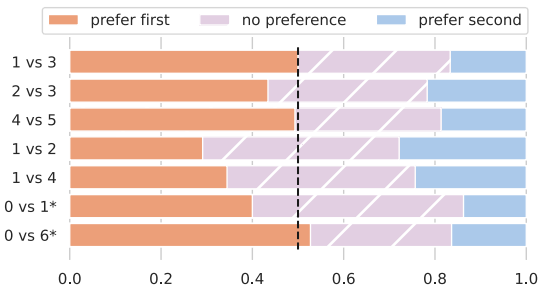


Figure 9. Preference distribution for each paired groups. *For conditions 0 vs 1 and 0 vs 6, we removed the cases where the listeners indicated having heard the song before.

- scratch is competitive with generating via basic melody.
- The human-composed songs used in this study are from the most popular ones in Chinese pop history. Even though raters may think they do not recognize the song, there is a chance that they have heard it. A large portion of the comments suggest that a lot of the test music sounds great and it was an enjoyable experience working on these surveys. However, some listeners point out that concentrating on relatively long excerpts was not a natural listening experience.

5. CONCLUSION

MusicFrameworks is a deep melody generation system using hierarchical music structure representations to enable a multi-level generative process. The key idea is to adopt an abstract representation, *music frameworks*, including long-term repetitive structures, phrase-level *basic melodies* and *basic rhythm forms*. We introduced analysis algorithms to obtain music frameworks from songs. We created a neural network that generates basic melody and additional networks to generate melodies. We also designed musical features and encodings to better introduce musical inductive bias into deep learning models.

Both objective and subjective evaluations show the importance of having music frameworks. About 50% of the generated songs are rated as good as or better than human-composed songs. Another important feature of the *MusicFrameworks* approach is *controllability* through manipulation of music frameworks, which can be freely edited and combined to guide compositions.

In the future, we hope to develop more intelligent ways to analyze music and music frameworks supporting a richer musical vocabulary, generation of harmony and polyphonic generation. We believe that hierarchical and structured representations offer a way to capture and imitate musical style, offering interesting new research opportunities.

6. ACKNOWLEDGMENTS

We would like to thank Dr. Stephen Neely for his insights on musical rhythm.

7. REFERENCES

- [1] S. Vasquez and M. Lewis, “Melnet: A generative model for audio in the frequency domain,” *arXiv preprint arXiv:1906.01083*, 2019.
- [2] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.
- [3] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdallah, M. Selvi, E. Newton-Rex, and K. Webster, “Structurenets: Inducing structure in generated melodies.” in *Proc. of 19th International Conference on Music Information Retrieval, ISMIR*, 2018, pp. 725–731.
- [4] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music transformer,” *arXiv preprint arXiv:1809.04281*, 2018.
- [5] I.-C. Wei, C.-W. Wu, and L. Su, “Generating structured drum pattern using variational autoencoder and self-similarity matrix.” in *Proc. of 20th International Conference on Music Information Retrieval, ISMIR*, 2019, pp. 847–854.
- [6] Y.-S. Huang and Y.-H. Yang, “Pop music transformer: Generating music with rhythm and harmony,” *arXiv preprint arXiv:2002.00212*, 2020.
- [7] S. H. Hakimi, N. Bhonker, and R. El-Yaniv, “Bebopnet: Deep neural models for personalized jazz improvisations,” in *Proc. of the 21st International Society for Music Information Retrieval Conference, ISMIR*. ISMIR, 2020.
- [8] L. Hiller, C. Ames, and R. Franki, “Automated composition: An installation at the 1985 international exposition in tsukuba, japan,” *Perspectives of New Music*, vol. 23, no. 2, pp. 196–215, 1985.
- [9] L. Hiller and L. Isaacson, *Experimental Music: Composition with an Electronic Computer*. New York: McGraw-Hill, 1959.
- [10] D. Cope, *Computers and musical style*. Oxford University Press Oxford, 1991, vol. 6.
- [11] —, *The Algorithmic Composer*. AR Editions, Inc., 2000.
- [12] —, *Computer models of musical creativity*. MIT Press Cambridge, 2005.
- [13] W. Schulze and B. Van Der Merwe, “Music generation with markov models,” *IEEE Annals of the History of Computing*, vol. 18, no. 03, pp. 78–85, 2011.
- [14] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription,” *arXiv preprint arXiv:1206.6392*, 2012.
- [15] J.-P. Briot, G. Hadjeres, and F. Pachet, *Deep learning techniques for music generation*. Springer, 2019, vol. 10.
- [16] F. Liang, “Bachbot: Automatic composition in the style of bach chorales,” *University of Cambridge*, vol. 8, pp. 19–48, 2016.
- [17] G. Hadjeres, F. Pachet, and F. Nielsen, “Deepbach: a steerable model for bach chorales generation,” in *Proc. of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1362–1371.
- [18] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by convolution,” *arXiv preprint arXiv:1903.07227*, 2019.
- [19] T. Collins and R. Laney, “Computer-generated stylistic compositions with long-term repetitive and phrasal structure,” *Journal of Creative Music Systems*, vol. 1, no. 2, 2017.
- [20] A. Elowsson and A. Friberg, “Algorithmic composition of popular music,” in *the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*, 2012, pp. 276–285.
- [21] S. Dai, X. Ma, Y. Wang, and R. B. Dannenberg, “Personalized popular music generation using imitation and structure,” *arXiv preprint arXiv:2105.04709*, 2021.
- [22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [23] C. Payne, “Musenet,” *OpenAI, openai.com/blog/musenet*, 2019.
- [24] J. Wu, X. Liu, X. Hu, and J. Zhu, “Popmnet: Generating structured pop music melodies using neural networks,” *Artificial Intelligence*, vol. 286, p. 103303, 2020.
- [25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial networks,” *ArXiv*, vol. abs/1406.2661, 2014.
- [26] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient,” in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [27] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018.
- [28] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

- [29] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International Conference on Machine Learning*, 2018.
- [30] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," *arXiv preprint arXiv:1906.03626*, 2019.
- [31] H. H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *Proc. of 21st International Conference on Music Information Retrieval, ISMIR*, 2020.
- [32] L. Kawai, P. Esling, and T. Harada, "Attributes-aware deep music transformation," in *Proc. of the 21st International Society for Music Information Retrieval Conference, ISMIR*. ISMIR, 2020.
- [33] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proc. of 21st International Conference on Music Information Retrieval, ISMIR*, 2020.
- [34] K. Chen, C.-i. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm," in *Proc. of the 21st International Society for Music Information Retrieval Conference*. ISMIR, 2020.
- [35] Z. Wang*, K. Chen*, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proc. of 21st International Conference on Music Information Retrieval, ISMIR*, 2020.
- [36] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," in *Proc. of the 2020 Joint Conference on AI Music Creativity (CSMC-MuMe)*, 2020.
- [37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.
- [38] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," in *in Proc. of the 2020 Joint Conference on AI Music Creativity (CSMC-MuMe 2020)*, 2020.

MSTRE-NET: MULTISTREAMING ACOUSTIC MODELING FOR AUTOMATIC LYRICS TRANSCRIPTION

Emir Demirel
Queen Mary
University of London
e.demirel@qmul.ac.uk

Sven Ahlbäck
Doremir Music Research AB
sven.ahlback@doremir.com


Simon Dixon
Queen Mary
University of London
s.e.dixon@qmul.ac.uk

ABSTRACT

This paper makes several contributions to automatic lyrics transcription (ALT) research. Our main contribution is a novel variant of the Multistreaming Time-Delay Neural Network (MTDNN) architecture, called MSTRE-Net, which processes the temporal information using multiple streams in parallel with varying resolutions keeping the network more compact, and thus with a faster inference and an improved recognition rate than having identical TDNN streams. In addition, two novel preprocessing steps prior to training the acoustic model are proposed. First, we suggest using recordings from both monophonic and polyphonic domains during training the acoustic model. Second, we tag monophonic and polyphonic recordings with distinct labels for discriminating non-vocal silence and music instances during alignment. Moreover, we present a new test set with a considerably larger size and a higher musical variability compared to the existing datasets used in ALT literature, while maintaining the gender balance of the singers. Our best performing model sets the state-of-the-art in lyrics transcription by a large margin. For reproducibility, we publicly share the identifiers to retrieve the data used in this paper.

1. INTRODUCTION

Empirical studies show that it is a challenging task even for human listeners to recognize sung words, and this is more challenging than speech, due to a number of performance, environment and listener related factors [1]. Thus the automatic retrieval of sung words through machine listening, i.e. automatic lyrics transcription (ALT), can potentially be impactful in easing some of the time consuming processes involved in composing music, audio/video/music score captioning and editing, lyrics alignment, music catalogue creation, etc. Despite its

ED received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 765068. 



© Emir Demirel, Sven Ahlbäck, Simon Dixon. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Emir Demirel, Sven Ahlbäck, Simon Dixon, "MSTRE-Net: Multistreaming Acoustic Modeling for Automatic Lyrics Transcription", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

potential, the current state of lyrics transcription is far from being sufficiently robust to be leveraged in such applications.

With recent advances in automatic speech recognition (ASR) research and its successful adaptation to singing data, considerable improvements have been reported in ALT research [2–4]. In addition to this, newly released datasets have accelerated the development of the research field [5, 6]. Through these improvements, the prospect of applying ALT in the music industry has become more realistic, assuming that progress continues. Although promising results have been obtained for a cappella recordings [2, 4, 7], recognition rates drop considerably in the presence of instrumental accompaniment [8, 9].

From the perspective of ASR, music accompaniment can be regarded as noise since non-vocal music signals generally include minimal or no information relevant to lyrics transcription, while their presence in the spectral domain increases the confusions during prediction. For building more robust acoustic models against noisy environments, the multistream approach in ASR was introduced [10], inspired by how the acoustic signals are split into multiple frequency bands and processed in parallel in the human auditory system [11]. While previous research suggested using multiresolution feature processing [12, 13] or reconstruction of a multi-band latent representation through autoencoders [14] to achieve multistreaming ASR, the neural network architecture recently introduced in [15], Multistreaming Time-Delay Neural Network (MTDNN), proposes a simplified solution which is utilized in producing the state-of-the-art for hybrid / Deep Neural Network - Hidden Markov Model (DNN-HMM) based ASR [16, 17]. In this work, we propose a compact variant of MTDNN, referred to as MSTRE-Net, where streams are diversified by having different numbers of layers with the goal of reducing the number of trainable parameters (i.e. model complexity), and thus the inference times and improving the word recognition rates.

Additionally, we propose a number of other novel contributions for improving lyrics transcription performance. We suggest combined training of the acoustic model on both monophonic (e.g. *DAMP-Sing!* 300x30x2 [6]) and polyphonic (e.g. *DALI* [5]) recordings, which is shown to improve performance for both cases. Furthermore, we propose tagging monophonic and polyphonic utterances with separate *music* and *silence*

tokens explicitly. Our goal for this is to generate alignments that are more robust against disruptions in the decoding path, potentially caused by the musical accompaniment during the non-vocal frames.

One major challenge in ALT research has been publishing reproducible results, due to the lack of publicly available evaluation data [18]. Dabike and Barker [2] shared manually verified annotations for a subset of *DAMP* which have been utilized for evaluation in a cappella singing [4, 7]. The Jamendo (lyrics) dataset [19] consists of 20 contemporary polyphonic music recordings released under an open source license. Moreover, despite their limited nature in terms of size and musical variability, Hansen [20] and Mauch [21] have been among the two most commonly used evaluation sets for ALT. In addition to these, we present a new test set with 240 polyphonic recordings having a larger span of release dates and better singer gender balance in order to establish a more comprehensive lyrics transcription evaluation.

The rest of the paper is structured as follows: we begin with a summary of essential concepts in the state-of-the-art approach for hybrid-ASR. The following section explains how the proposed MTDNN architecture is constructed. Next, we give details of the data used in experiments, and introduce a new evaluation set. Finally, we describe the experimental setup and present results verifying our design choices through ablation tests.

2. BACKGROUND

ALT can be considered as analogous to Large Vocabulary Continuous Speech Recognition (LVCSR) for the singing voice. Similarly, the goal for ALT is predicting the most likely word sequence, \mathbf{w} , given a stream of acoustic observations, \mathbf{O} , which can be expressed in mathematical terms as follows:

$$\begin{aligned} \hat{\mathbf{w}} &= \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}|\mathbf{O}) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w})p(\mathbf{O}|\mathbf{w}) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} P(\mathbf{w}) \sum_{\mathbf{Q}} p(\mathbf{O}|\mathbf{Q})P(\mathbf{Q}|\mathbf{w}), \end{aligned} \quad (1)$$

where elements of \mathbf{Q} represent the phoneme classes¹. In Equation 1, $p(\mathbf{O}|\mathbf{Q})$ is obtained via the acoustic model. Phonemes are converted to word labels using a lexicon which defines a mapping between words and their phonemic representations. The raw word posteriors are then smoothed with the language model, $P(\mathbf{w})$ for obtaining grammatically more plausible output transcriptions, $\hat{\mathbf{w}}$.

According to the probabilistic approach of ASR, phonemes are represented with HMMs where a transition between connected phone states occurs at every time step [22]. In our system, we employ the *Kaldi* toolkit [23], an open-source ASR framework that represents HMM states using Weighted Finite State Transducers (WFST)

¹ A phoneme is the basic sonic unit of speech. In linguistics, words are considered to be composed of sequences of phonemes.

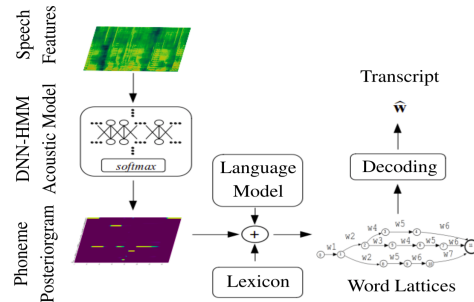


Figure 1. DNN-HMM based ASR at operation

[24]. In operation, a WFST graph is generated through composing posteriors retrieved from the acoustic, language and pronunciation models. The resulting directed paths of states form a *lattice*, a weighted acyclic graphical structure which can represent multiple output hypotheses.

2.1 Lattice-free Maximum Mutual Information

Most recent ALT systems utilize the state-of-the-art hybrid DNN-HMM framework where the neural networks are trained in a sequence discriminative fashion [25]. More specifically, the best performing lyrics transcribers to date [2–4, 7, 8] use Lattice-free Maximum Mutual Information training (LF-MMI) [26], where network parameters are tuned w.r.t. the MMI objective:

$$\mathcal{F}_{MMI} = \sum_u \log \frac{p(\mathbf{O}_u|Q_u)^{\mathcal{K}} P(W_u)}{\sum_W p(\mathbf{O}_u|Q)^{\mathcal{K}} P(W)} \quad (2)$$

where $p(\mathbf{O}_u|Q_u)$ is the probability of observing an acoustic instance O in the utterance u , in Markovian phone state Q_u , and the $P(W)$'s are the word sequence probabilities [27]. Optimization w.r.t. MMI aims at maximizing the shared information between the reference and target sequences. More explicitly, the terms in the numerator are calculated per utterance, whereas the denominator is computed over the entire training set. Hence, the network parameters are updated to maximize the probabilities in the numerator and minimize the denominator. In other words, the goal of MMI training is to discriminate a certain acoustic observation with its given utterance.

3. MULTISTREAMING TIME-DELAY NEURAL NETWORKS

The main body of MTDNN architectures consists of multiple streams of TDNN layers trained in parallel, where each stream has a unique time dilation rate (τ). Our proposed MTDNN variant differs from the original models [16, 17] by having different numbers of layers in the TDNN streams, depending on τ (Figures 2(b) and 2(c)).

Prior to the TDNN streams, input features \mathbf{x} are first processed by a single stream 2-D Convolutional Neural Network (CNN) in the front-end of the network,

$$\mathbf{h} = \textit{Stacked-2D-CNN}(\mathbf{x}) \quad (3)$$

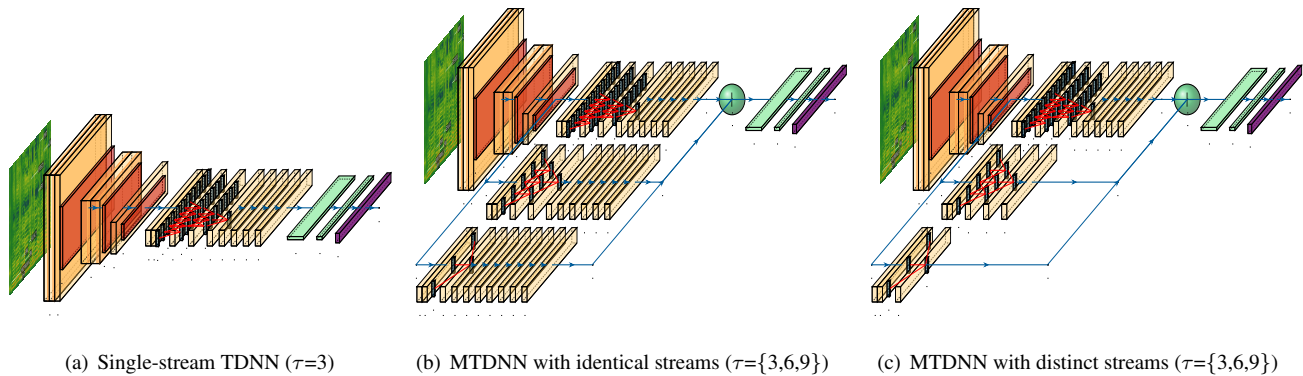


Figure 2. Different variants of TDNN architectures. From left-to-right, the orange, yellow and cyan blocks represent the front-end 2-D CNN, TDNN streams and final FC layers preceded by the purple softmax layer

where *Stacked-2D-CNN* stands for the stack of 2-D convolutional layers with 3×3 kernels. Inspired by [4], we apply subsampling on the height axis after each alternate layer with a factor of 2, in order to get compact embeddings, \mathbf{h} , which are then fed into multiple streams of TDNNs². Each stream of TDNNs has a unique time dilation rate, τ , encoding information in different temporal resolutions,

$$\mathbf{z}_t^n = \text{Stacked-TDNN}(\mathbf{h} | \tau = t, N = n), \quad (4)$$

where \mathbf{z}_t^n are the latent variables at the output of the final (N^{th}) TDNN layer, and $t \in \mathbb{Z}$. These are concatenated and projected to the classification (*softmax*) layer by a pair of fully connected (*FC*) layers,

$$a(s) = \text{softmax}(2 \times \text{FC}(\text{Concat}(\mathbf{z}_{T_1}^N, \mathbf{z}_{T_2}^N, \dots, \mathbf{z}_{T_K}^N))), \quad (5)$$

where $a(s)$ is the activation of the *softmax* layer corresponding to the phoneme state s and K is the number of TDNN streams. We decide the number of layers per stream w.r.t. to the receptive field (*RF*) of the nodes at the top TDNN layer,

$$\text{RF}_{\mathbf{z}_t^N} = 2 \times l \times \tau \times N, \quad (6)$$

where l is the frame length of the acoustic feature vectors. Note that we include an additional 1-D convolutional layer just before the TDNN streams. This layer does not use dilation, in order not to skip any frames.

4. DATA

4.1 Training Set

The acoustic models of the previously presented ALT systems in the literature are built on either monophonic or polyphonic music recordings. In general, monophonic models are trained on the *DAMP^{train}* dataset [2, 4, 7], while *DALI* is utilized for polyphonic models [9]. We merge these two datasets, exploiting their size and musical

² A time-delay neural network consists of 1-D convolutional layers where the convolution is applied with frames that are dilated on the time axis [28]. In our architecture, we employ the factorized variant of TDNN introduced in [29].

variability. We curated the polyphonic subset of the dataset on the recordings from the most recent version (v2.0) of the *DALI* dataset [30], and included only those songs for which the Youtube links were available and still in use at the time of the audio retrieval.

4.2 Evaluation Sets

We perform model selection and optimization on the subsets of the *DAMP* and *DALI* datasets, representing monophonic and polyphonic domains respectively. For *DAMP^{test}*, we use the test split introduced in [2]. For testing the lyrics transcription performance on polyphonic recordings, we curated a new subset of *DALI-v1.0*, which we give the data selection procedure below. Finally, we evaluate our best performing model on the three benchmark datasets used in the literature, namely the Jamendo, Hansen and Mauch sets and provide a comparison with the state of the art in Section 6.5.

Set	Words	Uniq. Words	# Utt.	# Rec	# Singers	Avg. Utt. Dur.	Total Dur.
<i>LM-corpus</i>	13M	100k	2M	N/A	N/A	N/A	N/A
<i>DAMP^{train}</i>	686k	5.3k	80k	4.2k	3k	5sec	112h
<i>DALI^{train}</i>	1.1M	25.5k	227k	4.1k	1.5k	2.48sec	156h
<i>DAMP^{dev}</i>	4k	695	482	66	38	5.12sec	41min
<i>DALI^{dev}</i>	5.7k	941	1.7k	34	16	2.41sec	48min
<i>DAMP^{test}</i>	4.6k	840	479	70	40	6sec	48min
<i>DALI^{test}</i>	62.8k	4.2k	240	240	160	233sec	15.5h
<i>Jamendo</i>	5.7k	1k	20	20	20	216sec	72min
<i>Hansen</i>	2.8k	585	10	10	9	214sec	35min
<i>Mauch</i>	5.2k	820	20	20	18	245sec	82min

Table 1. Statistics of datasets used in experiments

For tuning the hyperparameters during evaluation, the language model scaling factor and the word insertion penalty, we have used the combination of the data from *DAMP^{dev}* split [2] and 20 recordings from *DALI-v2.0*³.

4.2.1 The DALI-test set

In this section, we give details of the curation procedure for the *DALI^{test}* set. We began from the subset presented in [31], which initially had 513 recordings and filtered it according to a number of criteria. Numerous audio samples were not retrievable from the links provided. We obtained the Youtube links through *automatic search*

³ This combined development set is denoted as *dev* in Section 6.

using relevant key words. We discarded songs where the automatically retrieved version was a live performance, had low audio quality or contained extra background speech sections unrelated to its corresponding lyrics. For consistency and fair evaluation, we did not include songs where the dominant language was not English. We allowed for an artist to have at most 5 songs. Among the remaining recordings, we manually selected a subset having a relatively balanced distribution of singers’ gender, official release dates over decades (see Figure 3) and variability in terms of singing styles, vocal effects and music genre. Lyrics were initially obtained from the annotations provided in [5] and manually verified following the steps explained in Section 5.1. The final version of $DALI^{test}$ consists of 240 recordings, which sets the largest test set for lyrics transcription with clean annotations. For open science, we publicly share the data identifiers, cleaned lyrics annotations and a tutorial to retrieve the corresponding Youtube links at “<https://github.com/emirdemirel/DALI-TestSet4ALT>”.

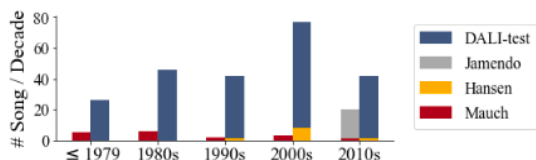


Figure 3. Songs per decade in ALT evaluation sets

5. EXPERIMENTAL SETUP

5.1 Lyrics Preprocessing

Prior to being utilized for training, raw lyrics data automatically retrieved from online resources (as in $DALI$) needs to be normalized, as the transcription rules applied by lyrics providers are not standardized. We remove all special ASCII characters except apostrophes. We convert numeric characters to their alphabetic correspondence. All text is converted to upper case. We observed several samples with erroneous hyphenation, explicit syllabification and repeating letters (possibly indicating longer uttered syllables or vowels). To cope with these, we apply automatic hyphenation correction and canonicalization using the standard open-source $NLTK$ tools⁴. The output lyrics are then verified and corrected manually.

5.2 Language and Pronunciation Models

Lyrics often contain uncommon words that are not very likely to exist in standard pronunciation dictionaries. For such words that are not in the lexicon, or out-of-vocabulary (OOV) words, we generate pronunciations using a pretrained grapheme-to-phoneme (G2P) converter [32]. In order to produce fair and reproducible results, we generate pronunciations for the OOV words in the evaluation sets as well, and do not skip these during evaluation. We utilize the commonly used CMU

⁴ These steps are potentially language specific.

English Pronunciation Dictionary⁵ as the lexicon and generate alternative pronunciations by duplicating the vowel phonemes for each word pronunciation, inspired by the improvements observed in [7, 33]. A 4-gram language model (LM) is constructed using the SRILM Toolkit [34]. We use the combination of the lyrics corpus in [2]⁶ and $DALI^{train}$. For scientific evaluation, we exclude any songs which overlap with those in the evaluation sets.

5.3 Discriminating Instrumental and Silent Regions

The hybrid DNN-HMM ASR framework approaches the continuous word recognition task essentially as a sequence decoding problem. Within this scope, the presence of instrumental accompaniment, especially during non-vocal regions, may disrupt the decoding path, potentially causing cumulative errors during transcription and alignment. Traditionally, non-speech regions are represented with a special silence token within the target class set during recognition. Here, we propose using separate tokens for the non-vocal instances in monophonic and polyphonic recordings. Prior to training, we associate these tokens with their corresponding silence/music instances by explicitly adding tags at the beginnings and the ends of the ground truth lyrics of each utterance (see Table 2). These tags are represented as words in the lexicon where their pronunciations correspond to the relevant silence token.

	w
Raw	$w_1 w_2 \dots w_N$
DAMP	<silence> $w_1 w_2 \dots w_N$ <silence>
DALI	<music> $w_1 w_2 \dots w_N$ <music>

Table 2. Music / silence tagging w.r.t. the dataset

For silence and music tagging, we exploit our training data. As we know the recordings in $DAMP$ and $DALI$ are monophonic and polyphonic respectively, we apply tagging w.r.t. the dataset. The pronunciations of these pseudo-word tags are represented with distinct phonemes in the lexicon.

5.4 Generating Phoneme Alignments

Since the neural network optimization is performed w.r.t. phoneme posteriors (as explained in Section 2.1), we need to extract their timings, i.e. *alignments*. To generate these, we train a triphone Gaussian Mixture Model (GMM) - HMM model on “singer adaptive” features [35], following the standard Kaldi recipe⁷. At this stage, we compute per-word pronunciation probabilities following the steps in [36], and retrain another triphone model using the updated lexicon transducer. Using this new model, we apply *forced alignment* [22] on the training data for generating the phoneme and word alignments.

⁵ Link: <https://github.com/Alexir/CMUdict/blob/master/cmudict-0.7b>.

⁶ This corpus contains lyrics from all the songs of the artists included in the Billboard charts between 2015-2018, plus the lyrics in $DAMP^{train}$.

⁷ We execute the GMM-HMM pipeline at <https://github.com/emirdemirel/ALTA>, which is almost the same procedure as the standard *librispeech* recipe, with tuned hyperparameters for singing data.

5.5 Neural Network Training

DNN training is based on the *Kaldi - chain* recipe. In the feature space, we use 40-band filterbank features extracted with a hop size of 10ms and window size of 30ms. To achieve singer-adaptive training, we utilize i-Vectors [37] which represent the singer identity information via global embedding vectors. Frame subsampling is applied with a factor of 3 in this training scheme where each subsampled frame in the input of the neural network is considered to represent $l = 3 \times 10\text{ms} = 30\text{ms}$ of context. The data is fed into the network as audio chunks of 4.2 seconds (140 frames) in minibatches of 32. We apply a decaying learning rate with beginning and final rates of 10^{-4} and 10^{-5} respectively. Stochastic gradient descent is used as the optimizer. The training is done for 6 epochs.

6. RESULTS

We report lyrics transcription results based on word error rate (WER). We begin by comparing performances obtained using *DAMP* and/or *DALI* in training the acoustic model. Then we test our proposed idea of discriminating silence and accompaniment instances by using separate tokens, and perform experiments testing different topologies of the MTDNN architecture. To boost the performance further, we train a final model on augmented data and provide a comparison of our results with previously published models.

6.1 Multi-Domain Training

According to Table 3, the model trained on *DAMP*^{train} performs relatively well on *DAMP*^{test}, however its performance drops dramatically on polyphonic recordings. On the other hand, much better recognition rates are observed on *DALI*^{test} when a polyphonic model is used, but then the polyphonic model performs poorly on a cappella recordings. Finally, using recordings from both the monophonic and polyphonic domains results in improved performance on both polyphonic and monophonic test sets, although the improvement is marginal on the monophonic *DAMP*^{test} set.

6.2 Music / Silence Modeling

Next, we test whether the explicit music/silence tagging improves transcription results. At this stage, we use a single stream architecture (M_8^{single} in Table 4). Tagging is applied only in constructing the GMM-HMM model and for generating alignments. The music/silence tags were removed during neural network training. Table 3 shows that alignment with music/silence tags did result in considerably improved recognition results for polyphonic recordings, but no improvement was evident for the monophonic case.

6.3 Neural Architecture Design

Here, we test various parameterizations of the multistreaming architecture. In this stage, we did not use the explicit music and silence tagging for training the models. As mentioned in Section 3, we diversify

Train Set	<i>DAMP</i> ^{test}	<i>DALI</i> ^{test}
<i>DAMP</i>	17.64	78.42
<i>DALI</i>	61.95	59.19
<i>DAMP</i> + <i>DALI</i>	17.14	53.86
+ music/sil tag	17.29	47.00

Table 3. Multi-domain training and music/silence tagging results

each stream of TDNNs in terms of the number of hidden layers and/or their dimensions. In addition to achieving improved performance, the goal of these modifications is to exploit the temporal context to its full extent. For this, we calculate the number of TDNN layers included w.r.t. the resulting $RF_{z_\tau^N}$.

In all MTDNN variants tested, we use 3 TDNN streams with $\tau \in \{3, 6, 9\}$. We begin with finding the optimal number of TDNN layers for the stream with the smallest τ . For rapid experimentation, we use single-streaming TDNN models (M_N^{single} in Table 4). According to Table 4, using 9 layers sets the optimal setup for $\tau = 3$, having $RF_{z_\tau^N} = 1620\text{ms}$. Note that further increasing the number of TDNN layers to 10 ($RF_{z_\tau^N} = 1800\text{ms}$) did not result in improved recognition, and the model complexity was much higher (Figure 4). Therefore, we chose as our baseline a single-stream model with 9 TDNN layers.

	Stream	Layers	Dimension	dev	<i>DAMP</i> ^{test}	<i>DALI</i> ^{test}
M_7^{single}	3	7	512	28.06	17.08	54.52
M_8^{single}	3	8	512	28.05	17.14	53.44
M_9^{single}	3	9	512	27.68	17.08	52.25
M_{10}^{single}	3	10	512	27.67	17.21	53.58
$M_{9,a}^{\text{multi}}$	3-6-9	(9,9,9)	(512,512,512)	26.69	16.75	51.38
$M_{9,b}^{\text{multi}}$	3-6-9	(9,4,3)	(512,512,512)	26.65	16.45	49.32
$M_{9,c}^{\text{multi}}$	3-6-9	(9,9,9)	(512,256,172)	27.13	16.08	52.54
$M_{9,d}^{\text{multi}}$	3-6-9	(9,4,3)	(512,256,172)	27.38	16.62	51.92

Table 4. Experiments on NN design

Next, we perform ablative tests on four variants of the multistream architecture (notated as $M_{9,\{a,b,c,d\}}^{\text{multi}}$). Model $M_{9,a}^{\text{multi}}$ have identical TDNN structures (except for τ), whereas the variants $M_{9,\{b,c\}}^{\text{multi}}$ have reduced N or hidden dimensions respectively w.r.t. τ at each stream. Both dimensions of model reduction are applied on $M_{9,d}^{\text{multi}}$. In models $M_{9,\{b,d\}}^{\text{multi}}$, we reduced the number of layers, N for the streams with larger τ to keep $RF_{z_\tau^N}$ similar across all streams. $M_{9,\{b,d\}}^{\text{multi}}$ have 4 and 3 layers at the streams with $\tau = 6$ and $\tau = 9$ having RF values of 1440 and 1620ms respectively. On the other hand, adding one more layer on the streams with $\tau = 6, 9$ would result in having $RF_{z_\tau^N} \geq 1800\text{ms}$ which is shown above to be suboptimal in the single-stream case (see results for M_{10}^{single}).

6.4 Model Selection

The proposed multistreaming setups except $M_{9,c}^{\text{multi}}$ outperformed their single-stream counterpart, M_9^{single} , particularly on *DALI*^{test}. The best results are achieved

with $M_{9,b}^{multi}$ which has unique N layers across all streams with the same hidden layer dimension.

To increase confidence in model selection, we investigate other operational aspects of the tested models. In Figure 4, we compare the number of trainable parameters which is a variable related to model complexity, and the real-time factor (RTF) that measures how fast a model operates during inference. We compute RTF’s based on the inference times across all the data used in evaluation. We repeat this 5 times and report the mean of all iterations per model. These iterations are performed on an Intel® Xeon® Gold 5218R CPU.

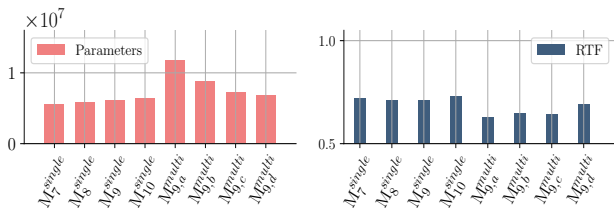


Figure 4. Num. trainable parameters (left) & RTFs (right)

According to Figure 4, our best performing model $M_{9,b}^{multi}$ has the second largest number of trainable parameters. Its model complexity is however much lower than that of $M_{9,a}^{multi}$, the architecture presented in [16]. In terms of run time, all multistreaming models performed faster than single-stream models, with $M_{9,b}^{multi}$ being among the fastest. This shows that our compact variant has a reduced inference time with an improved recognition rate as hypothesized in Section 1.

6.5 Comparison with the State of the Art

At this last step, we train a final model combining the music / silence aware alignment with the best performing MTDNN architecture, $M_{9,b}^{multi}$. To boost the performance further, we apply data augmentation via speed perturbation with the factors of 0.9 and 1.1, tripling the size of the training data. In Table 5, we compare our final model with other lyrics transcribers reported in the literature. We retrained the acoustic models in [2] ($M_{[2]}$), and [4] ($M_{[4]}$), using the corresponding publicly shared repositories. $M_{[9]}$ is based on the pretrained acoustic model shared at <https://github.com/chitralekha18/AutoLyrixAlign>. The same language model is used in constructing the decoding graphs for all the models in Table 5. Note that $M_{[2],[4]}$ are trained on *DAMP* (monophonic) and $M_{[9]}$ is trained on *DALI* (polyphonic) datasets. We used the best performing language model scaling factor when reporting the results in Table 5. We were not able to generate results on *DALI*^{test} using $M_{[9]}$ due to the model being highly memory intensive, as also reported in [8].

In addition to these, we provide a comparison with the state of the art. The best WER score reported on *DAMP*^{test} is based on $M_{[4]}$ and applies rescoring on the word lattices generated after the first-pass decoding using an RNNLM [38]. We did not apply RNNLM rescoring as we did not achieve consistent improvements across different test

sets according to our empirical observations. For a fair comparison, we also include the best results in [4] achieved via n-gram LMs (the scores in paranthesis in Table 5). On Jamendo, the best WER scores were reported in [8] where the inference was performed on source separated vocals. For Hansen and Mauch datasets, the best results are provided as reported in [9]⁸. In order to evade optimistic results, we have discarded the overlapping songs between Hansen, Mauch and *DALI*^{train} during training the final model.

	WER				
	<i>DAMP</i> ^{test}	<i>DALI</i> ^{test}	Jamendo	Hansen	Mauch
$M_{[2]}$	16.86	67.12	76.37	77.59	76.98
$M_{[9]}$	56.90	N/A	50.64	39.00	40.43
$M_{[4]}$	17.16	76.72	66.96	78.53	78.50
S.O.T.A	14.96 (17.01) [4]	N/A	51.76 [8]	47.01 [9]	44.02 [9]
MSTRE-Net	15.38	42.11	34.94	36.78	37.33

Table 5. Comparison with the state-of-the-art.

The results above show that MSTRE-Net outperforms all of the previously presented models on the polyphonic sets, with more than 15% , 7% and 6% absolute WER improvements achieved on the Jamendo, Hansen and Mauch datasets compared to the previous state of the art respectively. Notably, we achieved less than 50% WER on the large *DALI*^{test} set indicating more than half of the words across 240 songs were correctly predicted. Our model also has the best results on *DAMP*^{test} achieved via n-gram LM.

7. CONCLUSION

We have introduced MSTRE-Net, a novel compact variant of the multistreaming neural network architecture, which outperforms previously proposed automatic lyrics transcription models. Our model achieved these results with lower model complexity and inference time. In addition, we showed that recognition rates improved across all evaluation sets after leveraging both polyphonic and monophonic data in training the acoustic model. We proposed a novel data preprocessing method for generating alignments prior to neural network training which resulted in considerably better word recognition rates from polyphonic recordings compared to the baseline approach. Finally, we curated a new evaluation set that is more comprehensive and varied, while having a much larger size compared to the previous test data used in research. For reproducibility and open science, the identifiers and a tutorial on making use of this data will be shared with the research community.

Our final model outperformed the previously reported best ALT results by a large margin, setting the new state-of-the-art. Through these results, we have taken an important step in increasing the potential and the possibility for ALT being an applicable technology in both Music Information Retrieval research and the music technology industry.

⁸ Note that the reason for the WER difference between $M_{[9]}$ and the scores reported in [9] is due to the bigger language model we used, despite both models having the same acoustic model.

8. REFERENCES

- [1] P. A. Fine and J. Ginsborg, “Making myself understood: Perceived factors affecting the intelligibility of sung text,” *Frontiers in Psychology*, vol. 5, p. 809, 2014.
- [2] G. R. Dabike and J. Barker, “Automatic lyrics transcription from karaoke vocal tracks: Resources and a baseline system,” in *Interspeech*, 2019.
- [3] C. Gupta, R. Tong, H. Li, and Y. Wang, “Semi-supervised lyrics and solo-singing alignment,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [4] E. Demirel, S. Ahlbäck, and S. Dixon, “Automatic lyrics transcription using dilated convolutional neural networks with self-attention,” in *International Joint Conference on Neural Networks (IJCNN)*, 2020.
- [5] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, “DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [6] “Smule sing! 300x30x2 dataset,” accessed April, 2021, <https://ccrma.stanford.edu/damp/>.
- [7] E. Demirel, S. Ahlbäck, and S. Dixon, “Computational pronunciation analysis in sung utterances,” in *European Conference on Signal Processing (EUSIPCO)*, 2021.
- [8] —, “Low resource audio-to-lyrics alignment from polyphonic music recordings,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [9] C. Gupta, E. Yilmaz, and H. Li, “Automatic lyrics transcription in polyphonic music: Does background music help?” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [10] H. Bourlard and S. Dupont, “A new ASR approach based on independent processing and recombination of partial frequency bands,” in *Fourth International Conference on Spoken Language Processing (ICSLP)*, 1996.
- [11] J. Allen, “How do humans process and recognize speech?” *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 567–577, 1994.
- [12] H. Hermansky and P. Fousek, “Multi-resolution RASTA filtering for TANDEM-based ASR,” in *Interspeech*, 2005.
- [13] Z. Tüske, R. Schlüter, and H. Ney, “Acoustic modeling of speech waveform based on multi-resolution, neural network signal processing,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
- [14] S. H. Mallidi, T. Ogawa, K. Veselý, P. S. Nidadavolu, and H. Hermansky, “Autoencoder based multi-stream combination for noise robust speech recognition,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [15] K. J. Han, R. Prieto, and T. Ma, “State-of-the-art speech recognition using multi-stream self-attention with dilated 1D convolutions,” in *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2019.
- [16] K. J. Han, J. Pan, V. K. N. Tadala, T. Ma, and D. Povey, “Multistream CNN for robust acoustic modeling,” in *Interspeech*, 2020.
- [17] J. Pan, J. Shapiro, J. Wohlwend, K. J. Han, T. Lei, and T. Ma, “ASAPP-ASR: Multistream CNN and self-attentive SRU for SOTA speech recognition,” in *Interspeech*, 2020.
- [18] A. M. Kruspe, “Training phoneme models for singing with “songified” speech data.” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [19] D. Stoller, S. Durand, and S. Ewert, “End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [20] J. K. Hansen, “Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients,” in *Ninth Sound and Music Computing Conference (SMC)*, 2012.
- [21] M. Mauch, H. Fujihara, and M. Goto, “Integrating additional chord information into HMM-based lyrics-to-audio alignment,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 200–210, 2011.
- [22] M. Gales and S. Young, “The application of hidden Markov models in speech recognition,” *Foundations and Trends in Signal Processing*, 2008.
- [23] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, and P. Schwarz, “The Kaldi speech recognition toolkit,” in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.
- [24] M. Mohri, F. Pereira, and M. Riley, “Weighted finite-state transducers in speech recognition,” *Computer Speech & Language*, 2002.
- [25] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, “Sequence-discriminative training of deep neural networks,” in *Interspeech*, 2013.

- [26] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, “Purely sequence-trained neural networks for ASR based on lattice-free MMI.” in *Interspeech*, 2016.
- [27] L. Bahl, P. Brown, P. De Souza, and R. Mercer, “Maximum mutual information estimation of hidden Markov model parameters for speech recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1986.
- [28] V. Peddinti, D. Povey, and S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [29] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, “Semi-orthogonal low-rank matrix factorization for deep neural networks.” in *Interspeech*, 2018.
- [30] G. Meseguer-Brocal, R. Bittner, S. Durand, and B. Brost, “Data cleansing with contrastive learning for vocal note event annotations,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [31] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. d’Alché Buc, “Multilingual lyrics-to-audio alignment,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [32] J. R. Novak, D. Yang, N. Minematsu, and K. Hirose, “Phonetisaurus: A WFST-driven phoneticizer,” in *International Workshop on Finite State Methods and Natural Language Processing*, 2012.
- [33] C. Gupta, H. Li, and Y. Wang, “Automatic pronunciation evaluation of singing.” in *Interspeech*, 2018.
- [34] T. Alumäe and M. Kurimo, “Efficient estimation of maximum entropy language models with N-gram features: An SRILM extension,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.
- [35] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, “A compact model for speaker-adaptive training,” in *Fourth International Conference on Spoken Language Processing*, 1996.
- [36] G. Chen, H. Xu, M. Wu, D. Povey, and S. Khudanpur, “Pronunciation and silence probability modeling for ASR,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [37] G. Saon, H. Soltan, D. Nahamoo, and M. Picheny, “Speaker adaptation of neural network acoustic models using i-vectors,” in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013.
- [38] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey, and S. Khudanpur, “A pruned RNNLM lattice-rescoring algorithm for automatic speech recognition,” in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

TOWARDS AUTOMATIC INSTRUMENTATION BY LEARNING TO SEPARATE PARTS IN SYMBOLIC MULTITRACK MUSIC

Hao-Wen Dong¹ Chris Donahue² Taylor Berg-Kirkpatrick¹ Julian McAuley¹

¹ University of California San Diego ² Stanford University

¹{hwdong,tberg,jmcauley}@ucsd.edu ²cdonahue@cs.stanford.edu

ABSTRACT

Modern keyboards allow a musician to play multiple instruments at the same time by assigning zones—fixed pitch ranges of the keyboard—to different instruments. In this paper, we aim to further extend this idea and examine the feasibility of automatic instrumentation—dynamically assigning instruments to notes in solo music during performance. In addition to the online, real-time-capable setting for performative use cases, automatic instrumentation can also find applications in assistive composing tools in an offline setting. Due to the lack of paired data of original solo music and their full arrangements, we approach automatic instrumentation by learning to separate parts (e.g., voices, instruments and tracks) from their mixture in symbolic multitrack music, assuming that the mixture is to be played on a keyboard. We frame the task of part separation as a sequential multi-class classification problem and adopt machine learning to map sequences of notes into sequences of part labels. To examine the effectiveness of our proposed models, we conduct a comprehensive empirical evaluation over four diverse datasets of different genres and ensembles—Bach chorales, string quartets, game music and pop music. Our experiments show that the proposed models outperform various baselines. We also demonstrate the potential for our proposed models to produce alternative convincing instrumentations for an existing arrangement by separating its mixture into parts. All source code and audio samples can be found at <https://salu133445.github.io/arranger/>.

1. INTRODUCTION

Music is an art of time and sound. It often contains complex textures and possibly *parts* for multiple voices, instruments and tracks. While jointly following the global style and flow of the song, each part possesses its own characteristics and can develop different musical ideas independently. For example, in pop music, guitar and piano tend to play chords and might span across a large pitch

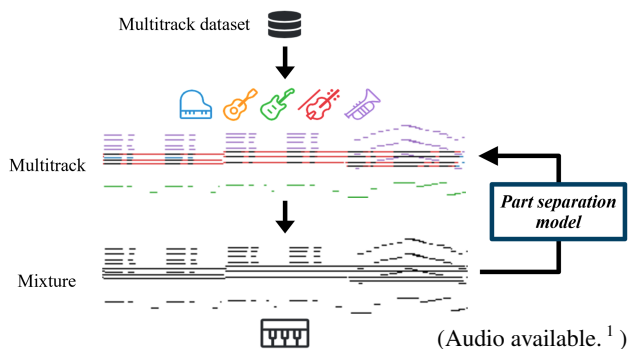


Figure 1. Proposed pipeline. By downmixing a symbolic multitrack into a single-track mixture, we acquire paired data of solo music and its instrumentation. We then use these paired data to train a *part separation* model that aims to infer the part label (e.g., one out of the five instruments in this example) for each single note in a mixture. Automatic instrumentation can subsequently be accomplished by treating input from a keyboard player as a downmixed mixture (bottom) and separating out the relevant parts (top). The music is visualized in the piano roll representation, where the x- and y-axes represent time and pitch, respectively. Colors indicate the instruments.

range, while bass is usually monophonic and stays in a lower range. While playing multiple instruments usually requires multiple performers, keyboardists potentially have the ability to control many instruments at once. Modern keyboards often offer the functionality of *zoning*, which allows a player to divide the pitch range into zones and assign each zone to a certain instrument. However, zoning is not ideal given its low flexibility that requires careful configuration and sometimes rearrangement of the music, and incapability for handling certain genres of music that have close and possibly overlapping harmony.

In this paper, we aim for the more ambitious goal of automatic instrumentation—a process that we define as dynamically assigning instruments to notes in solo music. A real-time, online automatic instrumentation model could allow a musician to have their keyboard performance instantaneously and seamlessly performed by a different ensemble. In addition to performative use cases, an offline automatic instrumentation model can also be useful to assist composers in suggesting proper instrumentation or providing a starting point for arranging a solo piece, especially for composers who have less experience arranging



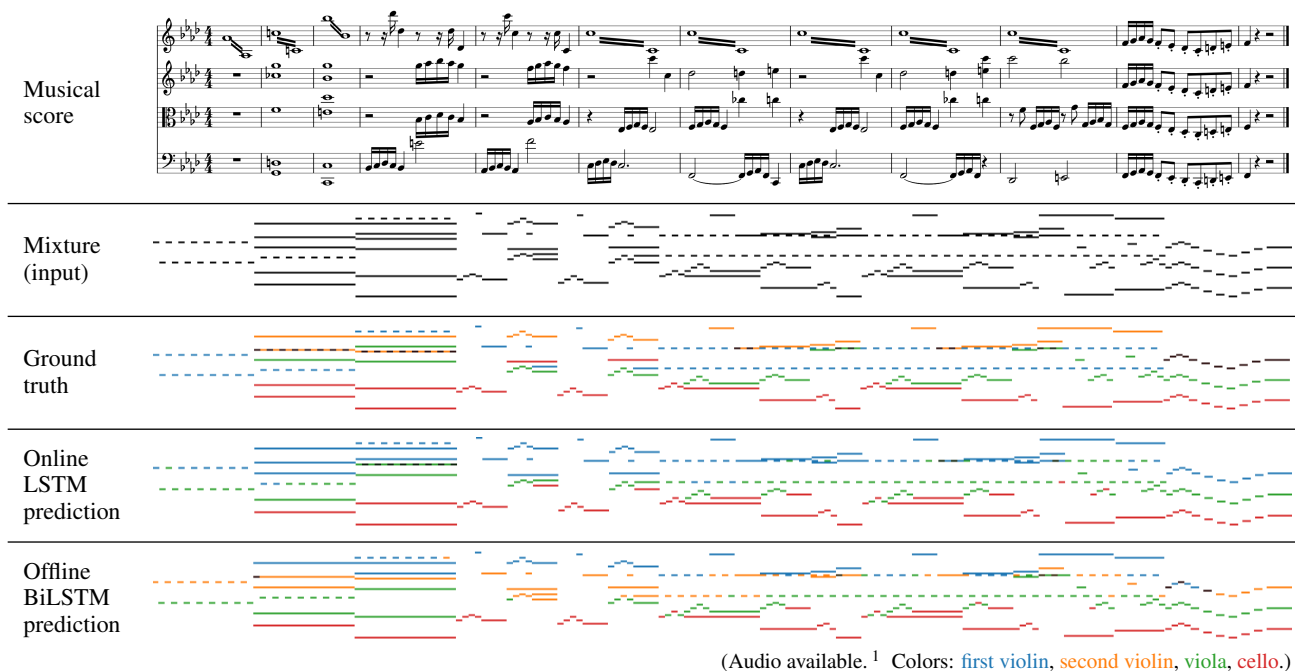


Figure 2. Hard excerpt in the string quartets dataset—Beethoven’s *String Quartet No. 11 in F minor, Op. 95*, movement 1, measures 72–83. The tremolos of the first violin (measures 1–3 and 6–10), the double stops for the second violin, viola and cello (measures 2–3) and the overlapping pitch ranges (measures 2–5) together compose a complex texture. Both models fail to handle the violins and viola properly, especially the second violin.

for a particular ensemble.

Automatic instrumentation is challenging as it requires domain knowledge of each target instrument, i.e., which pitches, rhythms, chords, and sequences thereof are playable, and it is hard to specify such knowledge by some fixed set of rules. In view of recent advances in machine learning, we propose to adopt a data-driven approach to this task. However, it can be laborious to acquire paired data of original solo music and their full arrangements. Given the abundance of multitrack music data, we approach automatic instrumentation by learning to separate parts from their mixture in multitrack music, a task we call *part separation* (see Figure 1). Assuming that the mixture is to be played on a keyboard and the multitrack is the target arrangement that we want to generate by an automatic instrumentation model, we thereby have paired data for solo music and full arrangements.

We frame the new task of part separation as a sequential multi-class classification problem that aims to map sequences of notes into sequences of part labels. We adopt long short-term memory (LSTM) [1] and Transformer [2] models for the task. We conduct an extensive empirical evaluation showing the superiority of our proposed models to baselines for the related task of voice separation as well as strategies found in commodity keyboards. To showcase the potential of our proposed models, we also demonstrate their ability to produce alternative convincing instrumentations for existing arrangements. Audio for all examples and more samples are available on the demo website.¹ All source code can be found in the project repository.²

¹ <https://salu133445.github.io/arranger/>

² <https://github.com/salu133445/arranger>

2. PRIOR WORK

Voice separation is a related task to part separation which involves separating blended scores into individual monophonic voices. While useful, voice separation is agnostic to constraints imposed by specific instruments—a composer using a voice separation algorithm would have to manually align voices to appropriate instruments. Some prior work investigates voice separation in small, carefully-annotated pop music datasets [3, 4]. Some prior work on voice separation allows synchronous or overlapping notes in a voice [5–8]. However, their results are only reported on small test sets in certain genres. Others have adopted multilayer perceptrons [3, 9] and convolutional neural networks [10] with hand-crafted input features for voice separation. Another relevant work on hand detection in piano music used LSTMs to separate notes played by right and left hands in piano MIDI data [11]. To the best of our knowledge, no past work has examined the task of part separation in a general setting for multiple music genres.

In addition to voice separation, prior work has explored automatic music arrangement. The primary focus of prior work for automatic music arrangement has been on reduction—mapping musical scores for large ensembles to parts playable by a single specific instrument such as the piano [12–17], guitar [18–20] or bass [21]. This past work focuses on identifying the least important notes to delete so that the resultant score is playable on a single instrument, whereas our work seeks to preserve the original score in its entirety and satisfy playability for multiple instruments simultaneously. As an exception, Crestel and Esling [22] explore strategies for arranging orchestral music from pi-

ano, though their approach does not guarantee that all notes in the input piano map to parts in the output.

Music generation is another body of work that has used neural network sequential models for processing symbolic music [23]. Simon and Oore [24] proposed a convenient approach for music generation which involved training recurrent neural network language models on a language-like “event-based” representation of music. Subsequently, recent work has explored event-based representations using Transformers [2, 25–30]. In this work, we explore a more compact input representation of music that passes all of the information about a note into the model at once, rather than spreading it out across several events. We also note that Payne [26] generate music which contains parts for several instruments, but their model cannot be directly used to perform part separation of existing musical material.

3. PROBLEM FORMULATION

Mathematically, we consider a piece of music x as a sequence of notes (x_1, \dots, x_N) , where N is the number of notes. Each note is represented by a tuple of time t_i and pitch p_i , i.e., $x_i = (t_i, p_i)$. Alternatively, we could also include duration d_i as an input and have $x_i = (t_i, p_i, d_i)$. Each note is associated with a label $y_i \in \{1, \dots, K\}$ that represents the part it is in, where K is the number of parts. The goal of part separation is to learn the mapping between notes and part labels. This formulation is rather flexible and has no assumptions on whether a part is monophonic or not—it can be a voice, an instrument, a track, etc.

In terms of the context given for predicting the label of each note, we can categorize part separation models into three classes: An *independent model* predicts the label for each note independently, without any context. An *online model* predicts the label of the current note x_i given only past information, i.e., notes (x_1, \dots, x_{i-1}) , as context. An *offline model* predicts the label of the current note x_i given past and future information, i.e., the full sequence of (x_1, \dots, x_N) , as context. While independent and online models are preferable for use cases that require real-time outputs, e.g., live performance. Moreover, the inability to look into the future makes the real-time setting more challenging than the offline setting. On the other hand, offline models can find applications in assistive composing tools.

4. MODELS

We consider the following input features for our models—(1) *time*: onset time, in time step,³ (2) *pitch*: pitch as a MIDI note number, (3) *duration*: note length, in time step, and (4) *frequency*: fundamental frequency of the pitch, in Hz, computed by the formula $f = 440 \cdot 2^{(p-69)/12}$. In addition, we also consider features that encode the metric time grid of music similar to the BAR and POSITION events proposed in [28]—(5) *beat*: onset time, in beat, and (6) *position*: position within a beat, in time step.

³ Assuming that the music is in metrical timing, a time step is a factor of some musically-meaningful unit (e.g., a quarter note) and can be adjusted to match the desired temporal resolution.

Moreover, to help the models better disambiguate parts, we also include two simple hints—(7) *entry hints*: onset position for each instrument, encoded as a unit step function centered at its onset time and all zero if the instrument is not used, and (8) *pitch hints*: average pitch of each track. These hints allow the musician to use interactively to make the instrumentation process more controllable. For example, entry hints can be used to control the instruments available as they serve as switches for the instruments.

For the machine learning models, we consider the LSTM [1] and its bidirectional version (BiLSTM) [35]. We use a three-layer stacked LSTM with 128 hidden units in each layer (64 hidden units per layer for BiLSTM). We also consider two variants of Transformer [36]—one based on the encoder (Transformer-Enc) and one based on the decoder (Transformer-Dec). They share the same architecture that is composed of three Transformer blocks, each of which has 128 hidden units and 8 heads in self-attention computation and 256 hidden units in the internal feedforward network. However, they have different attention masks: Transformer-Enc uses only the padding mask, while Transformer-Dec uses both the padding mask and the lookahead mask, which blocks its access to future information and makes it an online model. In this paper, the LSTM and Transformer-Dec models are made online models, and the BiLSTM and Transformer-Enc models are made offline models that take durations as inputs.

5. BASELINE MODELS

In order to gain an insight into how the proposed models perform, we include two heuristic algorithms and a voice separation model from the literature in our empirical study.

5.1 Zone-based algorithm

This algorithm simulates a common feature in modern keyboards where a player can preassign a pitch range (i.e., the ‘zone’) for each instrument and notes will automatically be assigned to the corresponding instrument as the player performs. This algorithm finds the optimal zones for the whole training data and uses these optimal zones at test time. For the oracle case, the optimal zones for each sample are computed and used at test time. We note that the oracle case might not be easily achievable as it can be hard for a musician to set the zones optimally beforehand, especially for improvisation.

5.2 Closest-pitch algorithm

The closest-pitch algorithm keeps track of the last active pitches p'_i for each track i . For each incoming pitch p , it finds the pitch among the last active pitches that has the closest pitch to p and assigns the upcoming note with the same label as the chosen pitch. This is a casual model and it also relies on the onset hints. We can formulate this algorithm as follows. For $i = 1, \dots, N$, we have

$$\hat{y}_i = \begin{cases} y_i, & \text{if } x_i \text{ is an onset} \\ \arg \min_{j \in \{1, \dots, K\}} (p_i - p'_j)^2 + Ma_i, & \text{otherwise} \end{cases},$$

Dataset	Hours	Files	Notes	Parts	Ensemble	Most common label
Bach chorales [31]	3.23	409	96.6K	4	soprano, alto, tenor, bass	bass (27.05%)
String quartets [32]	6.31	57	226K	4	first violin, second violin, viola, cello	first violin (38.72%)
Game music [33]	45.05	4.61K	2.46M	3	pulse wave I, pulse wave II, triangle wave	pulse wave II (39.35%)
Pop music [34]	1.02K	16.2K	63.6M	5	piano, guitar, bass, strings, brass	guitar (42.50%)

Table 1. Statistics of the four datasets considered in this paper.

where p'_i is the last active pitch of track i before time t and a_i indicates whether track i is active, i.e., a concurrent note has not yet been released. We set M to a large positive number when we assume each part is monophonic, which we will refer to as the ‘mono’ version of this algorithm, otherwise set $M = 0$.

5.3 Multilayer perceptron (MLP)

We adapt the voice separation model proposed in [9] to the task of part separation. This model uses multilayer perceptron (MLP) to predict the label for the current note based on hand-crafted features that encodes its nearby context. We use entry hints rather than predicting them by the proposed voice entry estimation heuristics. We remove the ‘interval’ feature as there is no upper bound for the number of concurrent notes and change the proximity function to L1 distance. The oracle case of this model replaces error-prone prior predictions with ground truth history labels. In our implementation, we use three fully-connected layers with 128 hidden units each.

6. DATA

In order to examine the effectiveness of the proposed models, we consider four datasets—(1) *Bach chorales* in Music21 [31], (2) *string quartets* in MusicNet [32], (3) *game music* in Nintendo Entertainment System (NES) Music Database [33] and (4) *pop music* in Lakh MIDI Dataset [34], which are diverse in their genres, sizes and ensembles (see Table 1 for a comparison).

As these datasets are noisy in different ways, we need to further clean the data. For the game music dataset, we discard the percussive noise track in the original dataset as they do not follow the standard 128-pitch system used in other tracks. For the pop music dataset, we use a cleaned subset derived in [37], which contains only pop songs. We mapped the instruments to the five most common instrument families—piano, guitar, bass, strings and brass. We follow the General MIDI 1 specification on the mapping from an instrument to its instrument family. Instruments that fall outside of these five families are discarded. We note that the lead melody track might occasionally be discarded during the mapping process due to the high variance on instruments used for the melody track.

Moreover, we discard songs with only one active track as the task becomes trivial in this case. We note that all Bach chorales, string quartets and most pop songs are in metrical timing, where a time step corresponds to some fraction of a quarter note. Thus, we downsample them into 24 time steps per quarter note, which can cover 32nd notes

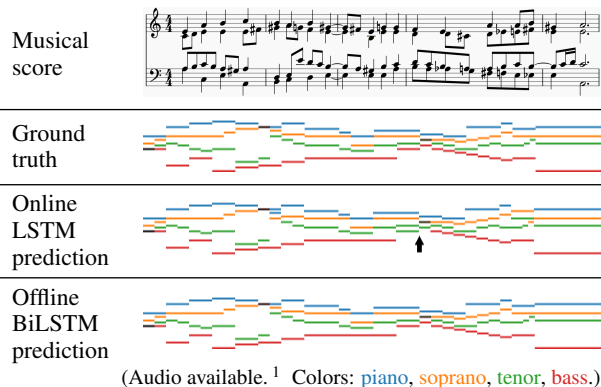


Figure 3. Example of the Bach chorales dataset—*Wer nur den lieben Gott läßt walten*, BWV 434, measures 1–5. The LSTM model makes two errors for the bass, as indicated by the arrow. The BiLSTM model gives a perfect prediction.

and triplets. As songs in the game music dataset are in absolute time, we downsample them to a temporal resolution equivalent to 24 time steps per quarter note in a tempo of 125 quarter notes per minute (qpm).

Finally, we split each dataset into train–test–validation sets with a ratio of 8 : 1 : 1 except the game music dataset, where we use the original splits provided with the NES Music Database. We use MusPy [38] and music21 [31] for processing MIDI and MusicXML files.

7. EXPERIMENTS

7.1 Implementation details

We use a batch size of 16, a sequence length of 500 for training and a maximum sequence length of 2000 for validation and testing. We clip the time by 4096 time steps (i.e., roughly 170 quarter notes), the beat by 4096 beats, and durations by 192 time steps (i.e., 8 quarter notes). We randomly transpose the music by -5 to +6 semitones during training for data augmentation. We use the cross entropy loss with the Adam optimizer with $\alpha = 0.001$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ [39]. We apply dropout [40] to prevent overfitting and layer normalization [41] to speed up the training. All models are implemented in TensorFlow [42] and experiments are run on NVIDIA GeForce RTX 2070s.

7.2 Qualitative results and error analysis

We present in Figures 2 to 5 several examples in the four datasets. Some representative cases include overlapping pitch ranges or chords for two polyphonic instruments (see Figures 2 and 5), overlapping melodies and chords

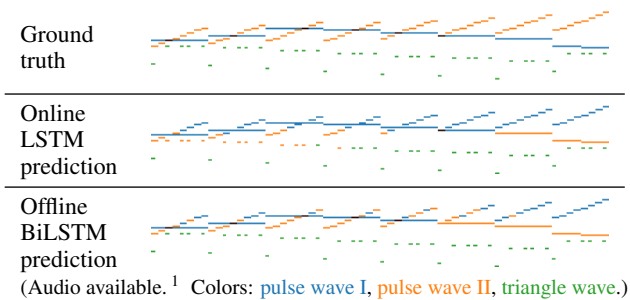


Figure 4. Hard excerpt in the game music dataset—*Theme of Universe* from *Miracle Ropit’s Adventure in 2100*. Both models perform poorly when there is a sequence of short notes crossing a single long note.

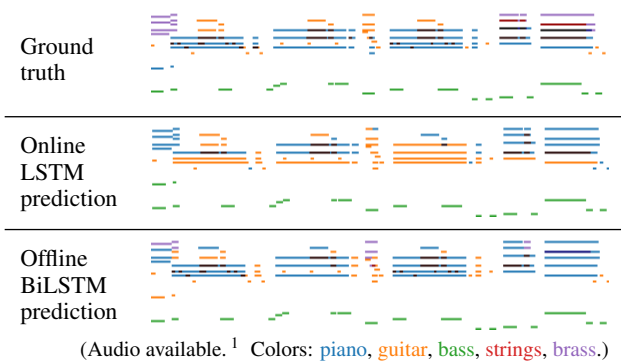


Figure 5. Hard excerpt in the pop music dataset—*Blame It On the Boogie* by The Jacksons. The BiLSTM model correctly identify and separate the overlapping guitar melody and piano chords, while the LSTM model fails in this case.

(see Figure 5) and a sequence of short notes crossing a single long note (see Figure 4).

7.3 Quantitative results

We conduct an extensive empirical evaluation over different dataset and models in different settings. We present the results in Table 2.⁴ First, we notice the improved performance for the oracle cases on the MLP baseline. The large gap of performance is possibly because it predicts each note independently and the errors can propagate over time. This emphasizes the need to incorporate sequential models for this task. Moreover, the BiLSTM model outperforms its LSTM counterpart for most cases. This is reasonable as the BiLSTM model has access to the future information, which could, for example, help identify the direction of an arpeggio. Further, the LSTM and BiLSTM models outperform their Transformer counterparts—Transformer-Dec and Transformer-Enc, respectively—across all settings. However, the Transformer models benefits from faster inference speed at test time as compared to the LSTM models. Finally, we notice that the proposed models perform relatively poorly on the string quartets and game music datasets, possibly because the two violins in the string quartets dataset and the two

⁴ Due to high computation cost, we report the oracle cases for the zone-based algorithm and MLP model on a subset of 100 test samples, and omit the oracle case of the zone-based algorithm for the pop music dataset.

Model	Bach	String	Game	Pop
Online models				
Zone-based	73.14	58.85	43.67	57.07
MLP [9]	81.63	29.85	43.08*	33.50*
LSTM	93.02	67.43	50.22	74.14
Transformer-Dec	91.51	57.03	45.82	62.14
Offline models				
Zone-based (oracle)	78.33	66.89	79.54*	†
MLP [9] (oracle)	97.59	58.16	65.30	44.62
Offline models (+entry hints)				
BiLSTM	97.13	74.38	52.93	77.23
Transformer-Enc	96.81	58.86	49.14	66.57
Online models (+entry hints)				
Closest-pitch	68.87	50.69	57.14	47.45
Closest-pitch (mono)	89.76	42.82	49.91	32.28
LSTM	92.70	62.64	62.11	74.19
Transformer-Dec	91.17	62.12	56.73	67.19
Offline models (+entry hints)				
BiLSTM	97.39	71.51	64.79	75.59
Transformer-Enc	93.81	56.72	54.67	67.23

*Reported on a subset of 100 test samples due to high computation cost.

†Omitted due to high computation cost.

Table 2. Comparison of our proposed models and baseline algorithms. Performance is measured in accuracy (%).

Emb	Dur	EH	PH	Bach	String	Game	Pop
				92.10	37.29	43.89	58.78
✓				93.02	67.43	50.22	74.14
✓	✓			96.17	66.96	51.38	78.17
✓		✓		92.70	62.64	62.11	74.19
✓	✓	✓		95.95	68.17	63.35	74.74
✓			✓	92.87	70.20	67.45	75.89

Table 3. Effects of input features to the online LSTM model. Performance is measured in accuracy (%). Abbreviations: ‘Emb’—pitch, beat and position embedding, ‘Dur’—duration, ‘EH’—entry hints, ‘PH’—pitch hints.

pulse waves in the game music dataset are sometimes used interchangeably. We examine the use of pitch hints to help the models in the following section.

7.4 Effect of input features

In order to compare the effectiveness of different input features, we also report in Table 3 the performance for the LSTM model with different input features. First of all, pitch, note and beat embedding leads to improvements on all datasets, especially significant on the string quartet (30% gain) and pop music (15% gain) datasets. Second, entry hints improve the performance by 10% for the game music dataset, which is possibly because it helps disambiguate the two interchangeable pulse wave tracks. Interestingly, they have negative impacts on the Bach chorales and pop music datasets. Third, duration inputs are always helpful and help achieve the highest accuracy on the Bach chorales and pop music datasets. For example, durations would be critical in distinguishing the overlapping guitar melody and piano chords in the example shown in Figure 5. Last, pitch hints improve the performance for all datasets but Bach chorales, possibly because the vocal ranges for

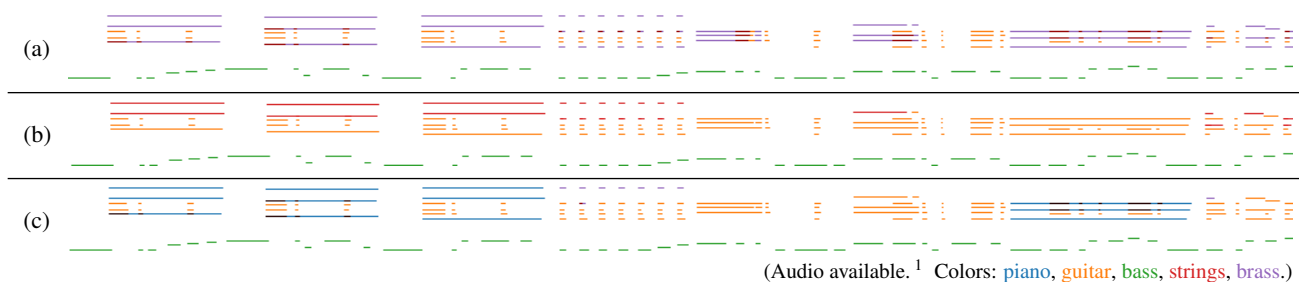


Figure 6. *Quando Quando Quando* by Tony Renis—(a) original instrumentation and the versions produced by (b) the online LSTM model without entry hints and (b) the offline BiLSTM model with entry hints. The LSTM model assigns the chords to the guitar, the most common instrument in the pop music dataset except the high pitches, which are assigned to the strings. The BiLSTM model is able to separate the long chords from the short ones and assigns the former to the piano.

Strategy	Bach	String	Game	Pop
Time encoding				
Raw time	91.97	37.26	44.10	37.92
Raw beat and position	93.13	66.72	48.60	68.42
Time embedding	92.21	68.31	49.32	70.64
Beat and position emb.	93.02	67.43	50.22	74.14
Data augmentation				
No augmentation	93.03	69.36	49.03	70.73
Light augmentation	92.85	68.66	46.38	71.10
Strong augmentation	93.02	67.43	50.22	74.14

Table 4. Comparisons of time encoding and data augmentation strategies for the online LSTM model. Performance is measured in accuracy (%).

SATB are strict in chorales. Pitch hints help achieve the highest accuracies for the string quartets and game music datasets as they help disambiguate interchangeable tracks.

7.5 Effects of time encoding

In this experiment, we examine the effects of time encoding. In particular, we consider four variants—(1) raw time as a number, (2) raw beat and position as two numbers, (3) time embedding and (4) beat and position embedding (see Section 4 for the definition of beat and position). We report in Table 4 the results and we can see that using raw time gives the worst performance. Interestingly, the other three encoding strategies achieve comparable performance.

7.6 Effects of data augmentation

In this experiment, we compare the following three strategies of data augmentation—(1) no augmentation, (2) *light augmentation*, where each song is randomly transposed by -1 to +1 semitone during training and (3) *strong augmentation*, where each song is randomly transposed by -5 to +6 semitones during training. We report in Table 4 the results. We can see that data augmentation is generally harmful for the Bach chorales and string quartets datasets, possibly because classical music has strict rules on the pitch ranges of voices and instruments. However, for game and pop music datasets, where rules on keys and pitch ranges in classical music are loosened, the models yield better performance with proper data augmentation.

8. DISCUSSION

In Figure 6, we depict the original instrumentation of the song *Quando Quando Quando* alongside the instrumentations generated by our best performing models for both the online and offline settings. While neither model produces an instrumentation identical to that of the original, both produce instrumentations that “cluster” notes similarly to the original and are reasonable rearrangements of the song. This indicates a fundamental ambiguity of the task, though we note that such ambiguity is less present in some genres than others—our models are able to achieve high accuracy on the Bach chorales dataset despite its small size. However, for larger and more diverse datasets (e.g., the pop music dataset), accuracy might not be the best metric for measuring the performance of the models, and we plan to include human evaluations in future work.

One limitation of this work lies in the generalizability to real keyboard music since the downmixed music might not be playable on a keyboard, e.g., having more than ten concurrent notes or impossible fingering. Moreover, we did not use the MIDI velocity information in our models, and it could provide an additional signal for separation.

Finally, in addition to its immediate musical applications, we believe that our proposed part separation task may be useful for large-scale pre-training of symbolic music models. Pre-training music generation models on large, heterogeneous music corpora has already been observed to improve performance [27, 43]. Given that our proposed task represents an additional source of musical knowledge supervision, we speculate that additionally pre-training on this task could improve performance for many downstream tasks, e.g., genre classification and melody extraction.

9. CONCLUSION

In this paper, we have proposed a new task of part separation in multitrack music and examined its feasibility under both the online and offline settings. Through a comprehensive empirical evaluation over four diverse datasets, we showed the effectiveness of our proposed models against various baselines. We also presented promising results for applying part separation models to automatic instrumentation. Moreover, we discussed the fundamental ambiguity and limitations of the task and future research directions.

10. REFERENCES

- [1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [2] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, “Compound word Transformer: Learning to compose full-song music over dynamic directed hypergraphs,” *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
- [3] P. Gray and R. Bunesco, “A neural greedy model for voice separation in symbolic music,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [4] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé, “Comparing voice and stream segmentation algorithms,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.
- [5] J. Kilian and H. H. Hoos, “Voice separation — a local optimisation approach,” in *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*, 2002.
- [6] E. Cambouropoulos, “‘Voice’ separation: theoretical, perceptual and computational perspectives,” in *Proceedings of the 9th International Conference on Music Perception & Cognition (ICMPC)*, 2006.
- [7] I. Karydis, A. Nanopoulos, A. Papadopoulos, and E. Cambouropoulos, “VISA: The voice integration/segregation algorithm,” in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.
- [8] E. Cambouropoulos, “Voice and stream: Perceptual and computational modeling of voice separation,” *Music Perception*, vol. 26, no. 1, 2008.
- [9] R. de Valk and T. Weyde, “Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [10] P. Gray and R. Bunesco, “From note-level to chord-level neural network models for voice separation in symbolic music,” *arXiv preprint arXiv:2011.03028*, 2020.
- [11] A. Hadjakos, S. Waloschek, and A. Leemhuis, “Detecting hands from piano MIDI data,” in *Mensch und Computer 2019 - Workshopband*, 2019.
- [12] S.-C. Chiu, M.-K. Shan, and J.-L. Huang, “Automatic system for the arrangement of piano reductions,” in *Proceedings of the 2009 11th IEEE International Symposium on Multimedia (ISM)*, 2009.
- [13] S. Onuma and M. Hamanaka, “Piano arrangement system based on composers’ arrangement processes,” in *Proceedings of the 2010 International Computer Music Conference (ICMC)*, 2010.
- [14] J.-L. Huang, S.-C. Chiu, and M.-K. Shan, “Towards an automatic music arrangement framework using score reduction,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 1, pp. 1–23, 2012.
- [15] E. Nakamura and S. Sagayama, “Automatic piano reduction from ensemble scores based on merged-output hidden Markov model,” in *Proceedings of the 2005 International Computer Music Conference (ICMC)*, 2015.
- [16] H. Takamori, H. Sato, T. Nakatsuka, and S. Morishima, “Automatic arranging musical score for piano using important musical elements,” in *Proceedings of the 14th Sound and Music Computing Conference, Aalto, Finland*, 2017, pp. 35–41.
- [17] E. Nakamura and K. Yoshii, “Statistical piano reduction controlling performance difficulty,” *APSIPA Transactions on Signal and Information Processing*, vol. 7, 2018.
- [18] D. R. Tuohy and W. D. Potter, “A genetic algorithm for the automatic generation of playable guitar tablature,” in *Proceedings of the 2005 International Computer Music Conference (ICMC)*, 2005.
- [19] G. Hori, Y. Yoshinaga, S. Fukayama, H. Kameoka, and S. Sagayama, “Automatic arrangement for guitars using hidden Markov model,” in *Proceedings of the 9th Sound and Music Computing Conference (SMC)*, 2012, pp. 450–456.
- [20] G. Hori, H. Kameoka, and S. Sagayama, “Input-output HMM applied to automatic arrangement for guitars,” *Information and Media Technologies*, vol. 8, no. 2, pp. 477–484, 2013.
- [21] Y. Abe, Y. Murakami, and M. Miura, “Automatic arrangement for the bass guitar in popular music using principle component analysis,” *Acoustical Science and Technology*, vol. 33, no. 4, pp. 229–238, 2012.
- [22] L. Crestel and P. Esling, “Live orchestral piano, a system for real-time orchestral music generation,” *arXiv preprint arXiv:1609.01203*, 2016.
- [23] J.-P. Briot, G. Hadjeres, and F. Pachet, “Deep learning techniques for music generation: A survey,” *arXiv preprint arXiv:1709.01620*, 2017.
- [24] I. Simon and S. Oore, “Performance RNN: Generating music with expressive timing and dynamics,” *Magenta Blog*, 2017. [Online]. Available: <https://magenta.tensorflow.org/performance-rnn>

- [25] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music Transformer: Generating music with long-term structure,” in *Proceedings of the 7th International Conference for Learning Representations (ICLR)*, 2019.
- [26] C. Payne, “MuseNet,” OpenAI, 2019. [Online]. Available: <https://openai.com/blog/musenet/>
- [27] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [28] Y.-S. Huang and Y.-H. Yang, “Pop music Transformer: Generating music with rhythm and harmony,” in *Proceedings of the 28th ACM International Conference on Multimedia (ACMMM)*, 2020.
- [29] J. Ens and P. Pasquier, “MMM: Exploring conditional multi-track music generation with the Transformer,” *arXiv preprint arXiv:2008.06048*, 2020.
- [30] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola, “Symbolic music generation with TransformerGANs,” in *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, 2021.
- [31] M. S. Cuthbert and C. Ariza, “Music21: A toolkit for computer-aided musicology and symbolic music data,” in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010.
- [32] J. Thickstun, Z. Harchaoui, and S. M. Kakade, “Learning features of music from scratch,” in *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.
- [33] C. Donahue, H. H. Mao, and J. McAuley, “The NES music database: A multi-instrumental dataset with expressive performance attributes,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [34] C. Raffel, “Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching,” Ph.D. dissertation, Columbia University, 2016.
- [35] M. Schuster and K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.
- [37] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
- [38] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, “MusPy: A toolkit for symbolic music generation,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [39] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, 2015.
- [40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [41] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” in *NeurIPS 2016 Deep Learning Symposium*, 2016.
- [42] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Symp. on Operating Systems Design and Implementation (OSDI)*, 2016.
- [43] H.-T. Hung, C.-Y. Wang, Y.-H. Yang, and H.-M. Wang, “Improving automatic jazz melody generation by transfer learning techniques,” in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019.

AN EMPIRICAL EVALUATION OF END-TO-END POLYPHONIC OPTICAL MUSIC RECOGNITION

Sachinda Edirisooriya Hao-Wen Dong Julian McAuley Taylor Berg-Kirkpatrick
University of California San Diego

{sediriso,hwdong,jmcauley,tberg}@ucsd.edu

ABSTRACT

Previous work has shown that neural architectures are able to perform optical music recognition (OMR) on monophonic and homophonic music with high accuracy. However, piano and orchestral scores frequently exhibit polyphonic passages, which add a second dimension to the task. Monophonic and homophonic music can be described as homorhythmic, or having a single musical rhythm. Polyphonic music, on the other hand, can be seen as having multiple rhythmic sequences, or voices, concurrently. We first introduce a workflow for creating large-scale polyphonic datasets suitable for end-to-end recognition from sheet music publicly available on the MuseScore forum. We then propose two novel formulations for end-to-end polyphonic OMR—one treating the problem as a type of multi-task binary classification, and the other treating it as multi-sequence detection. Building upon the encoder-decoder architecture and an image encoder proposed in past work on end-to-end OMR, we propose two novel decoder models—FlagDecoder and RNNDecoder—that correspond to the two formulations. Finally, we compare the empirical performance of these end-to-end approaches to polyphonic OMR and observe a new state-of-the-art performance with our multi-sequence detection decoder, RNNDecoder.

1. INTRODUCTION

As society continues to become more and more dependent on digitization as a means of storing information such as photos, addresses, and music, now is a perfect time to refine the technology required to digitize sheet music. Organizing scanned music can be a tedious task to do manually. For example, each image must be labeled with several attributes, the most basic being the title, composer, arranger, and page number. Assuming that all of these are input correctly, a user could then navigate through a large-scale collection somewhat easily. However what if a user wanted to filter the scores by attributes such as instrument, key signa-

The figure displays four musical staves. The top staff is a single melodic line in treble clef, marked with a mezzo-forte (mf) dynamic. The second staff shows a more complex texture with multiple voices in treble clef. The third staff is a dense polyphonic texture with many voices in treble clef. The fourth staff is another complex polyphonic texture in treble clef.

Figure 1. Examples of the MuseScore Polyphonic Dataset (MSPD) and its hard subset (MSPD-Hard)—(top) an easy excerpt in MSPD and (bottom) three excerpts that can be found in both MSPD and MSPD-Hard.

ture, time signature, or tempo? What if a user had a score, but wanted to transpose it to a different key? Manually doing all of the annotation required for these demands when uploading sheet music scans would be impractical, and this is where optical music recognition (OMR) can shine.

Over the past few years, data driven approaches to optical music recognition have become attractive ways to solve the problem. The improvement in the accuracy of systems built using these tools is very exciting, however they are far from perfect in challenging circumstances. One visual challenge relatively unique to optical music recognition is detecting multi-voice music, also known as polyphonic music. Previous work has mentioned that their approaches to OMR cannot sufficiently solve this problem to the same extent as they have solved monophonic OMR [1, 2].

In view of the lack of a large-scale polyphonic dataset for end-to-end polyphonic OMR, we introduce a workflow for acquiring annotated samples from sheet music publicly available on the MuseScore forum [3]. As an initial attempt to the challenges of polyphonic OMR, we only consider single-staff scores. Our objective with this dataset is to empirically evaluate the performance of three different architectures on the task of end-to-end polyphonic OMR, where the input is a staff line image and the output is a symbolic sequence that can be encoded into common music notation formats such as MusicXML. Voice information is not included in this process.

We propose two novel neural architectures for end-to-end OMR, namely FlagDecoder and RNNDecoder, which



are both encoder-decoder models based on the architecture proposed by Calvo-Zaragoza et al. for monophonic scores [4], hereafter referred to as the baseline. One feature common to all three models is their mechanism for encoding each image: a CNN followed by a bidirectional LSTM. The architectures differ, however, in their decoding mechanism. The baseline architecture uses a fully connected layer as a symbol classifier on the encoded image. Our proposed FlagDecoder treats polyphonic OMR as a multi-task binary prediction problem, simultaneously detecting whether each pitch and staff symbol is present or not, along with the rhythm if the symbol is a note. Our other proposed architecture, RNNDecoder, uses a vertical RNN decoder over the symbols appearing at each vertical image slice, giving the model the capacity to output a predetermined number of symbol predictions at a single horizontal position of an image.

In this paper, we first introduce the current state of polyphonic OMR research. Then we introduce our procedure for building a large-scale dataset of exclusively polyphonic music data. Finally we perform an empirical evaluation of our proposed architectures introduced above, and find that they both outperform past work in terms of symbol error rate, with the RNNDecoder achieving a new state-of-the-art performance on end-to-end polyphonic OMR. All source code is available at <https://github.com/sachindae/polyphonic-omr>.

2. BACKGROUND

2.1 Optical music recognition (OMR)

Previous work on polyphonic OMR has been limited. One of the main approaches to it has been a two-step process of first segmenting each musical symbol [5–9] and then identifying the relationships between them through a post-processing step [10]. The challenge with building a system using this method is that errors add up from both sub-systems. Particularly the latter task, more formally known as forming a musical notation graph, can be quite challenging, with the state-of-the-art being far from perfect [10]. Another approach to OMR has been to treat it as an end-to-end task as proposed by [4, 11], where the complete symbolic sequence corresponding to an image is output by the system. Among the end-to-end approaches, there have been a variety of training objectives used for the task such as Connectionist Temporal Classification (CTC) loss, cross-entropy loss, and Smooth-L1 loss [4, 11, 12]. While applied to monophonic and homophonic music [13] successfully, there have not been any conclusions on the framework’s capability of being extended to polyphonic notation.

2.2 Datasets for OMR

Several datasets have been proposed for musical symbol classification [14–19]. Others have been proposed for training end-to-end OMR systems on single-stave handwritten music scores [2] and typeset images for mono-

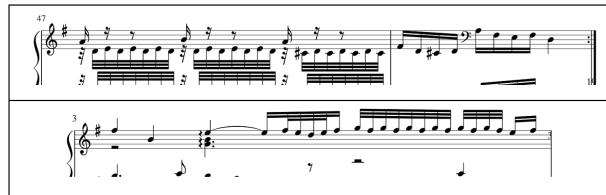


Figure 2. Examples from the dataset where the staff line is not cropped perfectly.

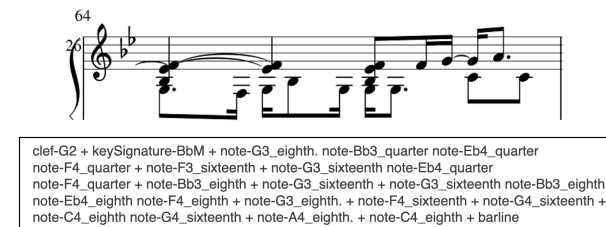


Figure 3. Example of the label encoding adopted in this paper.

phonic scores [11, 20, 21]. However, these datasets are either small in size or contain only monophonic scores.

3. MUSESORE POLYPHONIC DATASET

Given the size of datasets such as PrImuS [20] that have been used to show the effectiveness of an end-to-end architecture on monophonic OMR, we decided to come up with a dataset of similar size to do the same for polyphonic OMR. Using 19,432 MuseScore files available to download online [22], we were able to generate 124,671 varied-width single-staff images of exclusively polyphonic music along with their ground truth labels. To determine whether a given sample is polyphonic or not, we check if the music encoding defines multiple voices within a single measure.

To generate the dataset, we first used the MuseScore software plugin API (separate from the MuseScore forum mentioned above) to reduce the page height of the rendering to easily generate single staff line images. Then we executed a script to remove credit text which covered up some music symbols. After that, we used the MuseScore plugin API to generate MusicXML and PNG files from the MuseScore files. Lastly, we parsed the MusicXML to generate labels for each of the images, and removed sparse (music with only rests) and non-polyphonic data from the dataset.

The difficulty of the samples has quite a large range, from simple excerpts with two-note chords to dense notation as shown in Figure 1. Also, many samples are not perfectly cropped resulting in an additional implicit task of extracting the staff line from noisy environments. We argue however that this creates a more realistic environment as OMR systems should be able to handle these kinds of interferences shown in Figure 2.

For our experiments with this dataset, we used a 70/15/15 split for training, validation, and test respectively.

	MSPD			MSPD-Hard		
	Min	Mean	Max	Min	Mean	Max
Length (symbols)	3	70.59	819	41	79.2	679
Length (measures)	1	4.15	55	1	2.11	8
Density (symbols/measures)	3	20.3	165	41	52.8	165
Polyphony (voices)	2	2.05	4	2	2.42	4

Table 1. Statistics of the MuseScore Polyphonic Dataset (MSPD) and its hard subset (MSPD-Hard).

3.1 Data annotation

Due to our primary focus being polyphony, we chose to use a minimal symbol set sufficient to represent pitch and rhythm accurately, apart from tuplets. More specifically, we do not care about symbols such as dynamics, ties, tuplets, staccatos, accents, and other staff text. Instead, the only musical symbols we chose to label are clefs, key signatures, time signatures, barlines, and the notes themselves (pitch and rhythm). Inspired by the labeling scheme used to train models for end-to-end monophonic and homophonic OMR [4, 13], we approached the task of polyphonic OMR in the same way, with the ‘‘Advance position’’ encoding proposed by Alfaro et al. for representing homophonic music as a one-dimensional sequence reading from left to right. This encoding adds a ‘+’ symbol between each sequential occurrence of notes and symbols, and orders the individual notes of a chord from bottom to top, as seen in Figure 3. Throughout the rest of this paper, we will refer to the non-note music symbols described above (clefs, key signatures, time signatures, and barlines) as staff symbols.

3.2 MSPD-Hard

While we are interested in the performance on the average polyphonic images, we also want to have a means to push an OMR system to its limit so we can better determine an upper bound on the capabilities. To do this, we created a subset of the test set of all the samples with a density (defined by number of symbols per measure) of at least 41, resulting in 900 samples, which we will refer to as MSPD-Hard. The bottom three images in Figure 1 are samples from the hard subset.

The statistic that sticks out the most when comparing the two (see Table 1) is of course the density. While it may seem strange that the mean symbol length of the hard test set is similar to the full test set, we observed that this is because the more dense samples tend to contain measures that are filled with symbols, thus fewer measures can fit per image on average. Lastly, the hard subset samples tend to have a higher level of polyphony, measured by number of voices defined by the MusicXML files, as expected.

4. ARCHITECTURAL COMPARISON FOR POLYPHONIC OPTICAL MUSIC RECOGNITION

As we mentioned in Section 2, several training objectives have been used for end-to-end OMR. Due to many recent works showing the effectiveness of CTC [23] for OMR [2,

4, 13], we chose to train all of the considered architectures using the same objective. The equation below shows the objective that CTC aims to maximize, where y represents the target sequence, z the alignment, x the sequence of vertical image slices (i.e. fixed-width slices of a staff line as shown in Figure 4), and θ the model parameters.

$$\max_{\theta} P(y | x; \theta) = \max_{\theta} \sum_z P(y, z | x; \theta). \quad (1)$$

4.1 Architecture overview

At a high level, all of the architectures we compared share the same structure. The two components are an encoder and a decoder. The function of the encoder is firstly to extract features about the image while creating vertical slices through pooling, and secondly to give global context of the image encoding to each local image slice. The goal of the decoder is to use the representations created by the encoder and predict the symbols that are present in each of the vertical image slices. Finally, we use CTC to marginalize over the alignment of these slices for training. Due to the effectiveness of this end-to-end architecture on monophonic and homophonic music [4, 13], we chose to keep the encoder fixed and instead look to different decoding strategies that could be better suited for dense polyphonic music.

4.2 Encoder details

The encoder we use is the one described by Calvo-Zaragoza et al, with 2x less width pooling [21]. First, the image is fed through a deep convolutional neural network (CNN) to extract relevant learned features. These two-dimensional feature representations of each vertical image slice are then flattened to a vector to create a sequential representation suitable for a recurrent neural network (RNN). The purpose of the RNN in this context is to give some awareness of the surrounding representations to each image slice’s encoding, which is essential to help identify symbols that may span multiple image slices. We chose to use a bidirectional Long Short-Term Memory (LSTM) [24, 25] for its effectiveness in end-to-end OMR as empirically shown by Baro et al. [2]. A visualization of the encoder components can be seen in Figure 4.

4.3 Baseline decoder

The baseline architecture we evaluated includes the encoder mentioned above, followed by a simple decoder consisting of two parallel fully connected layers to classify the

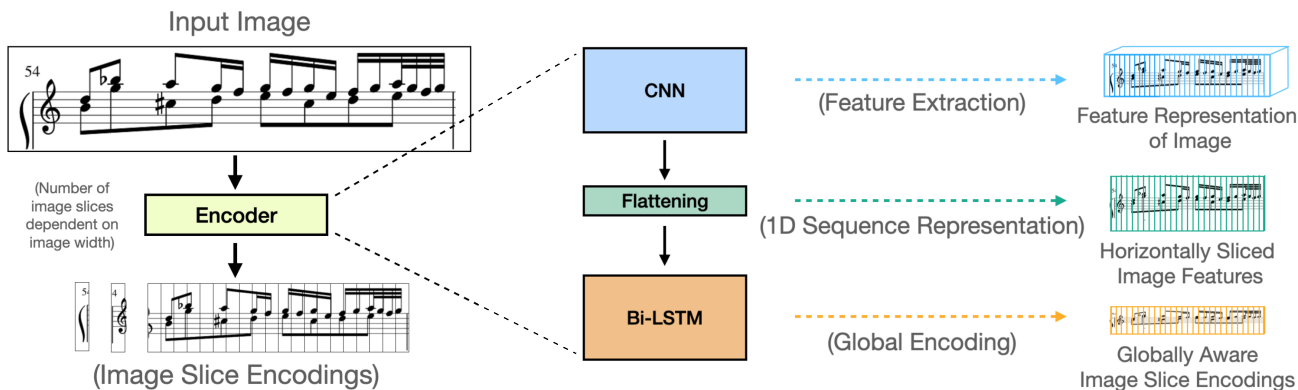


Figure 4. Illustration of the image encoder used in this paper.

pitch and rhythm of the symbol appearing in each vertical image slice. By design, this decoder is only capable of outputting a single symbol prediction at each image slice, a characteristic not ideal for polyphonic music.

4.4 FlagDecoder

To incorporate the multi-dimensional nature of polyphonic music while being trained on a one-dimensional sequence, we propose the BinaryVector. We observed that there are a fixed number of positions on a staff line that a note can appear on, indicating its pitch when combined with knowledge of the clef, key signature, and any accidentals. Based on this, one possible decoding for a vertical image slice could be an n -dimensional (where n is the sum of the number of note positions and the number of staff symbols) vector where the value of each dimension indicates whether or not the corresponding pitch or staff symbol appears in the image slice. This can be achieved using the sigmoid activation function. For inference, we apply a threshold of 0.5 to determine if a symbol is present or not.

Since we also need rhythm and accidental information for OMR, we modify this BinaryVector approach so that each note position on the staff has a corresponding rhythm and accidental classifier as opposed to a binary classifier that is only applicable for staff symbols. Visualizing this in Figure 5, there is a binary vector for the staff symbols, and a matrix for the notes, together resembling a flag hence the name FlagDecoder. To handle the case when two voices are playing the same note, we include two rows in the note matrix for each note position on the staff.

The FlagDecoder first uses two parallel fully connected (FC) layers for the note positions and staff symbols to reduce the dimensionality. Then the note latent representation goes through two separate FC layers (for rhythms and accidentals) to produce the note matrix, and the staff symbol latent representation is connected to another FC layer to produce the BinaryVector.

4.5 RNNDecoder

An alternative approach to those mentioned above, which maintain a single dimensional sequence, is to embrace the fact that polyphonic music is inherently multi-dimensional.

Based on the near perfect results that have been shown in previous published work on monophonic OMR with the baseline decoder, we took a new approach to the challenge of polyphony that breaks up the task into several simpler versions. Rather than representing polyphonic music as a one-dimensional sequence, we propose RNNDecoder, a recurrent decoder that is run across each image slice vertically, allowing for multiple outputs at a single image slice. This can ideally handle polyphony better than the baseline decoder, and can be trained trivially using the perspective that there are multiple one-dimensional sequences occurring from bottom to top in each image. To allow for alignment at inference time, we add a “noNote” symbol to the labels, as shown in Figure 6.

As we mentioned above, the RNNDecoder can generate a fixed number m of outputs per image slice. Since the largest number of notes and staff symbols we observed present in a single horizontal position of an image was 10, we chose to have $m = 10$ for our experiments. The key difference between the RNNDecoder and the baseline is that a hidden state is included while generating each output prediction, and is concatenated to the current image slice encoding before going through the classifier. More specifically, at each image slice as each output is generated, this hidden state is updated like the hidden state of an RNN as in Figure 5, and fed through the same classifier along with the image slice encoding 10 times. As shown in Figure 6, this new decoder can be trained without using the “Advance position” label encoding.

4.6 Objective functions

As we mentioned in Section 4, CTC aims to maximize the likelihood of a target symbolic sequence. For the baseline decoder, we use two target sequences – one for rhythm and one for pitch – and jointly minimize those two CTC losses. On the other hand for the FlagDecoder, we just use a single target sequence where each “symbol” is a unique flag configuration (see Figure 6), meaning it represents a combination of notes and staff symbols as opposed to just a single word from a symbol vocabulary. Lastly for the RNNDecoder, we use the same two target sequences used in the baseline, but we use 10 of them, thus optimizing the

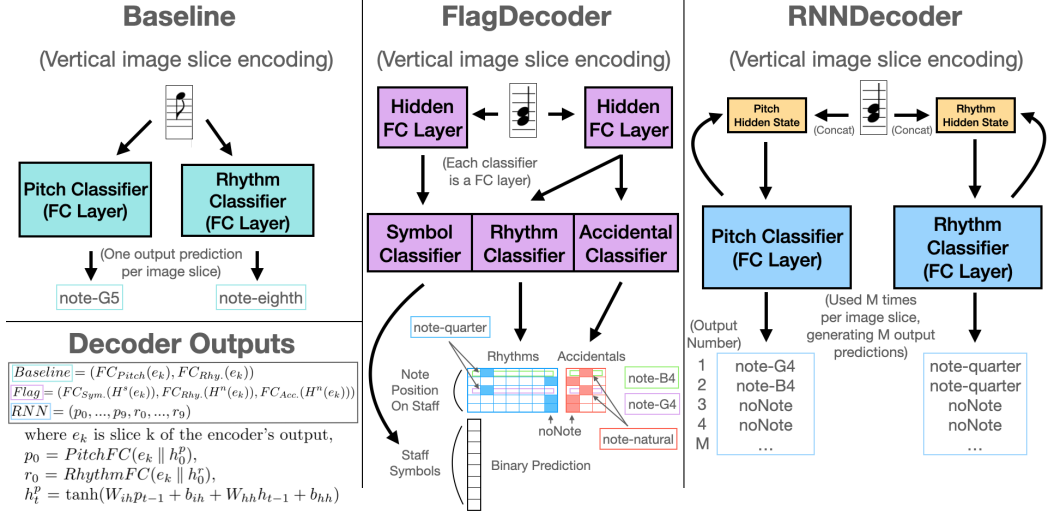


Figure 5. Illustrations of the three decoder architectures examined in this paper.

arithmetic mean of the CTC losses over the m sequences. The loss functions are shown below in Equation (2):

$$\begin{aligned}
 \mathcal{L}_{Baseline} &= \mathcal{L}_{Pitch} + \mathcal{L}_{Rhythm}, \\
 \mathcal{L}_{FlagDecoder} &= \mathcal{L}_{Flag}, \\
 \mathcal{L}_{RNNDecoder} &= \frac{1}{m} \sum_i^m (\mathcal{L}_{Pitch} + \mathcal{L}_{Rhythm}),
 \end{aligned} \tag{2}$$

where \mathcal{L}_{Pitch} , \mathcal{L}_{Rhythm} and \mathcal{L}_{Flag} are the corresponding CTC losses.

5. EXPERIMENTS

5.1 Implementation details

All models were trained using stochastic gradient descent with the Adam optimization algorithm [26]. We used a learning rate of 10^{-4} and a batch size of 16. Additionally, each image goes through a short preprocessing stage where they are inverted so that black pixels take on the highest value while white pixels are 0, and subsequently resized to a fixed height of 128 pixels.

5.2 Experiment setup

For evaluation, we follow recent literature [2] and use Pitch and Rhythm Symbol Error Rate (SER) as our evaluation metric of choice. This metric measures the edit distance at a symbol level between a predicted sequence and the ground truth sequence. More precisely, it can be written as the sum of insertions (I), deletions (D), and substitutions (S) normalized by the ground truth sequence length (N), i.e., $SER = (I + D + S)/N$.

In addition to staying consistent with previous literature, we also chose to evaluate Pitch and Rhythm SER separately to be able to compare the difficulties of the two tasks and gain insight on where improvements can be made. The two sets of data we used for evaluation are the full test set containing 18,700 images, and the hard test set containing 900 images, both of which were discussed in depth in Section 3.

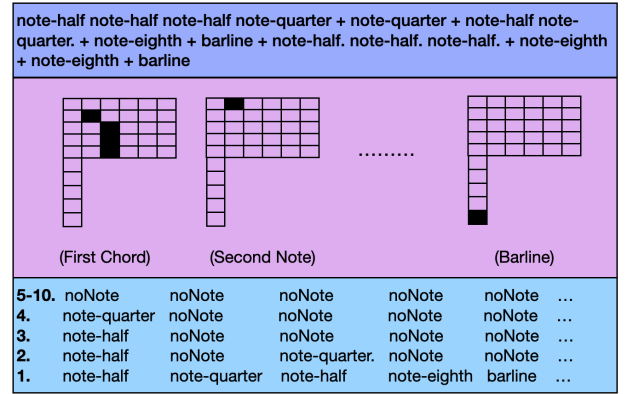


Figure 6. Examples of the different symbolic sequence representations used to optimize the decoders—(top) BaselineDecoder using “Advance position” encoding, (middle) FlagDecoder and (bottom) RNNDecoder, where number of outputs is 10.

5.3 Experiment results

We first compare the three different decoders using the full test set. An interesting result is that the Pitch SER is higher than the Rhythm SER across the board for all of the models. For the baseline and RNNDecoder, we believe this is due to the fact that pitch is affected by clef, thus when a clef prediction is off, or in particularly wide images, the correct clef may not be represented in an image slice encoding, resulting in incorrect decoding. Additionally when there are multiple neighboring notes together, identifying the correct pitches naturally seems more challenging than identifying the correct rhythms due to requiring more fine grained features to discern the exact pitches. We believe using an agnostic representation of pitch as described in [20] could potentially result in better pitch performance.

Decoder model	MSPD		MSPD-Hard	
	Rhythm SER (%)	Pitch SER (%)	Rhythm SER (%)	Pitch SER (%)
Baseline	7.39	10.28	14.11	18.83
FlagDecoder	6.67	9.82	9.86	17.98
RNNDecoder	3.92	5.64	5.82	8.57

Table 2. Performance of the three decoders on MuseScore Polyphonic Dataset (MSPD) and its hard subset (MSPD-Hard).

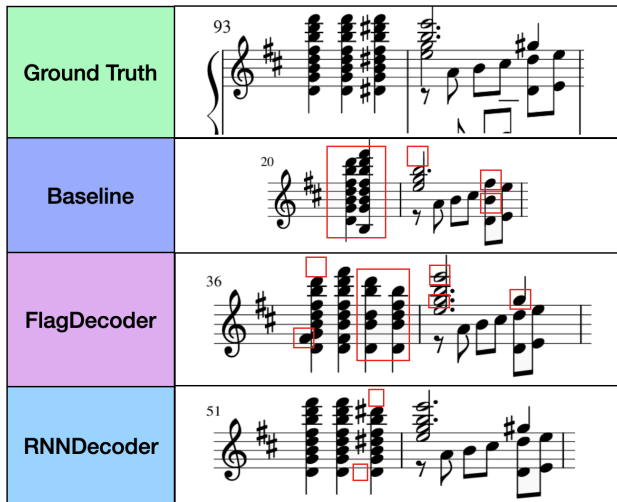


Figure 7. Comparison of the three decoders on an example with dense chords. Red blocks indicate the errors.

While our FlagDecoder only slightly outperforms the baseline on the full test set, the RNNDecoder appears to perform twice as well as the baseline on the full test set, achieving a new state-of-the-art performance. Further evaluating the performances of the new decoding methods on the hard test set (MSPD-Hard) highlights our decoders’ strengths over the baseline decoder. With the baseline, the error rate nearly doubles, whereas with the proposed decoding strategies, the error rate increases by a smaller factor when dealing with difficult data, with the Pitch SER of the FlagDecoder being the exception.

5.4 Error analysis

We also examine some qualitative results on examples from MSPD-Hard to examine the upper bound performance of the decoders. From Figure 7 we see that in the first measure which contains several huge chords nearby, the baseline is neither able to separate them by outputting a ‘+’ symbol nor output the correct number of notes. The FlagDecoder handles the first two chords well, but is likely thrown off by the sharps in the third chord. The RNNDecoder performs the best in the first measure, just missing a single note and accidental in the last chord, and deals with the polyphony in following measure perfectly. Figure 8 is challenging due to its high level of polyphony, which the baseline decoder clearly struggles with. In this example, both the FlagDecoder and RNNDecoder perform similarly, however the FlagDecoder actually handles the first chord

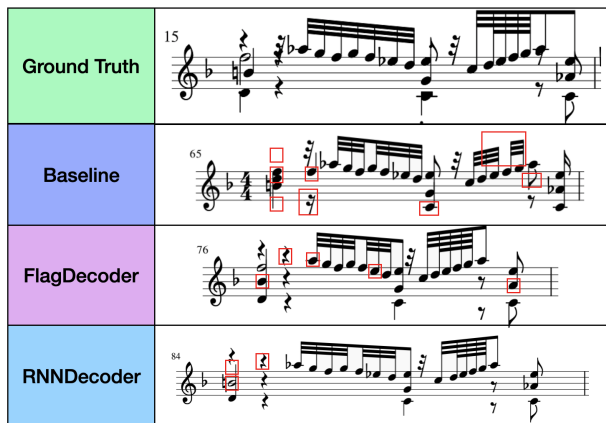


Figure 8. Comparison of the decoders on an example with four levels of polyphony. Red blocks indicate the errors.

with 4 voices correctly rhythmically whereas the RNNDecoder makes a minor mistake.

6. CONCLUSION

In this work, we first introduced a workflow for generating datasets from MuseScore files which will be beneficial for the research community given the massive amount of sheet music publicly available on the MuseScore forum. We then used this to create a large-scale dataset suitable for end-to-end polyphonic optical music recognition (OMR), and proposed two novel decoding strategies for the task, namely FlagDecoder and RNNDecoder. Further we performed an empirical comparison of the performance of these methods on polyphonic OMR, and observed a new state-of-the-art performance with the RNNDecoder.

One of the main limitations of this work is generalizability, a problem that many supervised machine learning systems struggle with. While our system can perform extremely well on images generated from the MuseScore engraver, it is not able to do well out of the box on music generated by other engravers. We are also aware that there are many viable implementations of the new decoding methods we proposed that could potentially give better performance, and hope to evaluate them in the future when we have the required computing resources. In addition, adapting these methods to multi-staff music will allow more versatility in usage. Lastly, we hope to address the challenge of generalizability in the future as it is currently one of the major barriers preventing neural network-based OMR systems from being deployed widely.

7. REFERENCES

- [1] J. Calvo-Zaragoza, J. Hajič jr., and A. Pacha, "Understanding optical music recognition," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–35, 2020.
- [2] A. Baró, P. Riba, J. Calvo-Zaragoza, and A. Fornés, "From optical music recognition to handwritten music recognition: A baseline," *Pattern Recognition Letters*, vol. 123, pp. 1–8, 2019.
- [3] "Musescore forum." [Online]. Available: <https://musescore.com>
- [4] J. Calvo-Zaragoza, J. J. Valero-Mas, and A. Pertusa, "End-to-end optical music recognition using neural networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [5] A. Pacha, K.-Y. Choi, B. Couasnon, Y. Ricquebourg, R. Zanibbi, and H. Eidenberger, "Handwritten music object detection: Open issues and baseline results," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, 2018.
- [6] L. Tuggener, I. Elezi, J. Schmidhuber, and T. Stadelmann, "Deep watershed detector for music object recognition," *arXiv preprint arXiv:1805.10548*, 2018.
- [7] J. Hajič jr., M. Dorfer, G. Widmer, and P. Pecina, "Towards full-pipeline handwritten omr with musical symbol detection by u-nets." in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [8] A. Pacha, J. Hajič jr., and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, p. 1488, 2018.
- [9] Z. Huang, X. Jia, and Y. Guo, "State-of-the-art model for music object recognition with deep learning," *Applied Sciences*, vol. 9, no. 13, p. 2645, 2019.
- [10] A. Pacha, J. Calvo-Zaragoza, and J. Hajič jr., "Learning notation graph construction for full-pipeline optical music recognition." in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [11] E. van der Wel and K. Ullrich, "Optical music recognition with convolutional sequence-to-sequence models," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [12] A. Baró, C. Badal, and A. Fornés, "Handwritten historical music recognition by sequence-to-sequence with attention mechanism," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2020, pp. 205–210.
- [13] M. Alfaro-Contreras, J. Calvo-Zaragoza, and J. M. Iñesta, "Approaching end-to-end optical music recognition for homophonic scores," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2019, pp. 147–158.
- [14] A. F. Desaedeleer, "Reading sheet music," Master's thesis, University of London, 2006.
- [15] A. Fornés, J. Lladós, and G. Sánchez, "Old handwritten musical symbol classification by a dynamic time warping based method," in *Graphics Recognition. Recent Advances and New Opportunities: 7th International Workshop (GREC)*, 2008.
- [16] A. Rebelo, G. Capela, and J. S. Cardoso, "Optical recognition of music symbols: A comparative study," *International Journal on Document Analysis and Recognition (IJ DAR)*, vol. 13, no. 1, 2010.
- [17] J. Calvo-Zaragoza and J. Oncina, "Recognition of pen-based music notation: The homus dataset," in *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR)*, 2014.
- [18] A. Pacha and H. Eidenberger, "Towards a universal music symbol classifier," in *14th International Conference on Document Analysis and Recognition (ICDAR)*, 2017, pp. 35–36.
- [19] J. Hajič jr. and P. Pecina, "The MUSCIMA++ dataset for handwritten optical music recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017.
- [20] J. Calvo-Zaragoza and D. Rizo, "End-to-end neural optical music recognition of monophonic scores," *Applied Sciences*, vol. 8, no. 4, 2018.
- [21] —, "Camera-PrIMuS: Neural end-to-end optical music recognition on realistic monophonic scores," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [22] "Musescore dataset." [Online]. Available: <https://github.com/Xmader/musescore-dataset>
- [23] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 369–376.
- [24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [25] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

A HARDANGER FIDDLE DATASET WITH PERFORMANCES SPANNING EMOTIONAL EXPRESSIONS AND ANNOTATIONS ALIGNED USING IMAGE REGISTRATION

Anders Elowsson

RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion; University of Oslo
anderseelowsson@gmail.com

Olivier Lartillot

olivier.lartillot@imv.uio.no

ABSTRACT

This paper presents a Hardanger fiddle dataset “HF1” with polyphonic performances spanning five different emotional expressions: normal, angry, sad, happy, and tender. The performances thus cover the four quadrants of the activity/valence-space. The onsets and offsets, together with an associated pitch, were human-annotated for each note in each performance by the fiddle players themselves. First, they annotated the normal version. These annotations were then transferred to the expressive performances using music alignment and finally human-verified. Two separate music alignment methods based on image registration were developed for this purpose; a B-spline implementation that produces a continuous temporal transformation curve and a Demons algorithm that produces displacement matrices for time and pitch that also account for local timing variations across the pitch range. Both methods start from an “Onsetgram” of onset salience across pitch and time and perform the alignment task accurately. Various settings of the Demons algorithm were further evaluated in an ablation study. The final dataset is around 43 minutes long and consists of 19 734 notes of Hardanger fiddle music, recorded in stereo. The dataset and source code are available online. The dataset will be used in MIR research for tasks involving polyphonic transcription, score alignment, beat tracking, downbeat tracking, tempo estimation, and classification of emotional expressions.

1. INTRODUCTION

1.1 Hardanger Fiddle Music

The Hardanger fiddle is a traditional stringed solo instrument played in the southern parts of Norway. It features resonance strings producing a characteristic resonating sound. The flat fingerboard and bridge enable the performer to play several strings simultaneously and the polyphony level of the music is generally 2. Fast trills are frequently used as ornaments. Lack of annotated audio excerpts makes data-driven research on Hardanger fiddle music hard and this study is an attempt to remedy the situation. Our vision is to create a dataset with annotated

pitched onsets and offsets so that accurate polyphonic transcription systems can be trained in future studies, enabling researchers to transcribe vast existing libraries of historical audio recordings.

1.2 Transcription Datasets in MIR

Researchers have used many different techniques to create annotated datasets for polyphonic transcription in the past. One method is to record individual voices in isolation to facilitate easier annotation. Examples include the four-voiced *Bach10* dataset [1], the *TRIOS* dataset [2] consisting of musical trios, a five-voiced woodwind recording [3], the audio-visual URMP dataset [4], and the *MedleyDB* multitracks dataset [5]. For polyphonic instruments, the annotation of many simultaneous notes can be cumbersome and time-consuming. Another method for those kinds of instruments has therefore been to generate the sounds and annotations directly from MIDI. The technique has been used for piano datasets [6-8], but has also been applied across the full range of the general MIDI instrument specification [9]. To increase the variability and the size of the dataset, researchers can use data augmentation, varying tempo, pitch, dynamics, and timbre during synthesis [9].

Although the MIDI generation strategy is appealing because of its efficiency, synthesized MIDI often lacks the full range of variation and complexities found in real performances. Researchers can in this case instead create datasets by synchronizing sheet music with an associated recording. This approach was adopted by Thickstun, et al. [10] who used dynamic time warping (DTW) applied to log-frequency spectrograms focused on lower frequencies.

1.3 Mood Datasets in MIR

Datasets spanning different moods/emotions are developed to enable researchers to train and test music emotion recognition (MER) systems. Many MER datasets use the valence-arousal model [11], with the valence and arousal variables annotated by human listeners. Examples include the *MoodSwings* [12], *Emotion in Music* [13], *AMG1608* [14], *DEAM* [15], and *PMemo* [16] datasets.

For a few datasets, performers have been asked to play the *same* piece of music with *different* emotional expressions. Li, et al. [17] asked violinists to perform classical compositions according to different expressive musical terms (e.g., *tranquillo*) and used the resulting dataset for modeling. Gabriëlsson and Juslin [18] asked performers to play with the emotional expressions “happy”, “sad”,



© A. Elowsson and O. Lartillot. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A. Elowsson and O. Lartillot, “A Hardanger Fiddle Dataset with Performances Spanning Emotional Expressions and Annotations Aligned using Image Registration”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

“angry”, “fearful”, “tender”, “solemn”, and “no expression”, analyzing the recordings both quantitatively and through listening tests. Performers control the musical expression by varying, e.g., phrasing, tempo, timing, articulation, and dynamics [19-24] and the perceptual aspect of such features has also been modeled extensively [25-27]. Note that these types of features are among those varied for data augmentation applied to MIDI (or audio files), but when they are introduced by real musicians, they will be richer in scope and better capture the variability that can be expected in other real performances. It is therefore appealing to create a dataset where each song is performed with several musical expressions, using music alignment to transfer annotations between the different performances. Not only will this bootstrap the annotation effort while retaining variation in the annotated notes, it will also introduce a new dataset for emotional expression, where researchers can, in extension to analyzing the audio files, utilize the annotations as a symbolic representation for MER. This strategy is therefore explored in this study.

1.4 Score Alignment

The task of aligning a musical score with an associated audio file has been fairly widely studied, with researchers often opting for various flavors of DTW. Implementations differ regarding how they compute a similarity metric/feature space for alignment. Researchers can either synthesize or add harmonics to the score [10, 28-30], convert both score and audio to a chroma-space [31], or alternatively learn the feature space for alignment [32-35], casting the task as an optimization problem.

The aforementioned strategies are aligning across full note lengths, but it is mainly the onsets that provide information about timing [36]. It has therefore been suggested that they can be improved by detecting onsets in the audio [30]. One strategy in this direction is to apply DTW to a half-wave rectified spectral flux (SF) [36]. Ewert, et al. [37] instead start from a chroma before computing the flux. Kwon, et al. [38] used a polyphonic pitch tracker to compute the feature space and found that the best results were achieved when including pitched onsets across the full 88-note range. This strategy concerning the feature space is the closest to our implementation, but we decided to forego DTW. Our motivation for, and implementation of, image registration techniques for music alignment are described in Section 3.

2. OVERVIEW AND MOTIVATION

Our primary objective with this study was to create a dataset of Hardanger fiddle music with annotated onsets and offsets. In particular, our focus was on the annotated onsets. Annotating Hardanger fiddle music is non-trivial. It is polyphonic and contains ornaments with very fast tone sequences. In our preliminary studies, we learned that it is rather time-consuming for Hardanger fiddle musicians to produce annotations for tunes that they are unfamiliar with, and accuracy may sometimes be lacking. Furthermore, our overarching project also strives to collect additional data on expressive Hardanger fiddle performances. These circumstances led to the following design:

1. Hardanger fiddle performers are tasked to record five versions of songs they are familiar with, using the expressions: *normal*, *sad*, *angry*, *happy*, and *tender*.
2. They annotate notes in the *normal* recording from scratch, using computer assistance tools as aid.
3. The *normal* recording is aligned with the expressive recordings using music alignment, so that the *normal* annotations can be automatically transferred to them.
4. Performers go through the aligned annotations and make adjustments to ensure that they are correct.

The strategy gives us a few advantages:

- *Does not introduce bias concerning timing.* Since the *normal* recording is annotated from scratch, and the score alignment only used for aligning the two audio recordings, we do not impose priors regarding the exact location of, e.g., onsets in the music, which would have been the case if an algorithm produces the initial annotations.
- *Ensures that annotators annotate songs they are familiar with.* It is easier to be accurate and efficient when annotating a song that you are familiar with, and note sheets are not exhaustive since they do not cover the rich ornamentation in Hardanger fiddle music.
- *Provides five times the training and testing data for polyphonic transcription.* With real performances of bowed instruments, the sound characteristics will vary each time a phrase is played. Thus, repeated sequences, particularly of ornaments, still provide training and testing data with high “entropy”.
- *Creates a dataset that can be used for additional tasks in future studies.* Our experimental design provides us with both audio and symbolic data of performances with varying emotional expressions. This data can be used to study how mood is expressed on the Hardanger fiddle and to develop music alignment systems.
- *Enables us to scale future annotation tasks within the same framework.* The method will connect each note in the expressive performances with the notes in the normal performance. Thus, if we assign higher-level features to these notes, such as their metrical position, we can automatically transfer that information to the expressive performances.

3. MUSIC ALIGNMENT ALGORITHMS

Tempo variations in music are often observed and modeled as gradual changes developing over several successive notes. Friberg [39] fitted “phrase arches” to piano performances, with *accelerando* in the start and *ritardando* in the end of the phrases. Other researchers fit their observations using spline-shaped profiles [40] or fit the final *ritardando* using a quadratic polynomial [41].

The DTW algorithm is “local” in scope and will not model differences in tempo and gradual tempo variations observed across longer sections. This means that it can, e.g., fail to accurately stretch matched notes of different lengths or, when the feature space is focused on onsets, fail to produce convincing tempo curves for sections where the feature space is empty. The resulting warping path can

therefore become rather irregular and is also discrete, not fitting to a finer scale than the time frame hop length. Various remedies have been proposed to alleviate these issues, for example introducing special silence frames to “stretch out” pauses between notes [28] or trying to smooth the warping path in post-processing [29]. This study explores if techniques developed for image registration can be useful as an alternative approach. Through a free form deformation with a B-spline grid [42] (Section 3.2), we optimize across multiple frames, utilizing a smoothness penalty to constrain neighboring grid points from moving independently while achieving sub-frame resolution. By adopting the Demons algorithm to music alignment (Section 3.3), we instead also test a 2-dimensional alignment approach, where individual pitch bins are allowed to diverge somewhat from the warping path in order to account for natural variations in timing between concurrent notes.

3.1 Onsetgram and Preprocessing

The temporal alignment is performed on a 2-dimensional “Onsetgram,” consisting of onset activations distributed across pitch and time. The onset activations are first computed using the polyphonic transcription system developed by Elowsson [9], trained on a wide variety of music. In that system, an initial network detects framewise f_0 activations, which are used to identify the contours of the music. An additional network then operates across each detected contour, computing an onset activation at each time frame of the contour. The smoothed thresholded onset activation function was used (cf. [Eqs. A8-A11, 9]). The onset activations were inserted at the corresponding pitch bin and time frame of the Onsetgram, which had a pitch resolution of 1 cent/bin. A Hann window of width 151 bins (cents) was then used to smooth the Onsetgram across pitch. Figure 1 shows the smoothed Onsetgram in green overlaying the f_0 activations in blue.

The pitch range of the Onsetgram was set to 2 semitones below the lowest annotated pitch to 2 semitones above the highest annotated pitch. The pitch resolution was also scaled down to 4 bins/semitone. To speed up processing, the hop size was set to 23.2 ms by keeping only every fourth time frame of the original Onsetgram.

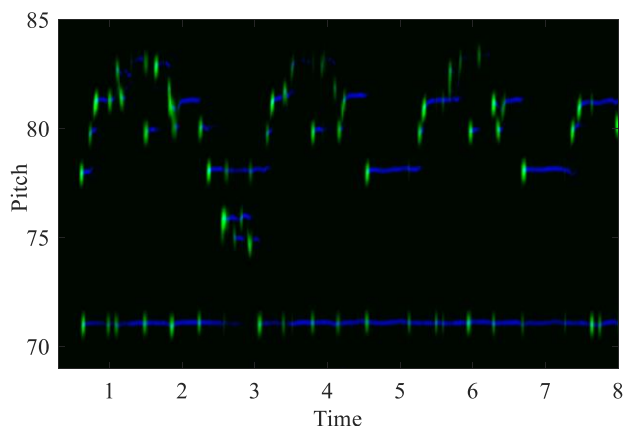


Figure 1. The Onsetgram used for music alignment in green overlaying f_0 activations in blue across which the onset activations were computed. The excerpt is from the song Haslebuskane, also featured in Figures 2 and 3.

Before applying the image registration algorithms, a start- and endpoint was computed for both audio files by finding the first and last time frame with a signal level within 10 dB of the average signal level of the audio file, as described by Elowsson and Friberg [43]. The *normal* Onsetgram was then re-scaled to have the same length as the Onsetgram of the emotional expression using linear interpolation. The annotations were also re-scaled using the same transformation.

3.2 B-spline Algorithm

The B-spline music alignment implementation uses low-level MATLAB functions for B-spline image registration from Kroon [44, 45]. The particular non-rigid B-spline alignment method was first introduced by Rueckert, et al. [42]. It is a free-form deformation with a B-spline grid, typically performed at multiple image scales (pyramid levels). For a precise mathematical formalization of the process, cf. [41, p. 64-65]. A multi-scale approach can be beneficial for two reasons – iterations performed at a coarser scale will converge fast, and the risk of reaching local minima is reduced. Since music may contain closely spaced repetitions, it seems reasonable to first align the coarser overall structure, ensuring that repetitions are not misaligned, and to then adjust notes at finer scales.

The temporal grid spacing for the first iteration was 256 frames (5.9 seconds), and at each subsequent iteration, this spacing was halved, ending with a grid spacing of 4 frames (93 ms) at the finest level. To avoid a too local scope with abrupt changes in the tempo curve at the finest level, the smoothness penalty of the B-spline implementation was used [44, 45]. This smoothness penalty constrains neighboring grid points from moving independently, simulating the bending energy of a thin plate of metal [42, 46]. We set the penalty to 0.3 at the finest pyramid level, halving it at each level such that it was 0.005 at the coarsest scale.

The pitch spacing was set such that the whole pitch dimension of the image was contained between two grid points at all pyramid levels, and the pitch dimension of these grid points reset after optimizing at each level.

After fitting the normal Onsetgram to the Onsetgram of the emotional expression, the resulting forward transformation field was applied to the annotations, changing their

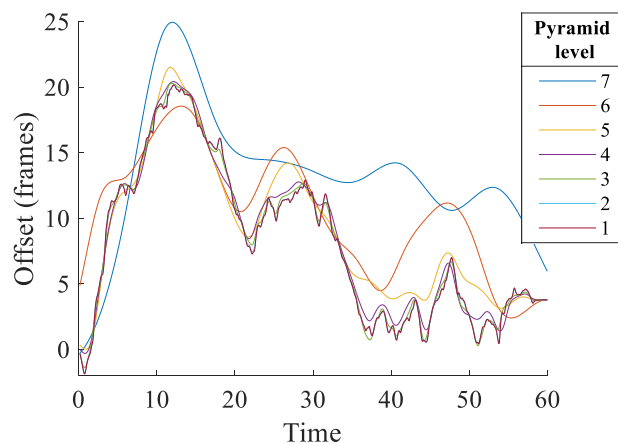


Figure 2. The forward transformation field at each pyramid level for aligning the *sad* and *normal* recordings of the tune Haslebuskane.

timing using linear interpolation. The aligned annotations were finally “tuned” as described in Section 3.4. Figure 2 shows the transformation field for all seven pyramid levels when aligning the *sad* and *normal* recording of the tune Haslebuskane.

3.3 Accelerated “Demons” Algorithm

The diffusion model known as the *Demons* algorithm for non-rigid image registration was introduced by Thirion [47]. It uses the gradient $\vec{\nabla}f$ from the fixed image f to compute a “demons” force for deforming a moving image m . Wang, et al. [48] modified the algorithm by also including the gradient of the moving image $\vec{\nabla}m$, using bi-directional forces,

$$\vec{u} = (m - f) \times \left(\frac{\vec{\nabla}f}{|\vec{\nabla}f|^2 + \alpha^2(f-m)^2} + \frac{\vec{\nabla}m}{|\vec{\nabla}m|^2 + \alpha^2(f-m)^2} \right). \quad (1)$$

The normalization factor α introduced by Cachier, et al. [49] allows the force strength to be adjusted adaptively in each iteration. The displacement field \vec{u} is computed for both time (\vec{u}_x) and pitch (\vec{u}_y) deformations in each iteration and added to the corresponding overall displacement fields T_x (time) and T_y (pitch). We used this “accelerated Demons” algorithm, operating over 7 pyramid-levels with 70 iterations at each level, setting α to 0.4 as proposed by Wang, et al. [48], using the basic demon example code from Kroon and Slump [50] as a starting point but adapting the registration to the music alignment task. The Onsetgram of the recording with an emotional expression was used as the moving image and the Onsetgram of the normal recording used as the fixed image. The computed displacement field could then be used as a backward transformation to transfer the annotations to the recordings with emotional expressions.

In its original formulation, the computed displacements \vec{u}_x and \vec{u}_y for each iteration is smoothed before being added to the overall displacement fields T_x and T_y . We instead opted to smooth T_x and T_y directly in each iteration. To understand why this improves performance, recall that the Onsetgram is sparse and that we must be able to accurately move annotations between locations in the moving and fixed image that contain no salience information (e.g., offsets). By applying the smoothing operator directly to T_x and T_y , we iteratively “saturate” the displacement field with deformations also at locations where no gradients can be found in the Onsetgrams. This process also helps us smooth out irregular displacements resulting from erroneous transcriptions. The smoothing was done using Hann windows of length 33 across time and length 3 across pitch for T_x and length 17 and 3 for T_y . The reader is further referred to Cachier, et al. [49] for a discussion concerning the benefits of smoothing operations applied at various stages of the process.

Restrictions were set on T_x and T_y to ensure that the deformations were not bigger than desirable from a music-theoretical standpoint. For T_x , during each iteration before smoothing, we thresholded the displacement at each bin to not diverge more than 100 ms from the average displacement in each time frame. This means that annotations at different pitches can be moved freely but not diverge relative to each other too much. Thus, an annotation of a bass

note and a note in the treble where the bass note is played slightly before the treble note in the fixed image, but where circumstances are reversed in the moving image, can be transferred receiving correct timing, but never to such an extent that the interpretation of the score would be vastly different (>100 ms). For T_y , a fixed threshold of 70 cents was instead used, such that the pitch could not be displaced more than this.

The displacements fields (backward transformations) were applied to the annotations, changing their timing using linear interpolation. Since the incorporation of a threshold on T_x could hinder the algorithm from displacing time globally, the mean displacement for \vec{u}_x across all pitch bins is also added to T_x before thresholding and smoothing. Furthermore, since the Onsetgram only activates at onsets, T_y may not be particularly suitable for tuning the annotations. As a default, the post-processing step for tuning (Section 3.4) was instead applied. However, applying T_y directly for tuning was tested in the ablation

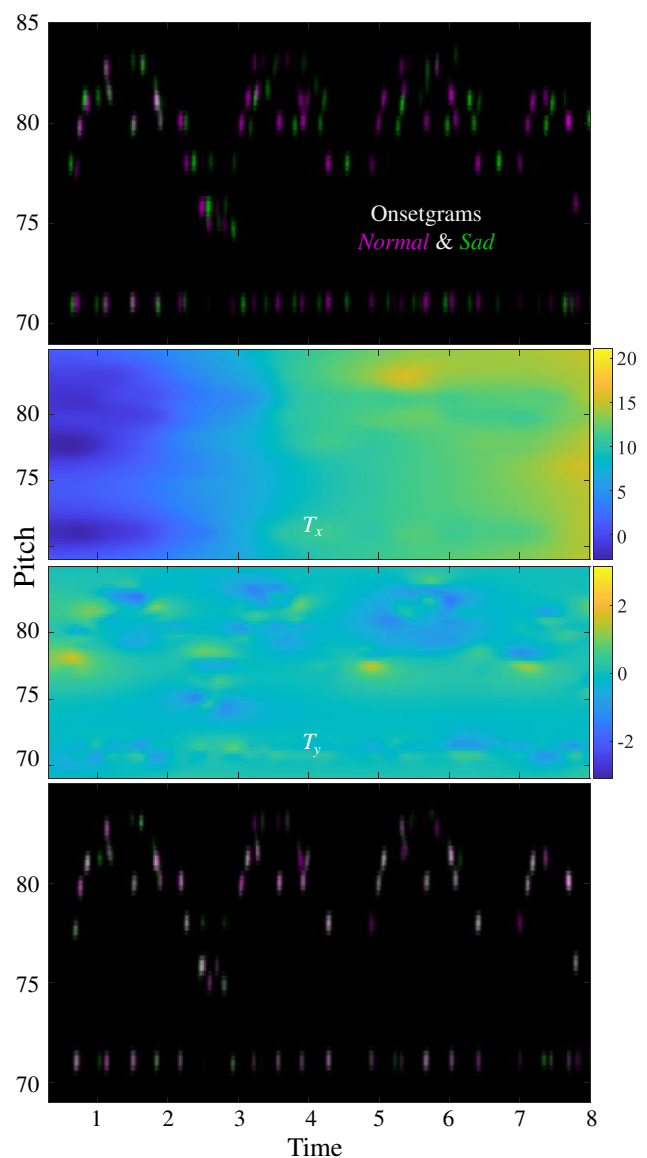


Figure 3. The Onsetgrams of both the *normal* and *sad* recordings of the tune Haslebuskane (pane 1), the backward transformation (displacement) fields T_x and T_y (panes 2 and 3), and the aligned Onsetgrams (pane 4).

study in Section 5.2. Figure 3 shows the Onsetgrams of both the *normal* and *sad* recordings of the tune Haslebuskane (pane 1), the displacement fields T_x and T_y (panes 2 and 3), and the aligned Onsetgrams (pane 4).

3.4 Tuning

A method for adjusting the pitch of each note was added as a post-processing step, motivated by the fact that the fiddle does not have fixed frets and the pitch of individual notes can vary relatively much. For each annotation to be tuned, a rectangular area was first extracted from the f_0 activations (blue in Figure 1), bounded by the onset and offset and extending 100 cents in both pitch directions from the annotated pitch. The average across time was computed and the resulting pitch vector smoothed with a Hann filter 41 cents wide. Only smoothed parts computed without zero-padded edges were kept, making the length 80 cents in both pitch directions. Peaks were detected and weighted based on how close they were to the annotated pitch as well as their pitch salience magnitude, opting to select the peak with the highest computed weight, and moving the annotation to its pitch.

Performances may drift “locally” in pitch through intonation on the fingerboard, such that the pitch of notes in short phrases with no open strings all are a bit higher or lower than in another recording of the same song. The tuning algorithm adapts to this by not allowing one note to be changed more than 45 cents in relation to the weighted average tuning change of other notes close in time and pitch. Due to space constraints, the reader is referred to the MATLAB implementation and its corresponding help text for precise details on all settings for the tuning algorithm.

4. DATASET

4.1 Recording and Annotation

The recordings were done by two Hardanger fiddle musicians, Henrik Nordtun Gjertsen (HNG) and Astrid Garmo (AG), who were students at the Norwegian Academy of Music. They recorded well-known Hardanger fiddle tunes in a relatively dry room in stereo using a Zoom H6 recorder.

The annotations were done by the same musicians using the software Annotemus¹ developed in MATLAB. Annotemus has a graphical user interface and provides functionality for creating annotations on top of a graphical representation of the audio file. We used the f_0 activations shown in blue in Figure 1 for this purpose. The aligned annotations were all initially created using the B-spline method which was being developed in conjunction with the annotation process.

The performers could use various key commands as an aid during annotation. This includes audio playback of the current window, playback between the start and end of one or several selected notes, playback that starts prior to a selected annotated note and ends at the annotated onset position, playback with a click at each annotated onset position, and playback with a synthesized version of the annotated score played in one of the stereo channels. The performers were instructed to first try the playback that ends at the

annotated onset position for locating the exact onset times for the *normal* recording and the click and synthesized functionality for verifying annotations, but were free to use whichever method they felt most comfortable with.

All playback functionality is offered with the option of slowing it down to an arbitrary speed selected by the annotator. Since Hardanger fiddle music contains frequent sequences of very fast note successions, the slowdown functionality was used extensively during the annotation process. The onset timing evaluation condition for polyphonic transcription is usually set to 50 ms. This means that we can only allow a very narrow margin of error for the annotations to ensure that they can be reliably used for evaluation. We encouraged performers to be very careful regarding onsets, and try to keep errors within 20 ms. Listeners notice time-displacements of just 10 ms on average [51], but since fiddle music has rather undefined transients at onsets, a narrower margin than 20 ms is very hard to achieve. For both annotators, their first annotations were rejected, and they were encouraged to improve the quality regarding aspects that did not meet our high standards.

4.2 Dataset Overview

The final dataset consists of 19 734 annotated notes across 40 stereo recordings of 8 tunes. The audio recordings and annotations are available online,² as well as MATLAB source code.³ The dataset is summarized in Table 1.

Title	Notes	Length	ID
Haslebuskane	2 828	4:35	HNG
Havbrusen	4 114	8:50	HNG
Ivar Jorde	1 665	3:52	AG
Låtten som bed om noko	1 819	4:51	AG
Signe Uladalen	2 177	4:30	AG
Silkjegulen	2 906	5:38	HNG
Valdresspringar	1 692	3:49	AG
Vossarull	2 533	6:34	HNG
Total	19 734	42:38	

Table 1. The eight tunes of the dataset, each performed with five different emotional expressions. The number of notes and the length of the recordings are computed as the total across the five variations. The ID identifies the musician. The last row provides totals across the dataset.

5. MUSIC ALIGNMENT EVALUATION

5.1 Main Results

The performance of the two methods was evaluated by matching onsets aligned from the *normal* version with the human-verified onset of the *expressive* version and measuring their distance. The two aligned recordings frequently vary, e.g., in ornaments, which means that many notes will not have a counterpart in the other recording. To account for this, we used weighted bipartite matching to first

¹ <https://www.uio.no/ritmo/english/projects/mirage/software/>
² <https://www.uio.no/ritmo/english/projects/mirage/databases/>

³ <https://github.com/aelowsson/music-alignment>

connect onsets of the two recordings, where the weight for how well a pair matches falls using a half Hann window up to a distance of 5 seconds and 70 cents respectively. Regular unweighted bipartite matching is not ideal in this circumstance since it can create incorrectly matched pairs containing ornaments with no counterparts, whenever two such ornaments, one in each recording, are within 5 seconds of a real correct pair of onsets with a similar pitch. The F-measure \mathcal{F} was measured for the matched onset pairs only, leaving out the around 3 % of onsets with no counterpart that were unmatched.

Table 2 shows the results, with the F-measure for onsets within 80 ms (\mathcal{F}_{80}) highlighted in bold. We note that the Demons algorithm was more accurate even though the B-spline method was used as a starting point for the aligned expressive performances. Since this algorithm is also faster (the full dataset aligned in 2.5 minutes on an i7-6700K processor), it was our focus in the ablation study.

	\mathcal{F}_{50}	\mathcal{F}_{80}	\mathcal{F}_{150}	\mathcal{F}_{300}	<i>Avg</i>
B-spline	91.1	95.9	98.2	99.2	28.9 ms
Demons	95.4	98.3	99.1	99.5	23.0 ms

Table 2. F-measures at different distance metrics as well as the average distance (*Avg*) between matched onsets for the B-spline and Demons music alignment methods.

5.2 Ablation Study

Various settings of the Demons algorithm were tested in an ablation study:

- **T_x Thresh:** Instead of a 100 ms threshold we tested a strict zero threshold (*0*) or used no threshold (*None*).
- **T_y :** Foregoing the use of T_y completely (*No T_y*), also skipping the tuning stage (*No TT*), applying T_y to the annotations instead of using the tuning algorithm (*Apply*), or using the default setting but without thresholding (*No Th*).
- **T_x Mean:** Testing to *not* add the mean displacement for \vec{u} to T_x before thresholding and smoothing (*None*).
- **\vec{u}_x :** Smoothing \vec{u}_x instead of smoothing T_x , tested across time (\vec{u}_x *Ti*), time and pitch (\vec{u}_x *TP*), or across pitch only (\vec{u}_x *Pi*).
- **T_x Smooth:** Smoothing T_x across time with shorter or longer Hann windows (*15* or *45*).

Figure 4 shows the results of the ablation study as the difference in performance at \mathcal{F}_{80} . The 95 % confidence intervals (CIs) illustrated with black bars were derived from the difference in \mathcal{F}_{80} for individual tunes between the default setting and the tested setting. This difference was sampled with replacement from the tunes $8 \times 4 = 32$ times to compute a single overall outcome, and the procedure repeated 10^6 times to compute a distribution of possible outcomes, from which the 5th and 95th percentile could be extracted.

6. CONCLUSIONS

We have created an annotated Hardanger fiddle dataset with performances spanning five emotional expressions.

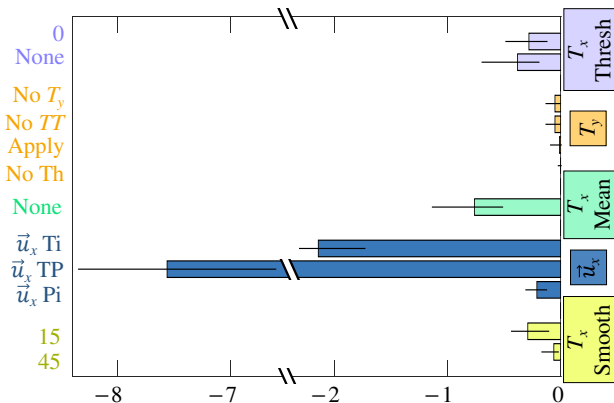


Figure 4. The results of the ablation study for the Demons method, showing the change in F-measure relative to the default setting. Black bars indicate 95 % CIs. Note that the x-axis has been spliced to accommodate the lower result for the \vec{u}_x *TP* setting.

The process of creating accurate note annotations for real polyphonic instrument recordings can be cumbersome, and we hope that the developed techniques and source code can be useful to other researchers in the field.

Two music alignment algorithms based on image registration were created and analyzed. The Demons algorithm is faster and easier to adapt to music and it also produces the best alignments. It can be noted that the alignment is evaluated using two separate annotations, so if a matched pair of notes have annotations that are 40 ms off each, they may just fail on the \mathcal{F}_{80} evaluation metric even if the alignment is performed perfectly. Furthermore, ornaments with no counterpart (see Section 5.1) may still be erroneously matched if they are within 5 seconds of each other. Thus, even with a few missed notes on the \mathcal{F}_{80} metric, we can still suspect that the alignment is very accurate overall. Informal closer analysis of the alignments also indicates that this is the case.

The ablation study indicates that the proposed default settings for the Demons algorithm are well-adjusted. We note that smoothing across T_x instead of \vec{u}_x is an important ingredient for successful Demons music alignment. The 100 ms threshold for individual pitch bin displacements in T_x relative to the mean displacement is an important addition (T_x Thresh), and should be combined with adding the mean displacement for \vec{u} to T_x before thresholding and smoothing (T_x Mean).

We intend to expand the annotations to also contain higher-level metrical information. Furthermore, we intend to develop models for polyphonic transcription and MER based on the dataset, something that we hope other research groups will do as well.

7. ACKNOWLEDGMENTS

We thank Astrid Garmo and Henrik Nordtun Gjertsen for their fiddle performances and annotations. This study was supported by the Research Council of Norway through its Centers of Excellence scheme, project number 262762, and the MIRAGE project, grant number 287152. The ablation study was performed on resources provided by UNINETT Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway.

8. REFERENCES

- [1] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE TASLP*, vol. 18, no. 8, pp. 2121-2133, 2010.
- [2] J. Fritsch and M. D. Plumbley, "Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 888-891: IEEE.
- [3] M. Bay, A. F. Ehmann, and J. S. Downie, "Evaluation of Multiple-F0 Estimation and Tracking Systems," in *ISMIR*, 2009, pp. 315-320.
- [4] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522-535, 2018.
- [5] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research," in *ISMIR*, 2014, vol. 14, pp. 155-160.
- [6] G. E. Poliner and D. P. Ellis, "A discriminative model for polyphonic piano transcription," *EURASIP J. on Adv. in Signal Proc.*, vol. 2007, no. 1, 2006.
- [7] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643-1654, 2010.
- [8] C. Hawthorne *et al.*, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," *arXiv preprint arXiv:1810.12247*, 2018.
- [9] A. Elowsson, "Polyphonic pitch tracking with deep layered learning," *Journal of the Acoustical Society of America*, vol. 148, no. 1, pp. 446-468, 2020.
- [10] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," in *International Conference on Learning Representations (ICLR)*, 2017.
- [11] J. Posner, J. A. Russell, and B. S. Peterson, "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology," *Development and psychopathology*, vol. 17, no. 3, p. 715, 2005.
- [12] J. A. Speck, E. M. Schmidt, B. G. Morton, and Y. E. Kim, "A Comparative Study of Collaborative vs. Traditional Musical Mood Annotation," in *ISMIR*, 2011, vol. 104, pp. 549-554: Citeseer.
- [13] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," in *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*, 2013, pp. 1-6.
- [14] Y.-A. Chen, Y.-H. Yang, J.-C. Wang, and H. Chen, "The AMG1608 dataset for music emotion recognition," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 693-697: IEEE.
- [15] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Developing a benchmark for emotional analysis of music," *PloS one*, vol. 12, no. 3, p. e0173392, 2017.
- [16] K. Zhang, H. Zhang, S. Li, C. Yang, and L. Sun, "The pmemo dataset for music emotion recognition," in *Proceedings of the 2018 acm on international conference on multimedia retrieval*, 2018, pp. 135-142.
- [17] P.-C. Li, L. Su, Y.-H. Yang, and A. W. Su, "Analysis of Expressive Musical Terms in Violin Using Score-Informed and Expression-Based Audio Features," in *ISMIR*, 2015, pp. 809-815.
- [18] A. Gabrielsson and P. N. Juslin, "Emotional expression in music performance: Between the performer's intention and the listener's experience," *Psychology of music*, vol. 24, no. 1, pp. 68-91, 1996.
- [19] A. Friberg, "A quantitative rule system for musical performance," KTH Royal Institute of Technology, 1995.
- [20] A. Friberg, "Generative rules for music performance: A formal description of a rule system," *Computer Music Journal*, vol. 15, no. 2, pp. 56-71, 1991.
- [21] R. Bresin and A. Friberg, "Emotional coloring of computer-controlled music performances," *Computer Music Journal*, vol. 24, no. 4, pp. 44-63, 2000.
- [22] R. Bresin and A. Friberg, "Emotional expression in music performance: synthesis and decoding," *TMH-QPSR*, vol. 39, pp. 085-094, 1998.
- [23] R. Bresin and G. Umberto Battel, "Articulation strategies in expressive piano performance analysis of legato, staccato, and repeated notes in performances of the andante movement of Mozart's sonata in g major (k 545)," *Journal of New Music Research*, vol. 29, no. 3, pp. 211-224, 2000.
- [24] A. Gabrielsson and P. N. Juslin, *Emotional expression in music*. Oxford University Press, 2003.
- [25] A. Friberg, E. Schoonderwaldt, A. Hedblad, M. Fabiani, and A. Elowsson, "Using listener-based perceptual features as intermediate representations in music information retrieval," *Journal of the Acoustic Society of America (JASA)*, vol. 136, no. 4, pp. 1951-1963, 2014.
- [26] A. Elowsson and A. Friberg, "Predicting the perception of performed dynamics in music audio with ensemble learning," *Journal of the Acoustic Society of America (JASA)*, vol. 141, no. 3, pp. 2224-2242, 2017.
- [27] A. Elowsson, A. Friberg, G. Madison, and J. Paulin, "Modelling the Speed of Music using Features from

- Harmonic/Percussive Separated Audio," presented at the ISMIR, 2013.
- [28] N. Orio and D. Schwarz, "Alignment of monophonic and polyphonic music to a score," in *International Computer Music Conference (ICMC)*, 2001, pp. 1-1.
- [29] R. J. Turetsky and D. P. Ellis, "Ground-truth transcriptions of real music from force-aligned midi syntheses," 2003.
- [30] F. Soulez, X. Rodet, and D. Schwarz, "Improving polyphonic and poly-instrumental music to score alignment," 2003.
- [31] N. Hu, R. B. Dannenberg, and G. Tzanetakis, "Polyphonic audio matching and alignment for music retrieval," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*, 2003, pp. 185-188: IEEE.
- [32] R. Lajugie, P. Bojanowski, P. Cuvillier, S. Arlot, and F. Bach, "A weakly-supervised discriminative model for audio-to-score alignment," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2484-2488: IEEE.
- [33] R. Lajugie, D. Garreau, F. R. Bach, and S. Arlot, "Metric Learning for Temporal Sequence Alignment," in *NIPS*, 2014.
- [34] Ö. İzmirlı and R. B. Dannenberg, "Understanding Features and Distance Functions for Music Sequence Alignment," in *ISMIR*, 2010, pp. 411-416: Citeseer.
- [35] C. Joder, S. Essid, and G. Richard, "Learning optimal features for polyphonic audio-to-score alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 10, pp. 2118-2128, 2013.
- [36] S. Dixon and G. Widmer, "MATCH: A Music Alignment Tool Chest," in *ISMIR*, 2005, pp. 492-497.
- [37] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1869-1872: IEEE.
- [38] T. Kwon, D. Jeong, and J. Nam, "Audio-to-score alignment of piano music using RNN-based automatic music transcription," presented at the Sound and Music Computing conference (SMC), 2017.
- [39] A. Friberg, "Matching the rule parameters of Phrase arch to performances of" Träumerei": A preliminary study," *STL-QPSR*, vol. 36, no. 2-3, pp. 063-070, 1995.
- [40] J. Feldman, D. Epstein, and W. Richards, "Force dynamics of tempo change in music," *Music perception*, vol. 10, no. 2, pp. 185-203, 1992.
- [41] A. Friberg and J. Sundberg, "Does music performance allude to locomotion? A model of final ritardandi derived from measurements of stopping runners," *The Journal of the Acoustical Society of America*, vol. 105, no. 3, pp. 1469-1484, 1999.
- [42] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: application to breast MR images," *IEEE transactions on medical imaging*, vol. 18, no. 8, pp. 712-721, 1999.
- [43] A. Elowsson and A. Friberg, "Modeling Music Modality with a Key-Class Invariant Pitch Chroma CNN," in *20th International Society for Music Information Retrieval Conference ISMIR*, Delft, Netherlands, 2019, pp. 541-548.
- [44] D.-J. Kroon, *Segmentation of the mandibular canal in cone-beam CT data*. Citeseer, 2011.
- [45] D.-J. Kroon, "B-spline Grid, Image and Point based Registration," ed. MATLAB Central File Exchange, 2020.
- [46] C. R. Meyer *et al.*, "Demonstration of accuracy and clinical versatility of mutual information for automatic multimodality image fusion using affine and thin-plate spline warped geometric deformations," *Medical image analysis*, vol. 1, no. 3, pp. 195-206, 1997.
- [47] J.-P. Thirion, "Image matching as a diffusion process: an analogy with Maxwell's demons," *Medical image analysis*, vol. 2, no. 3, pp. 243-260, 1998.
- [48] H. Wang *et al.*, "Validation of an accelerated 'demons' algorithm for deformable image registration in radiation therapy," *Physics in Medicine & Biology*, vol. 50, no. 12, p. 2887, 2005.
- [49] P. Cachier, X. Pennec, and N. Ayache, "Fast non rigid matching by gradient descent: Study and improvements of the" demons" algorithm," Inria, 1999.
- [50] D.-J. Kroon and C. H. Slump, "MRI modality transformation in demon registration," in *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2009, pp. 963-966: IEEE.
- [51] A. Friberg and J. Sundberg, "Perception of just-noticeable time displacement of a tone presented in a metrical sequence at different tempos," *The Journal of The Acoustical Society of America*, vol. 94, no. 3, pp. 1859-1859, 1993.

BUILDING THE METAMIDI DATASET: LINKING SYMBOLIC AND AUDIO MUSICAL DATA

Jeff Ens
Simon Fraser University
jeffe@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

ABSTRACT

We introduce the MetaMIDI Dataset (MMD), a large scale collection of 436,631 MIDI files and metadata. MMD contains artist and title metadata for 221,504 MIDI files, and genre metadata for 143,868 MIDI files, collected during the web-scraping process. MIDI files in MMD were matched against a collection of 32,000,000 30-second audio clips retrieved from Spotify, resulting in over 10,796,557 audio-MIDI matches. In addition, we linked 600,142 Spotify tracks with 1,094,901 MusicBrainz recordings to produce a set of 168,032 MIDI files that are matched to the MusicBrainz database. We also provide a set of 53,496 MIDI files using audio-MIDI matches where the derived metadata on Spotify is a fuzzy match to the web-scraped metadata. These links augment many files in the dataset with the extensive metadata available via the Spotify API and the MusicBrainz database. We anticipate that this collection of data will be of great use to MIR researchers addressing a variety of research topics.

1. INTRODUCTION

Large-scale metadata-rich MIDI datasets containing audio-MIDI matches [1–3] are indispensable in a wide variety of research contexts. For example, the Lakh Midi Dataset (LMD) [3] has been applied in many different contexts, including training generative music systems [4, 5], tempo-estimation [6], genre classification [7] and even as a primary data-source for new datasets [8, 9]. Motivated by the widespread demand for datasets of this nature, we created the MetaMIDI Dataset (MMD), which contains 2.4 times the number of MIDI files in the LMD, and audio-MIDI matches associating MIDI files with Spotify and MusicBrainz. To put the following numbers into context, we note that there is a many-to-one relationship between Spotify track ids and the actual audio recording. In this paper, we describe the process of assembling the dataset, which consists of the following contributions:

- Collection of 436,631 MIDI files.
- Scraped artist + title metadata for 221,504 MIDI files (10 times more than the LMD).

- Scraped genre metadata for 143,868 MIDI files.
- An improved audio-MIDI matching procedure, which produced 10,796,557 audio-MIDI matches linking 237,236 MIDI files to one or more tracks on Spotify.
- 829,728 high reliability audio-MIDI + scraped metadata (artist and title) matches linking 53,496 MIDI files to one or more tracks on Spotify.
- A method for linking Spotify tracks and MusicBrainz recordings, producing 8,263,482 unique links that associate 1,094,901 MusicBrainz recordings with 600,142 Spotify tracks.
- 168,032 MIDI files matched to MusicBrainz IDs via the Spotify/MusicBrainz linking procedure.

2. DATA COLLECTION

We scraped publicly available websites and were able to amass a collection of 436,631 unique MIDI files. Candidate websites were selected using a search engine to query various phrases including keywords such as MIDI, music, and a variety of musical genres. A list of the sites scraped and the number of MIDI files found on each site is provided in the dataset. Where possible, we also collected additional metadata, such as the artist, title and genre of associated with a particular MIDI file.

3. AUDIO MIDI MATCHING

To augment MMD with additional metadata, we match the MIDI files against a large metadata-rich collection of audio clips. Although the LMD is comprised of audio-MIDI matches against the Million Song Dataset [10], we decided to use 30-second preview clips made available through the Spotify API¹. The primary motivation for this decision was the fact that the Spotify API provides over an order of magnitude more data. Using the Spotify API, we were able to collect 32,000,000 30-second MP3 files (over 13TB of raw data). To compute the audio-MIDI matches, we model our approach after the procedure employed by Raffel [3], who matched the 176,581 MIDI files in the LMD with 1,000,000 audio files in the Million Song Dataset [10]. However, we make some modifications to the matching algorithm to accommodate the large amount of data which was collected.

¹ <https://developer.spotify.com/documentation/web-api/>



Raffel’s audio-MIDI matching procedure is comprised of two stages [3]. In the first stage, which we refer to as the blocking stage, audio-MIDI pairs which are unlikely to be a match are removed from consideration. In the second stage, which we refer to as the matching stage, a confidence score (on the range [0,1]) is computed for each remaining audio-MIDI pair. To be considered a valid match, the audio-MIDI pair must have a confidence score greater than 0.5. To compute the confidence score for an audio-MIDI pair, Raffel computes the Constant-Q Transform (CQT) [11] for the audio file and the audio-rendered MIDI file, using 48 logarithmically-spaced bins from C2 to B5 (12 bins per octave). Then, the dynamic time warping (DTW) algorithm is used to find the optimal alignment, from which the confidence score is directly computed [3]. Although this procedure produces good results, it is extremely slow, as DTW has quadratic run-time, which makes this approach intractable.

To speed up the matching process, Raffel proposes learning distance preserving low-dimension embedding spaces, which should allow for highly dissimilar matches to be efficiently removed from the search space. Raffel explores two approaches, an attention-based network (\mathcal{H}_∞) that embeds arbitrary length CQT matrices into a 128-bit hash code [12], and a convolution-based network (\mathcal{H}_k) that maps $k \times 48$ CQT matrices into 32-bit hash codes [13], which can be used to transform a $n \times 48$ CQT matrix into a sequence of $\lfloor \frac{n}{k} \rfloor$ 32-bit hash codes. Using trained embedding networks \mathcal{H}_∞ and \mathcal{H}_8 , Raffel employs the following procedure to match a single MIDI CQT m against a set of audio CQTs A .

1. Blocking Stage

- (a) Compute $\mathcal{D}_H(\mathcal{H}_\infty(a), \mathcal{H}_\infty(m))$ for each $a \in A$, where \mathcal{D}_H is the bitwise hamming distance.
- (b) Construct a set A' , containing the $t_1 = 100,000$ $a \in A$ that are closest to m , using the distances calculated in 1a.
- (c) Compute $\text{DTW}(\mathcal{H}_8(a), \mathcal{H}_8(m))$ for each $a \in A'$.
- (d) Construct a set A'' containing the $t_2 = 250$ $a \in A'$ that are closest to m , using the distances calculated in 1c.

2. Matching Stage

- (a) Compute $\text{DTW}(a, m)$ for each $a \in A''$ and record any matches with more than .5 confidence.

3.1 Modifications to the Matching Procedure

According to Raffel’s measurements, it takes an average of 108 seconds on a single CPU to match one MIDI file against 1,000,000 Audio files. As a result, without making modifications to Raffel’s procedure, it would take roughly 558 days on a 32-core CPU to match our collections of audio and MIDI files. In order to optimize the audio-MIDI matching procedure to our specific context, we make changes to the blocking stage. Notably, since we do not modify the second stage, and use Raffel’s code² to com-

pute the confidence scores, our matches can be considered to be the same quality as those found in the LMD.

The simplest modification involved implementing a c++ version of the DTW code for 32-bit hash sequences, used in the blocking stage, which runs 2 times faster than Raffel’s jit-compiled Cython implementation according to our measurements. We also reconsider the use of the attention based embedding network \mathcal{H}_∞ in Steps 1a and 1b. Using Raffel’s approach, Step 1a can be computed very quickly, accounting for less than 1% of the total algorithm run-time. However, due to the low reliability of distance measurements in this embedding space, relatively few audio files can be removed from consideration. As a result, Step 1c takes much longer to run, accounting for roughly half of the total run-time. One reason for the limited accuracy of this approach, is that \mathcal{H}_∞ must embed MIDI and audio CQTs into the same 128-bit hash code, despite MIDI files being much longer than the audio files.

To address this issue, we use \mathcal{D}_w , defined in Eq. 1, to compute the distances in Step 1a. Given an $n \times 48$ MIDI CQT m and an audio CQT a , we build a set of 30-second length sub-sequences (\mathcal{X}^m) from m , as defined in Eq. 1a, where s is the stride. Using \mathcal{H}_{128} we map each 30-second length CQT matrix (i.e. 646×48) x to a hash code by splitting x into contiguous windowed sub-sequences, computing $\mathcal{H}_{128}(\cdot)$ for each sub-sequence, and concatenating the resulting hash codes. Formally, we refer to this process as \mathcal{H}_k^* , which we define in Eq. 1b, where \oplus denotes concatenation. Then, as shown in Eq. 1c, we compute the bitwise hamming distance (\mathcal{D}_H) between $\mathcal{H}_{128}^*(x)$ and $\mathcal{H}_{128}^*(a)$ for each $x \in \mathcal{X}^m$, considering the minimum distance to be representative of the distance between m and a .

$$\mathcal{X}^m = \{m[si:si+646] : 0 \leq i < \lfloor \frac{n-646+1}{s} \rfloor\} \quad (1a)$$

$$\mathcal{H}_k^*(x) = \oplus \{\mathcal{H}_k(x[ki:k(i+1)]) : 0 \leq i < \lfloor \frac{\|x\|}{k} \rfloor\} \quad (1b)$$

$$\mathcal{D}_w(a, m) = \min(\{\mathcal{D}_H(\mathcal{H}_{128}^*(a), \mathcal{H}_{128}^*(x)) : x \in \mathcal{X}^m\}) \quad (1c)$$

3.2 Training the Embedding Networks

We derive our neural network architecture from the one used by Raffel [3]. The first section of the network is comprised of k groups, with each group is containing $2 \times 3 \times 3$ convolutional layers, followed by a 2×1 max pooling layer. The second section contains two dense layers with 2048 units each, followed by a 32-dimensional output. The ReLU activation is used in all layers, except for the last layer, which uses the \tanh activation function to effectively binarize the output. For the \mathcal{H}_{128} network, which learns to downsample a sequence of 128×48 CQT matrix into a 32-bit hash code, there are $k = 5$ groups, using the filter sizes 64, 64, 64, 32, and 16 for each group respectively. For the \mathcal{H}_8 network, which learns to downsample a 8×48 CQT matrix into a 32-bit hash code, there are $k = 3$ groups, using 64, 32, and 16 filters per group respectively. We train \mathcal{H}_{128} and \mathcal{H}_8 using the same triplet loss as Raffel.

² <https://github.com/craffel/midi-dataset>

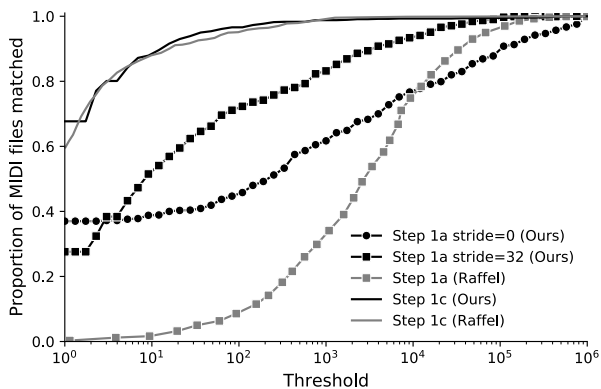


Figure 1. Percentage of MIDI files matched at thresholds.

In terms of training data, we use the 116,189 audio-MIDI matches from the LMD, which we split into testing, validation and training datasets. We train each network with a learning rate of $1e-4$, and early stopping on validation every 1000 batches, using Keras [14].

3.3 Evaluating the Embedding Networks

To evaluate the expected accuracy of distance calculations using our trained embedding networks, we use the same method proposed by Raffel. For a known audio-MIDI pair (m, a) , we measure the distance between m and a set of 1,000,000 audio files \mathcal{X} , with $a \in \mathcal{X}$, to determine the rank of the correct match. After repeating this process for 1,000 audio-MIDI pairs in our test set, we can measure the proportion of MIDI files where the correct match ranks below a particular threshold. The results are presented in Figure 1, including results previously presented by Raffel for purposes of comparison [3]. Although Raffel used different data to train and evaluate the embedding, we can be fairly confident in the reliability of our comparison, as the curve for our \mathcal{H}_8 embedding network (Step 1c (Ours)) is nearly identical to the curve for Raffel’s \mathcal{H}_8 embedding network (Step 1c (Raffel)). Although using \mathcal{D}_w slows down Step 1, the results demonstrate that it is much more accurate, which means we can reduce the number of comparisons needed in Steps 1c and 1d, which ultimately speeds up the algorithm, as Step 1c accounts for roughly half of the total run-time.

3.4 Matching Against 32,000,000 Audio Files

Clearly, a large factor contributing to the run-time of the matching algorithm is the threshold levels (t_1, t_2) for each stage of the search. Raffel et al. determine t_1 and t_2 based on the evaluation method presented in Figure 1. However, this approach is merely a proxy for what we are actually trying to accomplish. Put simply, in matching a large collection of MIDI files with a large collection of Audio Files, we are trying to maximize the number of matches. In order to get a sense of the relationship between run-time and the number of matches, we run our matching procedure with 1,000 MIDI files and 10,000,000 audio files, using various thresholds. The results in figure 2 show that we pay a high computational cost to increase the number of

MIDIs matched. For example, increasing the thresholds from $t_1 = 100,000 / t_2 = 250$ to $t_1 = 1,000,000 / t_2 = 2,500$ increases the run-time by 560%, while only yielding a 10% increase in the number of MIDIs matched and a 200% increase in the number of Audio files matched.

Due to memory limitations, it is not possible to match a MIDI CQT against all 32,000,000 audio CQTs at once. As a result, we subdivide the audio CQTs into four chunks, and process them each separately. In light of the results in the previous section, we decided to set $t_1 = 100,000$ and $t_2 = 250$ for each chunk. In Table 1, we report the results of the Audio-MIDI matching procedure. In comparison to the LMD, where only 26% of the MIDI files were matched to at least one Audio file, we were able to match 56% of the MIDI files, for a total of 237,236 MIDI files matched. Notably, our modifications to the matching procedure also had a substantial impact on the run-time, as the average run-time per match was only 3.3 times more than the run-time for LMD matching, despite matching against over 32 times more audio.

3.5 High Reliability Audio-MIDI Matches

Although the audio-MIDI matches are fairly reliable, Raffel notes that it is not uncommon for there to be false positives when an audio-MIDI pair share the same chord progression [3]. To address these issues, we produce a subset of the audio-MIDI matches which are more reliable, using artist+title metadata that was collected during the scraping process. In short, we only retain audio-MIDI matches where the title or artist scraped with the MIDI file is a fuzzy match to the metadata on Spotify. Since artists and title metadata frequently contain extraneous information, we remove all content in parenthesis or square brackets, and remove all content following a dash. As a result, the Spotify track titled "Rain Is Falling (Karaoke Version) - Originally Performed By Electric Light Orchestra" would be reduced to "Rain is Falling" after pre-processing. We measure the similarity between two strings using cosine similarity on their tri-gram profiles, and only keep matches when the similarity exceeds .8 for either the artist or the title metadata. Once this procedure has been completed, we are left with 53,496 (12%) matched MIDI files and 829,728 total matches.

4. LINKING SPOTIFY AND MUSICBRAINZ

To further expand the dataset, we make links between Spotify track ids to MusicBrainz recording ids using a classifier trained on audio features. Although AcousticBrainz Labs has provided an archive³ of the Echo Nest mappings between MusicBrainz and Spotify, we were only able to match 24,363 MIDI files to MusicBrainz IDs using this resource. To train our classifier, we gathered a set of ground truth data using International Standard Recording Codes (ISRC), which are provided by both Spotify and MusicBrainz. Although Spotify provides this information for almost all of their tracks via their API, only a percentage of recordings in the MusicBrainz database have been

³ <https://labs.acousticbrainz.org/million-song-dataset-echonest-archive/>

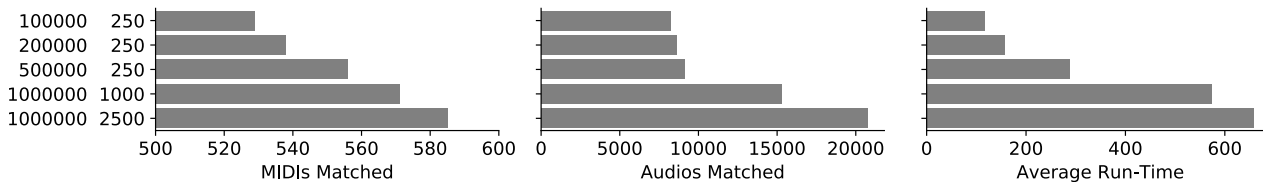


Figure 2. The number of MIDI files matched, Audio recordings matched and average match run-time for different thresholds. On the left, the first value denotes t_1 and the second value denotes t_2 .

Dataset	MIDIs	Audio Source	Matching Method	Matched MIDIs	Matched Audios	Total Matches	Percentage of MIDIs Matched
LMD	176,581	MSD [10]	Audio	45,129	31,034	116,189	25.6%
MMD	436,631	Spotify	Audio	237,236	2,209,941	10,796,557	52.7%
MMD	436,631	Spotify	Audio + Text	53,496	347,703	829,728	12.3%
MMD	436,631	MusicBrainz*	Audio	168,032	1,094,901	8,384,256	38.5%
MMD	436,631	MusicBrainz*	Audio + Text	34,174	408,922	1,232,909	7.8%

Table 1. Statistics for the audio-MIDI matching. Note that the MusicBrainz matches were computed by combining the Spotify audio-MIDI matches and the Spotify-MusicBrainz links (Section 4). The Percentage of MIDIs Matched column reports the percentage of MIDI files in the respective dataset that have at least one match to an audio file. Total Matches denotes the total number of unique audio-MIDI pairs matched.

labeled with an ISRC code. Using the ISRC codes which were available, we were able to compile about 100,000 unique ground truth matches. This data was divided into training, validation and testing sets.

We use the AcousticBrainz API⁴ to obtain features for recordings in the MusicBrainz database, since the actual audio is not provided by MusicBrainz or AcousticBrainz. To extract features from the 30-second Spotify preview clips, we use the same feature extractor as AcousticBrainz (Essentia [15]). Using the low-level features extracted via Essentia, we obtain a feature vector of dimension 1773 to represent each audio clip. Then we trained a classifier to predict whether a pair of vectors, one collected from the AcousticBrainz database, and another from Spotify, correspond to the same recording. To train the classifier, we expose the model to ground truth matches, where the AcousticBrainz recording and Spotify recording share the same ISRC, and negative matches, where both recordings do not share the same ISRC. To construct a negative match, we randomly select one recording from each data source (AcousticBrainz and Spotify). Note that for training, validation and testing we make sure the model is exposed to both conditions (ground truth and negative match) an equal number of times.

We use the XGBoost library [16] to train a gradient boosting model. To determine the optimal hyper-parameters for the model, we perform a grid search using the following parameters: nestimators {2500, 5000}, learning rate {0.1, 0.25, 0.5, 0.75}, and max depth of {2, 3, 4}. To evaluate the models, we calculate the accuracy with which the model was able to predict if the pair of recordings was a positive (ground-truth) or negative match. We found the model with nestimators=2500 learning rate=0.25 and max depth=4 to perform the best on the validation set, achieving 97.6% accuracy. To give us some indication that

we are not simply over-fitting on the validation set, we compute the accuracy of the best model using the testing set. Based on the fact that the best model scored 97.5% accuracy on the test set, which was only used once, we can be fairly confident that the model will generalize with this level of accuracy.

Since, at the time of writing, there are 5,534,103 unique recordings in the AcousticBrainz dataset, and 2,209,941 Spotify audio previews (see Table 1) which we want to match against, collecting the model’s predictions for each pairwise match would be extremely computationally expensive. To make this process feasible, we first match all the artists in the MusicBrainz database against a list of artists from Spotify using tri-gram cosine distance with a threshold of 0.7. Then we match each the titles of each recording if the artists were a match, once again using tri-gram cosine distance with a threshold of 0.7. Then for each potential match, we use the classifier to predict whether it is actually an audio match. Consequently, the error rate should be lower than 2.5% since matches must also have similar metadata (artist title) to be considered a match. The entire process took about 3 days on a single computer. In Table 2 below, we outline the results of the Spotify-MusicBrainz linking process. We provide details on the MIDI-MusicBrainz matches which were derived from the audio-MIDI matches in Table 1.

5. ANALYZING THE DATASET

5.1 Overview Statistics for the Midi Files

In order get a sense of the type of data that was collected, we compute the distributions for several features. We parse a MIDI file into a set of tracks, where a track is simply the set of note onsets and offsets belonging to a (MIDI track, channel, instrument) tuple. Each track is subdivided into a sequence of bars, using the time

⁴ <https://acousticbrainz.org/data>

	Matched Spotify IDs	Matched MusicBrainz IDs	Spotify-MusicBrainz Matches	MIDI-MusicBrainz Matches
MSD Echo Nest Archive	1,307,152	675,240	3,168,164	24,363
ISRC Matches	104,404	69,006	104,404	82,951
Ours	600,142	1,094,901	8,263,482	168,032

Table 2. Statistics for the Spotify-MusicBrainz matching.

signature information present in the MIDI file. Due to space limitations, we present a few of the most pertinent features below, providing a more comprehensive overview elsewhere⁵:

1. **Number of Tracks** : The number of tracks, as defined above, in a MIDI file.
2. **Beat Length** : The total length in quarter note beats of an entire MIDI file.
3. **Notes Per Bar** : The number of note onsets occurring in a bar. We measure this on each track separately, so that we do not conflate notes per bar and number of tracks.

We compute the distribution of each of these features across three different sets of data: the LMD, MMD, and their symmetric difference $MMD \Delta LMD$. These distributions are shown in Figure 3. On a whole, the graphs demonstrate that LMD, $MMD \Delta LMD$ and MMD are all fairly similar, however there are some differences worth noting. The two most obvious differences, are the beat length and number of tracks. The difference in beat length distributions can mainly be explained by the fact that two of the sites we scraped MIDI files from only provide 30s preview MIDI clips for free. Since the musical quality of these shorter MIDI files is comparable to that found in the LMD, we saw no real reason to exclude these files. The difference in track counts per MIDI does not have an obvious explanation, but is worth noting nonetheless.

5.2 Estimating the Reliability of Scraped Metadata

To gauge the reliability of the scraped metadata, we analyze instances where metadata was collected for the same MIDI file (md5 checksum) from multiple sources. In total, there are over 10,000 MIDI files which satisfy this criteria. For each of these MIDI files, we compare the title/artist and genre/category metadata separately. For the title/artist metadata, we concatenate this metadata into a single string, delimited by a "-", and compute cosine similarity on their tri-gram profiles. For the genre metadata, we compute tri-gram cosine similarity between each pairwise combination of elements between two genre/category lists, and report the maximum similarity. The mean similarity is 73.7% for title/artist metadata and 1.1% for genre metadata. Immediately apparent, is the significant discrepancy, as title/artist metadata appears to be fairly consistent from site to site, while genre metadata is not. Further manual analysis reveals that the genres/categories are often very generic, which may make them unsuitable for some purposes. In some respects, this is not altogether surprising,

as determining the genre/category of a piece of music is a highly subjective process, and other research has shown a significant level of disagreement [17]. However, with regards to the artist/title metadata, these results seem to indicate that we can be fairly confident in this form of metadata. It is worth noting that this type of analysis does not rule out cases where artist/title metadata on multiple sites was derived from a single inaccurate source to begin with.

5.3 False Positives and Audio Midi Matching

Using the standard and high-reliability sets of audio-MIDI matches, we can further analyze the source of false positives in the matching procedure. To do this, we compare the genre distribution of each set of audio-MIDI matches. Since Spotify uses more than 5,000 genres, many of which contain descriptors of particular locations (ex. Louisville Indie) or languages (ex. Spanish Indie Pop), we preprocess the data to remove geographical locations, demonyms and languoids. This results in about 2,500 genres. To further aggregate these genres into broader categories we employ a graph embedding approach. Using the Spotify API, we collect a list of genres for 336,507 different artists. For example, the band U2 has a genre list containing three genres: Irish Rock, Permanent Wave, and Rock. Note that after we apply our pre-processing procedure, U2 has two genres: Rock and Permanent Wave. Of particular interest for our purposes here, is artists which have a genre list containing more than one genre, as the overall frequency with which two genres co-occur within genre lists should provide a good indication of their similarity.

Then we construct a graph where each genre is a node, and the edge weights between nodes are the count of co-occurrences within the genre lists. To create the embedding, we use the Node2Vec algorithm [18], which creates an embedding space that is trained on relations found within the graph. Similar to the word2vec algorithm [19], where adjacent groupings of words inform the embedding, random walks on the graph are used to infer a context for each node. We use the nodevectors⁶ implementation of Node2Vec to learn a 32-dimensional embedding space, training with random walks of length 30 for 100 epochs. To determine a small set of k representative genres, we use Agglomerative Hierarchical Clustering with Ward linkage to partition the embedded genre vectors into k clusters. In order to give each cluster a human-readable label, we count the frequency with which each of the genres belonging to the cluster is used in the genre lists. The most frequently used genre is taken as the label for each genre. We set $k = 15$, which produces the following set of genres: indie, rock, experimental, jazz, pop, metal, musica, electronic,

⁵ <https://github.com/jeffreyjohnens/MetaMIDIdataset>

⁶ <https://github.com/VHRanger/nodevectors>

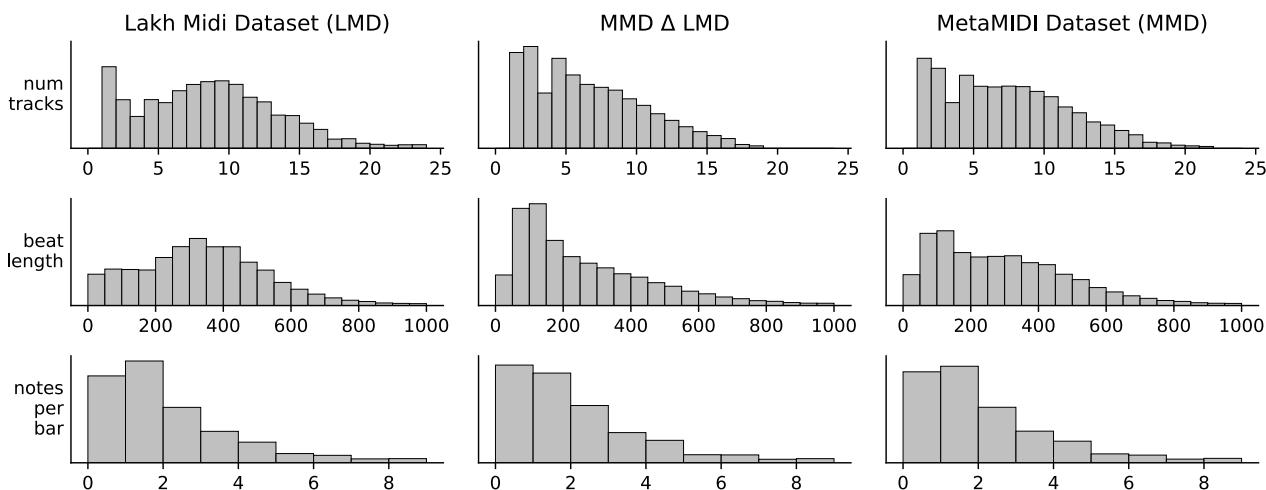


Figure 3. The distributions for various features computed on LMD, MMD Δ LMD and MMD.

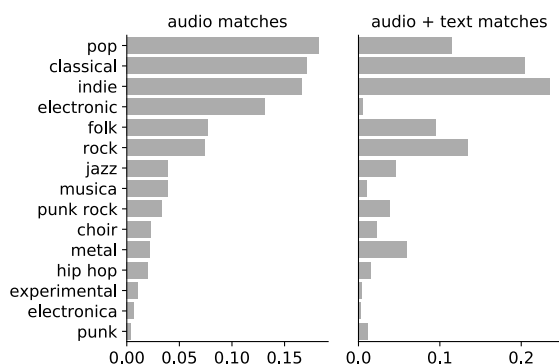


Figure 4. The distribution of genres for matched MIDI files using two methods: audio and audio + text.

folk, choir, classical, punk, punk rock, hip-hop, and electronica. We admit that our decision to set $k = 15$ is fairly arbitrary, however, due to the nature of our clustering procedure, selecting a different value for k would not have a large impact. For example, setting $k = 16$ produces the same set of 15 genres with one new genre cluster.

In Figure 4 the genre distributions are plotted for each version of the matching procedure. Since we can be fairly confident that the audio + text matches are more accurate, analyzing the discrepancies between the genre distributions can help identify some of the shortcomings of the DTW audio-MIDI match algorithm. In the audio matches distribution, we see a large increase in pieces classified as pop and electronic, which indicates these pieces are likely the source of most of the error. This may be a byproduct of their simple harmonic structure, and/or the prevalence of remixes and covers within these particular genres.

6. USING THE METAMIDI DATASET

The dataset, as well as a detailed description of its contents, can be accessed through the MetaMIDI Dataset reposi-

tory⁷. Throughout the dataset, MIDI files are identified by their md5 checksum. We provide mappings from md5 checksums to Spotify track ids and MusicBrainz recording ids, which can be used to access a plethora of metadata. The MusicBrainz database provides access to variety of linked entities including artists, recordings, releases, composers, producers, recording engineers and labels. Detailed attributes are available for most entities. For example, the MusicBrainz entry for the group Bon Iver, provides the date and location where the group was established, a list of aliases, a set of genre tags, and a comprehensive list of links to external websites. Using the Spotify API, a variety of metadata can be accessed, including track-based audio features such as danceability, valence, liveness and energy; and additional metadata ranging from genre to artist popularity.

7. CONCLUSION

Although the primary contribution is the dataset itself, we have also provided reusable insights related to the audio-MIDI matching algorithm and the Spotify-MusicBrainz linking procedure. One limitation worth noting, is the uncertainty in relying on Spotify’s 30-second clips to persist into the future, which unfortunately has already become an issue with the 7Digital clips in the Million Song Dataset [10]. With regards to the dataset, we anticipate a wide variety of potential use-cases for this data. Since many generative systems have been trained using the Lakh MIDI Dataset [4, 5, 9], the MMD will undoubtedly be a valuable asset to research in this area, as it features 2.4 times more MIDI files. More broadly, the metadata that our audio-MIDI matches provide access to, as well as the audio-MIDI matches themselves, can be used to support a variety of scientific inquiries related to MIR and Musicology.

⁷ <https://github.com/jeffreyjohnens/MetaMIDIataset>

8. ACKNOWLEDGMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Helmut & Hugo Eppich Family Graduate Scholarship.

9. REFERENCES

- [1] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” *arXiv preprint arXiv:1810.12247*, 2018.
- [2] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, “ASAP: a dataset of aligned scores and performances for piano transcription,” in *Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020.
- [3] C. Raffel, “Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching,” Ph.D. dissertation, Columbia University, 2016.
- [4] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *Proc. of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 685–692.
- [5] A. Roberts, J. H. Engel, C. Raffel, C. Hawthorne, and D. Eck, “A hierarchical latent vector model for learning long-term structure in music,” in *Proc. of the 35th International Conference on Machine Learning*, 2018, pp. 4361–4370.
- [6] H. Schreiber and M. Müller, “A single-step approach to musical tempo estimation using a convolutional neural network,” in *Proc. of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 98–105.
- [7] A. Ferraro and K. Lemström, “On large-scale genre classification in symbolically encoded music by automatic identification of repeating patterns,” in *Proc. of the 5th International Conference on Digital Libraries for Musicology*, 2018, pp. 34–37.
- [8] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, “Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2019, pp. 45–49.
- [9] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Proc. of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 34–41.
- [10] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *Proc. of the 12th International Society for Music Information Retrieval Conference*, 2011.
- [11] J. C. Brown, “Calculation of a constant q spectral transform,” *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.
- [12] C. Raffel and D. P. Ellis, “Pruning subsequence search with attention-based embedding,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2016, pp. 554–558.
- [13] —, “Large-scale content-based matching of midi and audio files,” in *Proc. of the 16th International Society for Music Information Retrieval Conference*, 2015, pp. 234–240.
- [14] F. Chollet *et al.*, “Keras,” <https://keras.io>, 2015.
- [15] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepas, J. Salamon, J. R. Zapata González, X. Serra *et al.*, “Essentia: An audio analysis library for music information retrieval,” in *Proc. of the 14th International Society for Music Information Retrieval Conference*, 2013.
- [16] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.
- [17] R. Brisson and R. Bianchi, “On the relevance of music genre-based analysis in research on musical tastes,” *Psychology of Music*, vol. 48, no. 6, pp. 777–794, 2020.
- [18] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.
- [19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

MODELING AND INFERRING PROTO-VOICE STRUCTURE IN FREE POLYPHONY

Christoph Finkensiep

École Polytechnique Fédérale de Lausanne
christoph.finkensiep@epfl.ch

Martin Rohrmeier

École Polytechnique Fédérale de Lausanne
martin.rohrmeier@epfl.ch

ABSTRACT

Voice leading is considered to play an important role in the structure of Western tonal music. However, the explicit voice assignment of a piece (if present at all) generally does not reflect all phenomena related to voice leading. Instead, voice-leading phenomena can occur in free textures (e.g., in most keyboard music), or cut across the explicitly notated voices (e.g., through *implicit polyphony* within a single voice). This paper presents a model of *proto-voices*, voice-like structures that encode sequential and vertical relations between notes without the need to assume explicit voices. Proto-voices are constructed by recursive combination of primitive structural operations, such as insertion of neighbor or passing notes, or horizontalization of simultaneous notes. Together, these operations give rise to a grammar-like hierarchical system that can be used to infer the structural fabric of a piece using a chart parsing algorithm. Such a model can serve as a foundation for defining higher-level latent entities (such as harmonies or voice-leading schemata), explicitly linking them to their realizations on the musical surface.

1. INTRODUCTION

A basic observation about tonal structure in music is that notes tend to form vertical and horizontal relations, which are generally not explicit in representations of the musical surface such as a score or a recording. An example of these relations can be seen in Figure 1. The initial line of sixteenth notes in the right hand, for example, forms an arpeggiation of a D-minor chord. A reduction or simplification of the piece might realize this chord as a single vertical entity, but the vertical relation between the notes D5, A4, F4, and D4 is not directly encoded in the score. Similarly, the two A4s of this arpeggiated chord are part of a line that first moves to the neighbor note Bb4 before returning to A4 on the fourth beat of the first bar. Again, this connection is not explicitly represented in the score, much less so in a recording.

Sequential relations between notes are sometimes equated with *voices* [1] that are either explicitly given (e.g.

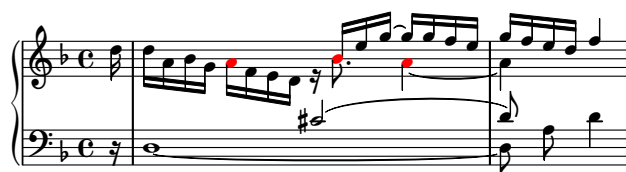


Figure 1: An example of free polyphony in J. S. Bach’s Allemande BWV 812 I. Sequential structures (such as the A-Bb-A motion across the first measure) are generally not explicit in the score.

in monophonic melodies or strict polyphony), or inferred through voice separation [2, 3, 4, 5, 6, 7, 8, 9]. However, sequential relations do not always coincide with voices: A single voice can exhibit *implicit polyphony* [10, p. 367][11] (also called implied or latent polyphony), i.e., consist itself of several implied sub-voices. For example, in the upper voice in Figure 1, the notes of the D-minor chord belong to separate voices on a more abstract level. Similarly, sequential connections can go across different voices, such as the A4 moving to Bb4 while the notated voice continues to C#4.

Implicit (or more generally free) polyphony is commonly understood as forming a set of parallel and independent *auditory streams* [12, 13, 1] that are inferred from the musical surface by connecting notes into sequences. The present paper, in contrast, proposes a model of free polyphony that departs from this view in several respects: First, free polyphony is understood as a network of lines that can be connected to each other rather than a set of independent streams. Second, this network is not defined through inference from the surface, but rather explicitly constructed in a generative process that creates the network in successive steps. Inferring this network from a piece is then based on inverting this process, i.e., *parsing* the piece. Third, connections between notes are not based on continuing a stream, but instead follow from *elaboration* of existing structures through fundamental and musically interpretable operations, adopting a top-down view instead of a left-to-right view on voice-leading structure [14, 15]. We name the resulting lines in the network *proto-voices*, since – like voices – they connect notes to sequential lines but cannot be themselves implicitly polyphonic. This paper presents a formal definition of the proto-voice model as a recursive process, and describes a parsing algorithm that can infer the proto-voice structure from a score.



This model does not yet account for other musical aspects such as rhythm and meter, harmony, form, or motivic and thematic material. However, it is intended to further the understanding of polyphonic structure on formal grounds, and could potentially serve as a module in a more complete system for musical analysis.

The idea of modeling free polyphony as a recursively generated network of lines is central to Schenkerian analysis [16]. However, the constructions in Schenkerian analysis are specific to Western Common Practice music and more high-level than the generic operations that give rise to proto-voices. Thus, the proto-voice model can be understood as a formal foundation for describing richer concepts of musical structure (such as the ones appearing in Schenkerian analysis or other analytical frameworks), and it is applicable to a wider range of musical styles that make use of implicit or free polyphony (such as Jazz or melodies in Pop/Rock).¹ However, because of these similar ideas, the model presented here is related to models that formalize sub-systems of Schenkerian analysis [18, 19, 20, 21, 22, 23, 24, 25, 26, 27], and to grammatical models of musical structure in general [14, 15, 28, 29, 17, 30, 31, 32, 33]. While proto-voices inherit some of their concepts, most notably the interval-replacement method developed in [25], modeling the structure of free polyphony has yet been an unsolved problem.

2. THE PROTO-VOICE MODEL

2.1 Constructing Proto-Voices

At the core of the model proposed in this paper are a number of operations that establish primitive and strictly stepwise horizontal relations between notes. These relations include *repetitions*, stepwise ornaments to a note (*neighbor notes*), and notes that fill larger intervals stepwise (*passing notes*). While the notion of a step generally depends on what is considered a step in the respective style, we consider a step to be a diatonic second for the purpose of modeling tonal music in the diatonic tradition.

All of these operations relate notes to one or two reference notes, or *parents*. Following Yust [25], operations with two parents are represented by *edge replacement*: If the two parent notes p_1 and p_2 are connected by an edge $p_1 \rightarrow p_2$, then this edge can be replaced by a child note together with two new edges to the parents: $p_1 \rightarrow c \rightarrow p_2$.

Formally, proto-voices are represented as a graph that contains one vertex per note, one vertex each for the beginning (\bowtie) and the end (\bowtie) of the piece, and two types of edges: *Regular edges* indicate a sequential connection between two notes (or \bowtie/\bowtie) that may be used for elaboration by introducing a repetition or a neighbor of either parent note (or of both if the parents have the same pitch). The interval along a regular edge is always within the range of a step (unless one of its vertices is \bowtie or \bowtie), and this property

¹ The principle of recursive ornamentation is also used in non-Western styles, such as Indian classical music [17], the model presented here is specifically inspired by Western tonal music. However, some of the formal techniques presented here might also be useful for expressing structural relations specific to other styles.

is maintained through the elaboration operations. *Passing edges* indicate connections between two notes with an interval that is larger than a step (introducing a new, subordinate proto-voice). They must be filled with passing notes from either end until only stepwise connections remain.

The generation of a piece starts with the empty piece $\bowtie \rightarrow \bowtie$ and recursively applies one of several elaborations rules. *Single-sided* rules pick a note and insert either a repetition or a neighbor note to its left or right:

$$x \implies x' \rightarrow x \quad \text{repeat-before} \quad (1)$$

$$x \implies x \rightarrow x' \quad \text{repeat-after} \quad (2)$$

$$x \implies n \rightarrow x \quad \text{left-neighbor} \quad (3)$$

$$x \implies x \rightarrow n \quad \text{right-neighbor} \quad (4)$$

Double-sided rules pick an edge and insert along it one new note and two new edges:

$$\bowtie \rightarrow \bowtie \implies \bowtie \rightarrow x \rightarrow \bowtie \quad \text{root-note} \quad (5)$$

$$x_1 \rightarrow x_2 \implies x_1 \rightarrow x' \rightarrow x_2 \quad \text{full-repeat} \quad (6)$$

$$x \rightarrow y \implies x \rightarrow y' \rightarrow y \quad \text{repeat-before}' \quad (7)$$

$$x \rightarrow y \implies x \rightarrow x' \rightarrow y \quad \text{repeat-after}' \quad (8)$$

$$x_1 \rightarrow x_2 \implies x_1 \rightarrow n \rightarrow x_2 \quad \text{full-neighbor} \quad (9)$$

Passing rules, finally, fill passing edges with passing notes from either end until the progression is fully stepwise:

$$x \dashrightarrow y \implies x \rightarrow p \dashrightarrow y \quad \text{passing-left} \quad (10)$$

$$x \dashrightarrow y \implies x \dashrightarrow p \rightarrow y \quad \text{passing-right} \quad (11)$$

$$x \dashrightarrow y \implies x \rightarrow p \rightarrow y \quad \text{passing-final} \quad (12)$$

In these rules, matching letters indicate matching pitches, indices disambiguate parent notes with the same pitch, and apostrophes mark inserted repetitions of parent notes. Neighbor notes n must be a step away from their parents, (disregarding their octaves to allow for octave displacement). Similarly, passing notes p must be a step from the parent(s) they are directly connected to and lie within the interval spanned by both parents. Note that none of these rules produce passing edges, which establish new connections between previously unconnected lines and thus require some additional structure (see Section 2.2. An example proto-voice derivation of the previous example (Figure 1) is shown in Figure 2.

2.2 Temporal Organization

While proto-voices model the sequential organization of notes, they do not specify when notes are simultaneous. On the musical surface, simultaneity of notes is implied by their onsets and durations. However, notes that are temporally displaced on the surface can often be regarded as forming a vertical sonority on a higher level of abstraction, such as the arpeggiated d-minor chord in the beginning of Figure 1. In order to express these latent vertical configurations, simultaneity is modeled through *slices*, segments of a piece in which the same notes sounds. A piece (or a reduction of a piece) is then represented as a sequence

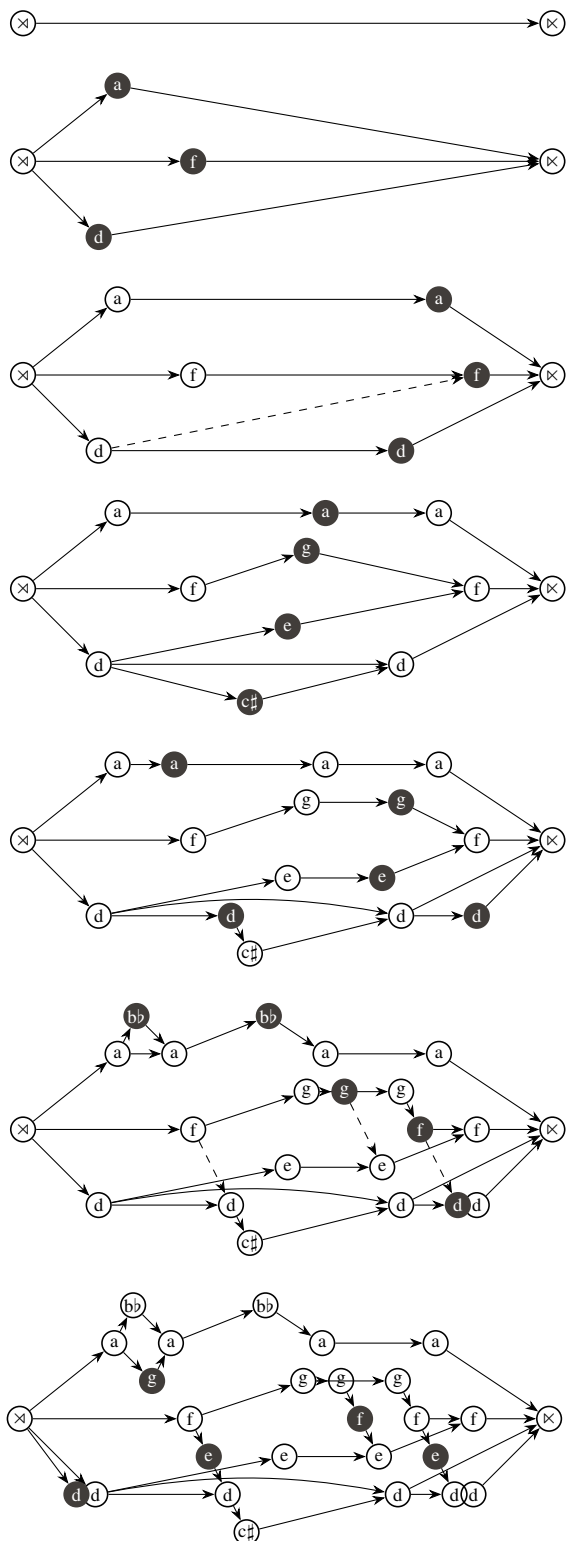


Figure 2: A proto-voice derivation of the notes in Figure 1. The position of a note is chosen to indicate its pitch and onset in the piece. Later derivation steps hide some edges from earlier steps in the interest of readability. Note that each note is shown exactly once here, unlike in the final model, which represents each note once per slice it occurs in. Furthermore, pitches in different octaves have been merged to simplify the graph.

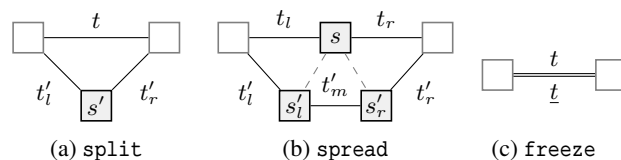


Figure 3: The three operations on outer structure. The slices and transitions to be elaborated are shown at the top while the lower part shows the generated structure.

of slices. Notes that are simultaneous with several non-simultaneous notes (such as the bass note D in Figure 1) are split among the corresponding slices but remain connected by edges, thus ensuring that a surface note is generated through a single generation process.

Proto-voices are integrated into the slice structure by attaching their edges to the *transitions* between two slices. Note that transitions can only contain edges that connect notes in the slices adjacent to the transition. Long-distance edges are thus represented in latent transitions, i.e. transitions in a reduction of the piece. As a consequence, edges “vanish” in a well-defined manner during the generation process, namely whenever a transition is replaced through one of the generative operations. Since slices and transitions contain notes and edges, respectively, we call the slices and transitions *outer structure*, and the notes and edges *inner structure*.

Formally, a slice s is defined as a multiset (or bag) of pitches. A transition $t = (s_l, e, s_r)$ relates two slices s_l and s_r and a configuration of edges $e = (e_{\text{reg}}, e_{\text{pass}})$, which in turn consists of a set of regular edges e_{reg} (which must be used at least once by a subsequent operation) and a multiset of passing edges (which must be used exactly once).²

Outer structure is transformed by three operations: A *split* (Figure 3a) is a rule of the form

$$t \longrightarrow t'_l \ s' \ t'_r \quad (13)$$

that replaces a transition t by inserting a new slice s' and two new transitions t'_l and t'_r . During this operation, each edge in the transition and each note in an adjacent slice can be elaborated by one or more inner operations. The resulting edges can either be discarded, or kept to form the new edges of t'_l and t'_r . As a result, each transition only contains edges that will be used subsequently.

A horizontalization, or *spread* (Figure 3b) has the form

$$t_l \ s \ t_r \longrightarrow t'_l \ s'_l \ t'_m \ s'_r \ t'_r, \quad (14)$$

and replaces a slice s by distributing its notes to two child slices s'_l and s'_r . This way, a latent vertical configuration of notes can be sequentialized. In order to simplify parsing, a restriction is made on this distribution: At least one side must inherit all instances of a specific pitch, while the other may inherit fewer instances, i.e.,

$$p^k \in s \implies p^k \in s'_l \quad p^{k-m} \in s'_r \quad \text{or} \quad (15)$$

$$p^k \in s \implies p^{k-m} \in s'_l \quad p^k \in s'_r, \quad (16)$$

² Passing edges are treated differently to avoid filling a single passing edge several times.

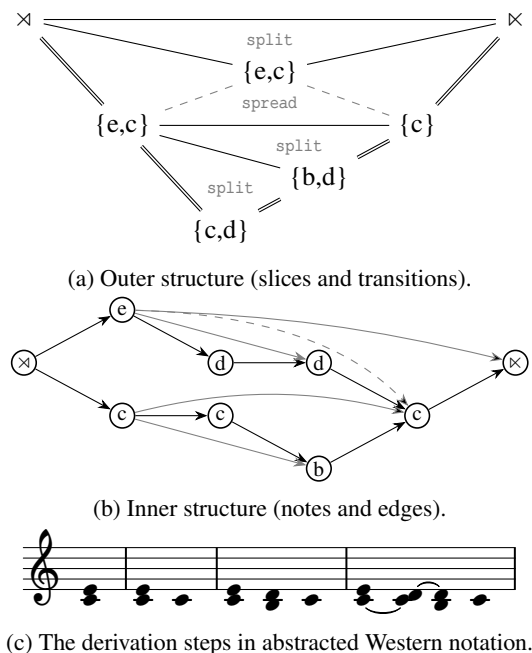


Figure 4: An example derivation of a short cadential phrase. In each *split* operation, the edges of the elaborated transition (grey in (b)) are replaced using inner elaboration operations. The passing edge from *e* to *c* is introduced during the *spread* of the top-level $\{e, c\}$ slice.

where k denotes the number of occurrences of pitch p in s , and $0 \leq m \leq k$. This way, the s can always be inferred deterministically from s'_l and s'_r by taking for each pitch the maximum number of occurrences in s'_l or s'_r .

In the process of a *spread*, passing edges may be introduced between arbitrary pairs of notes, and regular edges may be introduced between notes with the same pitch. This way, the introduction of passing edges becomes a local operation that is guaranteed to respect the temporal order of notes. Since all edges in a transition must be used, a *spread* is only allowed when no edges from the parent transitions t_l and t_r are lost by moving notes to the opposite side. While this operation does not change the contents of t_l and t_r , it replaces s with s'_l and s'_r respectively, which makes this operation context-sensitive.

Finally, a *freeze* (Figure 3c) marks a transition as terminal, stopping the generation process for this transition:

$$t \longrightarrow \underline{t}. \tag{17}$$

It is only allowed when the transition contains only repetition edges, which are turned into ties, creating notes that span several surface slices.

An example derivation using these three operations can be seen in Figure 4. We use a notation similar to the maximal outerplanar graphs (MOPs) introduced in [25], with the root transition on top, surface of the piece on the bottom, and rule applications indicated by polygons. However, since the derivations here contain latent slices that do not occur on the surface, these derivation graphs are not outerplanar.

3. A PARSING ALGORITHM FOR PROTO-VOICES

3.1 Representing Derivations

The parsing algorithm for the proto-voice model produces a set of possible derivations of the input score. Such a derivation can be represented as a list of rule applications in *leftmost derivation* order. This representation is known from context-free grammars: the result of the derivation is obtained by applying each rule in the list to the leftmost non-terminal symbol of the current sequence. This is possible because the derivation below each non-terminal of a string is independent from the derivations below all other non-terminals of the string. In the proto-voice grammar, this independence property does not hold, because the context-sensitive operation *spread* can link two otherwise independent transitions (and all their ancestors). However, the idea of a leftmost derivation can still be applied here.

The maximal left-hand side of a single rule consists of two transitions. Thus, instead of the leftmost non-terminal, we consider the two leftmost non-terminal transitions as the context for each rule application. Freezing the left of the two transitions moves the context to the right. A *spread* consumes both transitions of the context and pushes its children onto the list of open transitions. In order to allow the right parent of a *spread* to be the result of a *split*, *splits* can be applied to either the left or the right transition of the current context. However, in order to disambiguate the derivation order, we restrict right *splits* to always happen *after* left *splits* or *freezes*. If only a single transition is left, then only a *split* or *freeze* can be performed. Thus, the derivation shown in Figure 4a can be unambiguously described as the leftmost derivation *split, spread, freeze, split-left, split-left, freeze, freeze, freeze, freeze*.

Under these restrictions, certain configurations are not possible. In particular, the right parent transition of a *spread* cannot be the left child transition of another *spread*. However, this *outer* configuration is equivalent – with respect to the resulting *inner* structure – to another configuration where the two *spreads* are applied in reverse order. Thus, the generative power of the grammar (with respect to proto-voice structure) is not restricted by excluding this non-leftmost configuration.

A similar observation above can be made between *splits* and *spreads*: Whenever a *split* is made *after* a *spread* (i.e. on its left or right child transition), it could as well have been made before the *spread* (generating its left or right parent transition, respectively), generating the same inner structure. Therefore, we can add another restriction on the derivation order that forbids *splits* to be applied to the left or right child transitions of a *spread*, further removing the redundancy between (internally) equivalent derivations.

In a similar fashion, it is possible to reduce the number of derivations further by eliminating redundancy in the internal structure. For example, slices that are exact repetitions of one of their neighbors can be generated in two

ways, either by a `split` that only uses `repeat-*` operations on one side, or by a `spread` that produces identical child slices. Since the latter is required for passing edges, the former case might be excluded as redundant. Similarly, the repeated horizontalization of a vertical configuration can generate the same surface configuration in many different ways, which can be prevented by restricting spreads to be strictly left- or right-branching (unless intercepted by a `split`). Both of these restrictions, however, exclude some derivations with slightly different semantics than their permitted counterparts, so it depends on the use cases whether such restrictions are appropriate.³

3.2 Parsing

Previous models of hierarchical tonal structure have relied on two approaches to structural inference: Grammar-based models use variants of classical parsing techniques such as chart parsing [28, 27] while MOP-based models work with triangulations of polygons [25, 21]. The proto-voice model can be parsed using a bottom-up chart parsing algorithm that is adapted to account for the context-sensitive spread operation. A *transition chart* stores all potential latent transitions, similar to the non-terminal chart in a context-free parser. In addition, a *verticalization chart* stores items that represent the “core” of a spread, i.e. the parent slice and the middle transition (including the two child slices). This core is then combined independently with the left and right child transitions, disentangling the two reductions and reducing the combinatorial complexity.⁴

The items in the transition chart are tuples (t, σ, I_l, I_r) , consisting of a transition t , a score σ , and two IDs I_l and I_r that express combination restrictions on the left and the right of the transition, respectively. By default I_l and I_r have a default value `*` which indicates that they can combine with other transitions with the default value. The left and the right parent transitions of a `spread`, however, depend on each other through a common child (the `spread` operation itself). They are therefore marked with a special ID on their adjacent sides and can only combine with other transitions with a compatible ID. IDs are based on the *left side* of the verticalization, i.e. its left child slice and its parent slice. The details of the `spread` operation as well as the middle and right child transitions are stored in the item of the right parent transition, while the left parent transition only keeps a reference to the left child transition. This way, combining any pair of compatible left and right parent transitions restores a complete and valid `spread` operation with all its children. While this “trick” reduces complexity by exploiting some properties specific to the proto-voice grammar, it is not known whether it reduces the overall complexity of the parser from exponential to polynomial in the number of input slices.

³ For example, with a strictly right-branching model, the expansion of the D-minor chord in Figure 1 must happen from left to right. If it is desired to split the chord first into quarter-note slices and then into eighth-note slices (to respect the metric structure), strict right-branching does not work.

⁴ For a given verticalization, instead of considering each pair of left and right transitions ($|L| \cdot |R|$ operations), the left and right transitions can be processed independently ($|L| + |R|$ operations).

The score σ of a transition represents the set of leftmost derivations from the transition to the surface it covers. It is computed bottom-up by combining the scores of the transition’s children. When two transitions are combined, their scores are combined by concatenating each alternative on the left with each alternative on the right.⁵ When parsing a `split` operation, this result is prepended with the `split` itself, which yields the score of the parent transition. The score representing a `spread` operation, however, must be distributed across the two parent transitions. This follows the same scheme as described above: the left parent keeps the score of the left child L ; the right parent takes the scores of the right child R , the the middle child M , and the rule application the `spread` itself h . However, since the correct leftmost sequence of operations should apply the scores in the order $hLMR$ the scores of the parent edges are *partial*, and the parser ensures that these fragmented derivations are handled in a way that always restores the correct sequence of derivation steps when recombined.⁶

Algorithm 1 The steps of the parsing algorithm.

```

V ← {}
T ← unfreeze each input transition
for n from 2 to |input| - 1 do
    V ←∪ verticalizations of all Tn
    T ←∪ left vert. of all Tn ⊗ V≤n and T<n ⊗ Vn
    T ←∪ right vert. of all Vn ⊗ T≤n and V<n ⊗ Tn
    T ←∪ merges of all Tn ⊗ T≤n and T<n ⊗ Tn
return T∞→∞
    
```

The parser fills the chart bottom-up using the algorithm shown in Algorithm 1. Here, *merge* refers to the inverse of a `split`, *left* and *right verticalization* refer to combining a left or right child with a verticalization item, respectively. T_n and V_n refer to the sets of chart items with a surface coverage of n slices, and \otimes creates the pairs of those items that are adjacent (i.e. their connecting slices match with respect to position and content) and have compatible IDs.

The inner structure of each operation is parsed by inverting the operation, computing all possible inputs. For `spread` and `freeze`, this is trivial since their parent elements are unique, if they exist. For `split`, all possible parent transitions are computed that generate every note in s' using all mandatory edges in t'_l and t'_r (and possibly other edges that have been dropped and thus not included in t'_l and t'_r).

A reference implementation of the parser written in Haskell is provided.⁷

⁵ In the parser, this operation is represented symbolically, which is more efficient than actually computing all combinations of alternatives.

⁶ In particular, since fragmented derivation sets are not always recombined right away, they need to combine with other operations such as `splits` and other `spreads`. The formal details of this are beyond the scope of this paper, but they are documented in the parser implementation.

⁷ <https://github.com/DCMLab/protovoices-haskell/tree/ismir2021>

4. DISCUSSION AND CONCLUSION

The proto-voice model is flexible enough to express highly complex configurations of free polyphony. However, this generative power comes at the cost of being highly ambiguous. The suspension sequence in Figure 4, for example, has 131 valid derivations, while the first half measure (including the upbeat) of the Bach example (Figure 1) already has 119,940 derivations. While this flexibility of the model allows analysts to express very subtle interpretative nuances, it also generates the problem that a single piece or excerpt has too many derivations to reasonably compare, and that any non-trivial piece takes far too long to parse exhaustively in practice. The first problem can be solved by introducing a probabilistic variant of the model that weights derivations according to their probability [32, 28]. The second problem might be resolved by a heuristic parser that does not guarantee globally optimal solutions.

There are structural configurations assumed in some theories that require an even higher flexibility than what is provided by the proto-voice model. For example, Schenkerian theory allows the *unfolding* (i.e. horizontalization) of entire progressions (such as the parallel thirds 3-2 and 1-7 in Figure 4) into a single sequence (such as 1-7-3-2(-1)). Such an operation would either require the progression to be represented as a single entity (to which the operation could be applied), or the ability to apply operations to non-entity contexts (similar to how *spread* is applied to two transitions and a slice).

The inner structure and operations of proto-voices are similar to those of MOP-based approaches [25, 21, 24] for monophonic and homophonic sequences. From these, the model inherits the ability to represent double parents and, by extension, lines of notes with a start and a goal. However, proto-voices use these ideas to solve the much more complex problem of free polyphony. The key insight that makes this extension possible is the separation of adjacency on the surface and adjacency in a line of notes, and the explicit representation of line adjacency in the proto-voice graph. In monophonic sequences, surface and line adjacency seem to be the same, but even this assumption does not generally hold: As the example of implied polyphony shows, even monophonic voices can (and generally do) have a polyphonic latent structure. Put bluntly, there is no such thing as a monophonic melody.

The outer structure (and its integration of inner operations) is similar to an approach presented by Marsden [27], that parses single-sided Schenkerian operations based on a grammar on slices. In particular, Marsden’s grammar uses *context notes* to model conditions of two-sided operations, which makes the grammar context-sensitive in a very similar way as proto-voices.⁸ While Marsden’s model does not rely on explicit voices – and thus in principle can parse inputs in free polyphony – it also does not generate voice-like structure among the notes, but rather individual bi-

nary dependency relations. A similar point can be made for models working on piano-roll representations such as many neural network approaches [34, 35, 36]: While they can work with freely polyphonic inputs, they generally do not explicitly establish polyphonic *structure* among the notes in the score.

There is, however, a deeper, more philosophical difference between the proto-voice model and the other approaches based on Schenkerian analysis: The proto-voices attempt to isolate and formalize the *structural principles* and *primitives* that give rise to free polyphony, instead of encoding the higher-level concepts and operations of a particular analytical framework. The two structural principles here are *elaboration* and *recursion*, where the former consists of the application of primitives and the latter just arises from the fact that elaboration can be applied to the output of a previous elaboration of the same kind. The structural primitives boil down to essentially two operations: stepwise insertion of notes (in all its variants) and horizontalization of simultaneous elements, which operate with the two basic relations on simultaneity and sequentiality in complementary ways.

These operations are *primitive* for two reasons: First, they provide what can be considered the lowest level of musically meaningful relations. Even more basic representations of music (such as audio or piano-roll representations) do not express musical relations (except incidental simultaneity) explicitly. Second, the basic entities and relations can be combined to express higher-level entities and relations from more specific analytical frameworks, such as different forms of harmonic analysis, Schenkerian analysis, or schema theory. A simple example of such a high-level concept can be seen in Figure 4, which constructs the voice-leading pattern of a 2-3 *suspension* in a principled way: first, a progression is generated that moves two voices down in parallel thirds, then another time interval is inserted in which the upper voice moves while the lower voice remains, creating the dissonant second. Similarly, the derivation in Figure 2 explicitly constructs an *initial ascent* [37] from D to F and the harmonic progression $I - V^7 - I$, and describes their relation to the musical surface. The preparatory function of the dominant chord and its dependency on the tonic [15] are even reflected by its notes, which are all ornaments of the following tonic harmony.

The structural principles and primitives postulated by this model are certainly not exhaustive. For one, they do not account for musical parameters such as harmony and key, timbre, or rhythm and meter. Furthermore, there might be additional structural primitives that establish other relations between objects than stepwise motion and simultaneity. Finally, there might be other relevant structural principles, such as abstraction of particular configurations into patterns, or the repetition of complex structures or patterns. However, since principles and primitives are generally orthogonal, the current model can be considered as a module of a more comprehensive model of musical structure.

⁸ In [27], this context-sensitiveness is handled by parsing with a context-free parser and then removing inconsistent derivations, while the proto-voice parser only constructs consistent derivations, but this is just an implementation detail.

5. ACKNOWLEDGEMENTS

We thank all members of the Digital and Cognitive Musicology Lab at EPFL (in particular Petter Ericson and Daniel Harasim) as well as the anonymous reviewers for their valuable feedback, and Claude Latour for generously supporting this research. This project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme under grant agreement No 760081 – PMSB.

6. REFERENCES

- [1] D. Huron. *Voice Leading: The Science Behind a Musical Art*. MIT Press, Sept. 2, 2016. 273 pp.
- [2] E. Chew and X. Wu. “Separating Voices in Polyphonic Music: A Contig Mapping Approach”. In: *Computer Music Modeling and Retrieval*. Ed. by U. K. Wiil. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 1–20. DOI: 10.1007/978-3-540-31807-1_1.
- [3] R. de Valk and T. Weyde. “Deep Neural Networks with Voice Entry Estimation Heuristics for Voice Separation in Symbolic Music Representations”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference*. 19th International Society for Music Information Retrieval Conference (ISMIR 2018). Paris, France, May 25, 2018.
- [4] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé. “Comparing Voice and Stream Segmentation Algorithms”. In: *Proceedings of the 16th ISMIR Conference*. International Society for Music Information Retrieval Conference (ISMIR 2015). Malaga, Spain, Oct. 2015, pp. 493–499.
- [5] J. Kilian and H. H. Hoos. “Voice Separation—A Local Optimization Approach.” In: International Conference on Music Information Retrieval. 2002.
- [6] P. B. Kirlin and P. E. Utgoff. “VOISE: Learning to Segregate Voices in Explicit and Implicit Polyphony.” In: *Proceedings of the Sixth International Conference on Music Information Retrieval*. ISMIR. London, 2005, pp. 552–557.
- [7] D. Makris, I. Karydis, and E. Cambouropoulos. “VISA3: Refining the Voice Integration/Segregation Algorithm”. In: *Proceedings of the Sound and Music Computing Conference 2016*. Hamburg, Germany, 2016.
- [8] A. McLeod and M. Steedman. “HMM-Based Voice Separation of MIDI Performance”. In: *Journal of New Music Research* 45.1 (Jan. 2, 2016), pp. 17–26. DOI: 10.1080/09298215.2015.1136650.
- [9] D. Temperley. “A Unified Probabilistic Model for Polyphonic Music Analysis”. In: *Journal of New Music Research* 38.1 (Mar. 1, 2009), pp. 3–18. DOI: 10.1080/09298210902928495.
- [10] E. Aldwell and A. Cadwallader. *Harmony and Voice Leading*. Cengage Learning, 2018. 722 pp.
- [11] E. Cambouropoulos. “‘Voice’ Separation: Theoretical, Perceptual and Computational Perspectives”. In: Int. Conf. on Music Perception and Cognition (ICMPC). 2006, p. 12.
- [12] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT press, 1994.
- [13] E. Cambouropoulos. “Voice And Stream: Perceptual And Computational Modeling Of Voice Separation”. In: *Music Perception* 26.1 (Sept. 1, 2008), pp. 75–94. DOI: 10.1525/mp.2008.26.1.75.
- [14] F. Lerdahl and R. S. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1985.
- [15] M. Rohrmeier. “Towards a Generative Syntax of Tonal Harmony”. In: *Journal of Mathematics and Music* 5.1 (Mar. 1, 2011), pp. 35–53. DOI: 10.1080/17459737.2011.573676.
- [16] H. Schenker. *Free Composition:(Der Freie Satz): Heinrich Schenker; Translated and Edited by Ernst Oster*. Longman, 1979.
- [17] C. Finkensiep, R. Widdess, and M. Rohrmeier. “Modelling the Syntax of North Indian Melodies with a Generalized Graph Grammar”. In: *Proceedings of the 20th International Society for Music Information Retrieval Conference* (Delft, The Netherlands). Delft, The Netherlands: ISMIR, Nov. 4, 2019, pp. 462–469. DOI: 10.5281/zenodo.3527844.
- [18] R. E. Frankel, S. J. Rosenschein, and S. W. Smoliar. “Schenker’s Theory of Tonal Music—Its Explication through Computational Processes”. In: *International Journal of Man-Machine Studies* 10.2 (Mar. 1, 1978), pp. 121–138. DOI: 10.1016/S0020-7373(78)80008-X.
- [19] S. W. Smoliar. “A Computer Aid for Schenkerian Analysis”. In: *Proceedings of the 1979 Annual Conference*. ACM ’79. New York, NY, USA: ACM, 1979, pp. 110–115. DOI: 10.1145/800177.810043.
- [20] J. Rahn. “Logic, Set Theory, Music Theory”. In: *College Music Symposium* 19.1 (1979), pp. 114–127.
- [21] P. B. Kirlin and P. E. Utgoff. “A Framework for Automated Schenkerian Analysis”. In: (2008), p. 6.
- [22] P. B. Kirlin and J. Yust. “Analysis of Analysis: Using Machine Learning to Evaluate the Importance of Music Parameters for Schenkerian Analysis”. In: *Journal of Mathematics and Music* 10.2 (May 3, 2016), pp. 127–148. DOI: 10.1080/17459737.2016.1209588.

- [23] P. B. Kirlin and D. D. Jensen. “Probabilistic Modeling of Hierarchical Music Analysis.” In: *Proceedings of the 12th International Society for Music Information Retrieval Conference*. 12th International Society for Music Information Retrieval Conference (ISMIR 2011). 2011, pp. 393–398.
- [24] P. B. Kirlin and D. L. Thomas. “Extending a Model of Monophonic Hierarchical Music Analysis to Homophony”. In: *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*. Ed. by M. Müller and F. Wiering. 2015, pp. 715–721.
- [25] J. D. Yust. “Formal Models of Prolongation”. University of Washington, 2006.
- [26] A. Marsden. “Representing Melodic Patterns as Networks of Elaborations”. In: *Computers and the Humanities* 35.1 (Feb. 1, 2001), pp. 37–54. DOI: 10.1023/A:1002705506386.
- [27] A. Marsden. “Schenkerian Analysis by Computer: A Proof of Concept”. In: *Journal of New Music Research* 39.3 (Sept. 1, 2010), pp. 269–289. DOI: 10.1080/09298215.2010.503898.
- [28] D. Harasim, M. Rohrmeier, and T. J. O’Donnell. “A Generalized Parsing Framework for Generative Models of Harmonic Syntax”. In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*. Ed. by E. Gómez, X. Hu, E. Humphrey, and E. Benetos. 2018, pp. 152–159.
- [29] D. Harasim, T. J. O’Donnell, and M. A. Rohrmeier. “Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony”. In: *Proceedings of the 20th ISMIR Conference*. 20th International Society for Music Information Retrieval Conference. CONF. ISMIR, 2019, p. 335. DOI: 10.5281/zenodo.3527812.
- [30] É. Gilbert and D. Conklin. “A Probabilistic Context-Free Grammar for Melodic Reduction”. In: *International Workshop on Artificial Intelligence and Music, IJCAI-07*. 2007.
- [31] M. Granroth-Wilding and M. Steedman. “A Robust Parser-Interpreter for Jazz Chord Sequences”. In: *Journal of New Music Research* 43.4 (Oct. 2, 2014), pp. 355–374. DOI: 10.1080/09298215.2014.910532.
- [32] S. Abdallah, N. Gold, and A. Marsden. “Analysing Symbolic Music with Probabilistic Grammars”. In: *Computational Music Analysis*. Ed. by D. Meredith. Cham: Springer International Publishing, 2016, pp. 157–189. DOI: 10.1007/978-3-319-25931-4_7.
- [33] O. Melkonian. “Music as Language: Putting Probabilistic Temporal Graph Grammars to Good Use”. In: *Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design*. FARM 2019. Berlin, Germany: Association for Computing Machinery, Aug. 23, 2019, pp. 1–10. DOI: 10.1145/3331543.3342576.
- [34] W. Chi, P. Kumar, S. Yaddanapudi, S. Rahul, and U. Isik. “Generating Music with a Self-Correcting Non-Chronological Autoregressive Model”. In: *Proceedings of the 21st International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval Conference (ISMIR 2020). Montreal, Canada: ISMIR, Oct. 11, 2020, pp. 893–900. DOI: 10.5281/zenodo.4245578.
- [35] Z. Wang, D. Wang, Y. Zhang, and G. Xia. “Learning Interpretable Representation for Controllable Polyphonic Music Generation”. In: *Proceedings of the 21st International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval Conference (ISMIR 2020). Montreal, Canada: ISMIR, Oct. 11, 2020, pp. 662–669. DOI: 10.5281/zenodo.4245518.
- [36] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, G. Xia, and J. Zhao. “PianoTree VAE: Structured Representation Learning for Polyphonic Music”. In: *Proceedings of the 21st International Society for Music Information Retrieval Conference*. International Society for Music Information Retrieval Conference (ISMIR 2020). Montreal, Canada: ISMIR, Oct. 11, 2020, pp. 368–375. DOI: 10.5281/zenodo.4245446.
- [37] A. Cadwallader and D. Gagné. *Analysis of Tonal Music: A Schenkerian Approach*. Third Edition. Oxford, New York: Oxford University Press, Mar. 24, 2011. 432 pp.

PKSPELL: DATA-DRIVEN PITCH SPELLING AND KEY SIGNATURE ESTIMATION

Francesco Foscarin Nicolas Audebert Raphaël Fournier S’niehotta
 CEDRIC (EA4629), CNAM Paris, HESAM Université, France
 {francesco.foscarin, nicolas.audebert, fournier}@cnam.fr

ABSTRACT

We present PKSpell: a data-driven approach for the joint estimation of pitch spelling and key signatures from MIDI files. Both elements are fundamental for the production of a full-fledged musical score and facilitate many MIR tasks such as harmonic analysis, section identification, melodic similarity, and search in a digital music library.

We design a deep recurrent neural network model that only requires information readily available in all kinds of MIDI files, including performances, or other symbolic encodings. We release a model trained on the ASAP dataset. Our system can be used with these pre-trained parameters and is easy to integrate into a MIR pipeline. We also propose a data augmentation procedure that helps re-training on small datasets.

PKSpell achieves strong key signature estimation performance on a challenging dataset. Most importantly, this model establishes a new state-of-the-art performance on the MuseData pitch spelling dataset without retraining.

1. INTRODUCTION

Music listening is a complex cognitive process that involves the organization of sound events in time and frequency structures. This process creates patterns of thought that depend either on general principles of cognitive psychology or on prior knowledge and common practices of the relevant musical style system [1]. As far as tonal music is concerned, pitches are related and arranged to create a complex system of perceived stability and instability that gravitates around a tonal center, usually identified by a scale or a chord [2]. In this paper, we focus on two aspects of the tonal framework: pitch spelling and key signature.

1.1 Pitch Spelling

In tonal music, the classification of pitches in 12 *pitch-classes*, each one uniquely identifying a set of frequencies that are n octaves apart, is enriched with some tonal information and creates the higher-level representation of *tonal-pitch-classes*. Each *tonal-pitch-class* consists of a *diatonic*

Pitch-class	Pitch spelling	Tonal-pitch-class
11		$B_{\natural}, C_{\flat}, A_{\times}$
10		$A_{\sharp}, B_{\flat}, C_{\flat\flat}$
9		$A_{\flat}, G_{\times}, B_{\flat\flat}$
8		G_{\sharp}, A_{\flat}
7		$G_{\flat}, F_{\times}, A_{\flat\flat}$
6		$F_{\sharp}, G_{\flat}, E_{\times}$
5		$F_{\flat}, E_{\sharp}, G_{\flat\flat}$
4		$E_{\sharp}, F_{\flat}, D_{\times}$
3		$D_{\sharp}, E_{\flat}, F_{\flat\flat}$
2		$D_{\flat}, C_{\times}, E_{\flat\flat}$
1		$C_{\sharp}, D_{\flat}, B_{\times}$
0		$C_{\flat}, B_{\sharp}, D_{\flat\flat}$

Figure 1. Pitch-classes and corresponding tonal-pitch-classes.

name in $[C, D, E, F, G, A, B]$ and an *accidental* among double-flat ($\flat\flat$), flat (\flat), natural (\natural), sharp (\sharp), double-sharp (\times).¹ For every pitch-class, there are multiple corresponding tonal-pitch-classes, called *enharmonic equivalents* (see Fig. 1). The task of choosing a single name between the possible enharmonic equivalents is called *pitch spelling*.

1.2 Key Signature

Key signatures inform about the tonal center of the piece and are commonly identified in music by a certain number of sharps or flats (whose positions are fixed) or by a tonal-pitch-class (see Fig. 2). Combined with pitch spelling, they also improve readability, according to the principle of *notation parsimony* [3], *i.e.*, the minimization of the number of symbols (accidentals) displayed in the score.

Tightly related to the key signature, the *key* also adds information about the mode (major or minor). Keys are considered in literature either on a global scale (*i.e.*, one *global key* for one piece) or a local scale (*i.e.*, multiple *local keys* for one piece) resulting from modulations and tonicizations (see [4] for a complete description of these concepts). A formalization of the relation between local keys, global key, and key signatures is missing in the literature and it is not the goal of this paper to have a musicological discussion on this topic. As a first approximation, the key signature can

¹ More accidental types, like triple sharps could exist in theory, but they are not used in common music notation.



Figure 2. Corresponding key signatures representation in a musical score, tonal-pitch-class and number of accidentals.

be considered a global feature of the piece. It occasionally changes if the tonal center shifts significantly from its previous position, although it does not move as much as the local key. A possible motivation is that, since the key signature is employed in music notation, the principle of parsimony applies and it “smoothes” short key changes.

1.3 Pitch Spelling and Key Signature Estimation

Multiple reasons motivate systems that perform pitch spelling and key signature estimation. Such information is necessary for the creation of full-fledged musical scores, e.g., as the last step of automatic music transcription or music generation systems. Furthermore, those elements are employed for other MIR tasks: key signature annotations are useful for music search and section identification, and pitch spelling facilitates harmonic analysis and melodic pattern matching [3, 5]. However, they are not represented in “low-level” formats, such as audio and MIDI files², which constitute most of the available digital musical encoding.

1.4 Related Works

Several authors have developed automatic systems that perform pitch spelling from MIDI data. Most of them consist in algorithmic approaches based on musicological insights only, without any learning. Some authors think the selection of the appropriate tonal-pitch-class is a function of the local key [6, 7] or a combination of the local key and voice-leading information [5, 8], i.e., the temporal evolution of notes within each voice. Others base pitch spelling on a type of interval optimizations [9], on the principle of notational parsimony, or a combination of the two criteria [3]. Another formulation of the problem as a generative probabilistic model is proposed by Teodoru et Raphael [10]. They model voices with independent Markov chains conditioned on a hidden state that contains the local key information.

For evaluation purposes, the early algorithmic approaches were compared by Meredith [7] on a dataset of 216 pieces by 8 baroque and classical composers. The highest accuracy at the time was obtained by Meredith’s ps13 algorithm, especially when small temporal variations were introduced to simulate a MIDI file generated from a

performance. Teodoru and Raphael [10] also tested their approach on the same dataset (without tempo deviation) with high accuracy. More recent works [9, 11] did not increase the total accuracy. In this paper, we compare our results with Meredith’s ps13 algorithm and Teodoru and Raphael’s approach with conditional independent voices (CIV).

While those approaches yield a very high accuracy ($\geq 99\%$ of notes are correctly spelled), they are not designed to be easily employed in larger MIR pipelines. ps13 has 3 different outputs, whose tonal pitch classes are transposed by diminished seconds (e.g., one piece in C#, one in D \flat and one in B \times) and does not indicate how to select the “best” version. Both ps13 and CIV parameters are set by hand³, thus making them difficult to generalize across composers and datasets. Moreover, the code of CIV is not publicly available. Finally, both approaches do not provide key signatures, so they cannot be used alone to produce a complete pitch encoding for a musical score. Other pitch spelling systems are implemented in music notation software (e.g., Finale, MuseScore) but no information about their functioning mechanism and performance is available and they require key signatures as input.

Little attention has been given in the literature to the *key signature estimation* problem from MIDI files, in favor of the related task of key estimation [12–15] (see Section 1.2). A direct comparison of our results with other approaches is therefore not possible, but we put our results into perspective by comparing them with the state-of-the-art approach for global key estimation of López et al. [15].

1.5 Our approach

We propose the PKSpell system for jointly estimating pitch spelling and key signature. It yields high accuracy, is easy to integrate into a MIR pipeline, and works on any kind of MIDI file (including MIDI generated from human performance) or other symbolic encodings. Trained on the ASAP dataset [16], PKSpell achieves strong key signature estimation performance on the Albrecht dataset [14] and establishes a new state-of-the-art pitch spelling performance on the MuseData dataset [7] without retraining. Implementation and pre-trained model are publicly available⁴.

We design a deep learning approach based on recurrent neural networks (RNN) that can model correlations in input sequences of variable lengths [17]. We use a dedicated network structure inspired by musicological considerations on the relation between local keys, pitch spelling, and key signatures (details in Section 2.3). Our system models each input note with a pair (pitch-class, duration) that does not require high-level temporal information such as downbeat and time signature or voice separation to be produced. With the proposed dedicated preprocessing of the note durations and data augmentation procedure (see Sections 2.1 and 2.2), our approach can generalize to different tempos, time signatures, and key signatures. This makes it possible to train our model on a small-size dataset of musical scores. Moreover,

² Albeit key signatures can be encoded in a MIDI file as metadata, they are often absent.

³ CIV could be trained, but the authors report that the results are worse than with handset parameters.

⁴ See <https://github.com/fosfrancesco/pkspell>

by cross-validating on a separated dataset, we show that with the pre-trained parameters, our system can correctly handle a variety of different tonal pieces. Finally, our multi-task approach to pitch spelling and key signature estimation increases the performance on both tasks.

2. METHOD

The pitch spelling and key signature information are the results of the relations between different notes. We employ a recurrent neural network architecture, which can learn all those relations for pieces of different lengths. This class of models employs a big number of parameters to correctly learn the input-output function. Our choice of input/output representation helps to keep their number as low as possible, and the data augmentation procedure we propose allows us to learn them from a relatively small dataset.

2.1 Input and output formats

Multiple approaches have been proposed for the problem of transforming a MIDI file into a sequential representation [18–20]. Since we target all kinds of MIDI inputs (see [21] for a description of different MIDI formats), we cannot assume to have information such as voice separation, time signature, downbeat, and beat positions. We employ a simple note-based representation that was proposed by Lopez [15]: a piece is modeled as a sequence of notes $\aleph = \{\eta_1, \eta_2, \dots, \eta_N\}$, ordered according to the temporal position of their onsets. If multiple notes have the same onset position, we take the notes in low-to-high order.⁵ For each note η , we consider two features: pitch-class $p[\eta]$ and duration $d[\eta]$. The usage of note durations is common in key estimation approaches (e.g., [1, 12]), and stems from the musicological intuition that longer notes have a stronger impact than shorter notes in defining the tonal context.

To group all possible note durations in a limited number of classes, we run an (unsupervised) 1-dimensional k -means algorithm for all note durations (see Fig. 3). We use the dynamic programming algorithm presented by Gronlund et al. [22] that has complexity $O(kN + N \log N)$, for k classes and a sequence of length N . We select $k = 4$. This classification cannot be considered more than a rough indication of relative duration and it is not meant to precisely identify beats, downbeats, and other metrical information. Moreover, it will show its limits if there are tempo changes or time signature changes inside the piece. Yet, we found that it improved our model results at a small computational cost and, compared to other possible approaches, e.g., quantization to some discrete durations, it has the advantage of generalizing to different tempos and time signatures.

The output is a sequence of tonal-pitch-classes $\tau[\eta]$ (among the 35 in Fig. 1) and a sequence of key signatures $\kappa[\eta]$ (among the 15 in Fig. 2), one for each note η in input. Computing a key signature for each note might seem excessive, since we may expect the key signature to change only a few times during a piece. However, this is necessary as

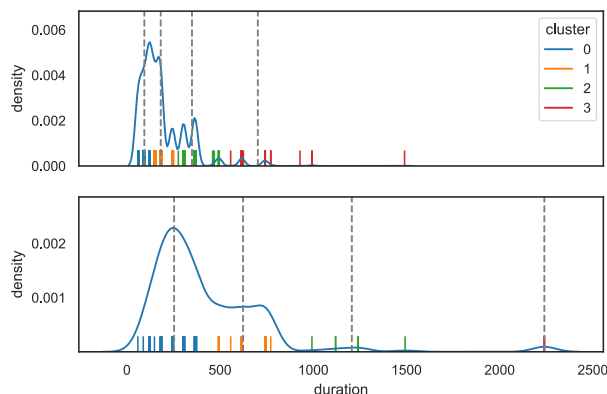


Figure 3. One-dimensional clustering of note durations for 2 pieces expressed in milliseconds. Each vertical line represent a duration and dotted grey lines are cluster centroids. The piece at top is faster than the piece at the bottom. To facilitate the understanding, the kernel density estimation of the note durations is also displayed.

we do not know in advance how many changes there are in a piece, nor do we have precise metrical information.

2.2 Data augmentation

Studies in cognitive psychology [23] have proved that listeners perceive as identical two pieces if all their notes are transposed by the same *interval*. It is common to use this property of music perception to augment a dataset by transposition [18, 24]. For our goals, we must transpose pitch-classes (input) and tonal-pitch-classes and key signatures (output) to have a correct ground truth for training.

The possible transpositions of tonal-pitch-classes and key signatures move through a *spiral of fifths* [12]. When reported to the same octave, each transposition can be identified with a *diatonic interval*, notated with an interval number and type, e.g., augmented 4th, perfect 5th, minor 2nd. For pitch-classes, instead, we are limited to 12 *chromatic intervals* that can be simply identified by integers. In Fig. 4 we report the most common diatonic intervals (the ones closer to the center of the spiral of fifths) associated with their respective chromatic interval.

Since our goal is to learn an input-output mapping, we cannot accept multiple sequences of tonal-pitch-classes that correspond to the same sequence of pitch-class. That means that we need to select only one diatonic interval for every chromatic interval. To do so, we use a heuristic based on the principle of parsimony, i.e., we choose the diatonic transposition which generates the set of tonal pitch classes with the lowest number of accidentals. For each piece and each chromatic transposition, we transpose the tonal pitch class by all corresponding diatonic intervals, then we discard transpositions that contain non-accepted accidentals (e.g., triple sharps). Finally, we select as the “valid” diatonic transposition the one with the smallest count of accidentals, where \flat and \times count as 2, and \sharp and \flat count as 1. This allows us to produce up to 11 variants for each piece, although the number is lower if all diatonic intervals for a certain

⁵ However, the ordering choice does not impact the results.



Figure 4. Chromatic intervals (orange), corresponding diatonic intervals (black) and examples from the tonal-pitch-class C . The abbreviation for diatonic types are “P”: perfect, “M”: major, “m”:minor, “A”: augmented, “d”: diminished, “AA”: doubly augmented.

chromatic interval generate a discarded transposition.

2.3 RNN Architecture

With our input and output choice, the pitch spelling and key signature estimation problems for one piece can be seen as a sequence of multi-task classification problems. Our goal is to assign to each input (*i.e.*, pitch-class p + duration d) two labels: one among the available tonal-pitch-classes τ and one among the key signatures κ . Formally, for a piece C and each note $\eta \in \aleph_C$ we seek to learn the function $F : x[\eta] \rightarrow y[\eta]$, where $x[\eta] = p[\eta], d[\eta]$ is our input and $y[\eta] = \tau[\eta], \kappa[\eta]$ is our output.

We want the output for each note to be dependent on the notes around it, and the length of our sequences (*i.e.*, the number of notes in a piece) is not fixed. We select as the core of our model a recurrent neural network (RNN), that can store information about the “context” in its internal state for variable-length sequences of inputs. Such a model can be trained in a supervised fashion on an annotated dataset that contains for each piece a sequence of pairs $\{(x[\eta], y[\eta]), \forall \eta \in \aleph_C\}$. We optimize the parameters θ w.r.t. a classification loss function \mathcal{L} :

$$\theta^* = \arg \min_{\theta} \sum_C \sum_{\eta \in \aleph_C} \mathcal{L}(F_{\theta}(x[\eta]), y[\eta])$$

We select \mathcal{L} as the sum of two cross-entropy-loss functions [25], one for the key signature and one for the tonal-pitch-class. Since \mathcal{L} is differentiable, the model parameters can be optimized using Stochastic Gradient Descent (SGD).

We use a bidirectional RNN to tie the output at a certain note to the past and future inputs. From a musicological standpoint, “seeing the future” is useful *e.g.*, to know where a certain note will *resolve*. As shown in Fig. 5, the model has two recurrent layers, each one coupled with a linear layer. The first produces the tonal-pitch-classes and the second produces the key signatures. This architecture is based on the musicological hypothesis that pitch spelling depends on the local key [6, 7] and the key signature is a “smoothed” version of the local key. Assuming the first layer encodes information about the local key, we aggregate this information in the second recurrent layer to infer the key signature.

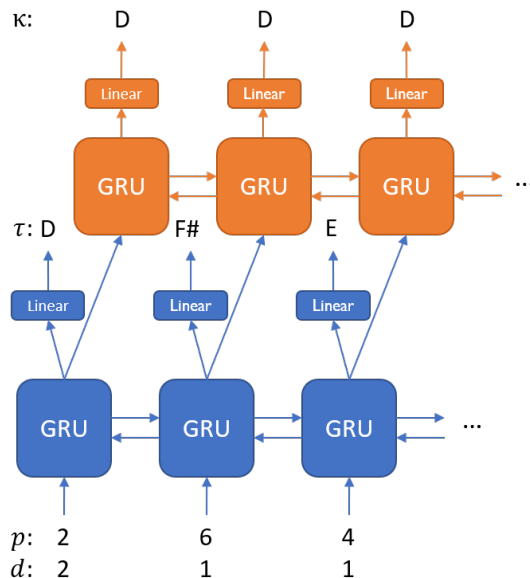


Figure 5. Model architecture. For each pitch-class p and duration d in input, a tonal-pitch-class τ and a key signature κ are produced.

According to the principle of *multi-task learning* [26,27], jointly producing two outputs allows the model to learn shared representations, thus enabling a better generalization on both our original tasks.

3. EXPERIMENTS AND RESULTS

We train our model on the pieces from the ASAP dataset [16]. The dataset provides 222 compositions from several composers over two centuries: Bach, Beethoven, Chopin, Schubert, Haydn, Liszt, Schumann, Mozart, Rachmaninoff, Ravel, Debussy, Scriabin, Glinka, Brahms, Prokofiev, Balakirev. We remove two pieces because they overlap with the dataset that we use for pitch spelling evaluation (see Section 3.2). All the information we need for training can be easily extracted from digitally encoded musical scores, *i.e.*, MusicXML scores for the ASAP dataset. Such scores contain note durations, tonal-pitch-classes, and key signatures. From tonal-pitch-classes it is straightforward to produce pitch-classes using the function in Fig. 1. Duration and pitch-classes are then encoded as one-hot vectors, concatenated to a single vector, and fed into our model. Training on real MIDI performances could also be performed as long as a note-wise score to performance alignment is available. Unfortunately, ASAP does not provide this alignment, and other datasets that provide such information are considerably smaller.

To evaluate the benchmarked approaches, we use the accuracy, *i.e.*, the percentage of correctly inferred symbols. We also use the error rate ($1 - \text{accuracy}$) to improve the readability of some results.

3.1 Hyperparameter search

Our model has five major hyperparameters to consider: type of optimizer, learning rate, batch size, type of recurrent cell, and number of parameters (number of layers and hidden state dimension for each RNN). To find the best combination, we perform a grid search by training our model on 85% of ASAP (187 pieces, 2033 after data augmentation) and validating on the remaining 15% (33 pieces).

Optimizer. We compare various optimizers from the deep learning literature, including gradient descent algorithms with adaptive learning rates or momentum. We find no significant difference in accuracy; though Adam [28] is easier to tune and converges faster than traditional SGD.

Learning rate. All networks are trained for 40 epochs, *i.e.*, 40 passes over the whole augmented training set. We find that a starting learning rate of 0.01, divided by 10 after 20 epochs, allows for fast and robust convergence.

Batch size. We feed mini-batches of sequences in parallel to our model. For batches larger than 32, changing the batch size has no impact on performance. Larger batches tend to increase convergence speed but with a higher GPU memory consumption; thus we settle for a batch size of 32.

Recurrent cell. We compare LSTM [29] and GRU [30] cells. While the accuracy is similar with the two cells, we find that GRU cells are faster and use less memory.

Number of parameters. There are two ways to increase the number of parameters in the RNN: widen it by increasing the hidden dimension, or deepen it by stacking more layers. More parameters generally entail better performance on the train set but not necessarily on the test set due to overfitting. Dropout [31] is required to alleviate overfitting for a depth greater than 1 or a hidden dimension higher than ≈ 200 . For each of the two RNN in our final model, we set depth 1 and hidden dimension 300 (150 in each direction).

3.2 Main results

We evaluate the pitch spelling and key signature estimation results separately.

For pitch spelling, we compare with ps13 and CIV on the MuseData dataset proposed by Meredith [7]. It consists of 216 pieces (195 972 notes) from 8 classical composers (see Table 1) and is also available in a “noisy” version, where some noise is artificially introduced in the onset and offset positions to roughly simulate human performances. There are no pieces in common with our training dataset. It is worth noting that both ps13 and CIV tune their parameters on the test data; this can induce overfitting and thus an optimistic estimate of the system’s performance. We may notice, for example, that the pieces used to train CIV (all Beethoven’s, one-third of Haydn’s, and one-third of Mozart’s), correspond to the pieces in which CIV has the highest performance compared to our system. Moreover, both ps13 and our system are evaluated on the “noisy” dataset, while CIV is evaluated on the quantized dataset. We report our evaluation results in Table 1. PKSpell establishes new state-of-the-art performances on the pitch spelling task. It does 256 errors, around 25% less than CIV (343) and 75% less than ps13 (1064). The learned model has an excellent

generalization, going close to zero error if the pieces have a unique strong tonal center (*e.g.*, Corelli, Handel, Telemann, Vivaldi). More difficult are instances when multiple modulations happen during a short time span (*e.g.*, Bach chorales), and especially around abrupt key signature changes (*e.g.*, Beethoven). The error count for Haydn is particularly high because of the piece Symphony No. 100 in G major, where, at measure 166 there is a sudden change from D \flat major to C \sharp , to make the music easier to read with fewer accidentals. This kind of enharmonic key change is rare in music and our system fails to detect it.⁶ It is worth noticing that, while our model allows a pitch-class to be classified in a non-correspondent tonal-pitch-class (*e.g.*, 2 \rightarrow G \sharp), this kind of error is completely absent in our results. Our model learns very easily to perform the mapping of Fig. 1, especially when using the augmented dataset.

While a direct evaluation of the key signature estimation is not possible due to the lack of other approaches targeting this task, we put into perspective our results by evaluating our system on the Albrecht dataset [14] and comparing it with the state-of-the-art results for the global key signature estimation of López et al. [15]. Our task can be considered easier because we do not need to separate major keys from minor keys; however, while we are considering all enharmonic key signatures (*e.g.*, D \flat with five flats is considered equivalent to C \sharp with seven sharps), López is flattening all enharmonic versions of a key to the same class. After removing the pieces in common with ASAP, we are left with 932 pieces. We correctly classify 97% of the global key signatures. For comparison, López et al. correctly classify 94% of the keys. Of the 29 misspelled pieces, eight are mapped to enharmonically equivalent key signatures, in particular, there is confusion between the C \sharp and D \flat and between F \sharp and G \flat . The remaining 21 wrong estimations are off by one accidental, *i.e.*, the predicted key signature is the relative 4th or 5th of the true key signature.

3.3 Ablation Studies

We perform several ablation studies to understand how our design choices impact the model performance. For each experiment, we remove one element from PKSpell and see how this affects the model performance. If the element is useful, we expect a reduction in the accuracy (see Fig. 6).

Single RNN. As previously introduced, PKSpell uses two recurrent layers, one for pitch spelling and another for key signature estimation. To evaluate this idea, we build a model that has a single recurrent layer for both tasks. A detailed analysis shows that one-layer-PKSpell slightly outperforms the two-layer model when the tonal center is very stable. However, the two-layer model majorly improves the results for more pieces with modulations and key changes. On the ensemble of considered composers, the usage of two separate RNN layers boosts the accuracy, especially for key signature estimation (+3%).

Separate learning. Multi-task learning can help leverage domain-specific information contained in related but

⁶ ps13 manually transposes half of this piece before evaluation. There is no indication of its treatment by CIV.

	Bach	Beethoven	Corelli	Handel	Haydn	Mozart	Telemann	Vivaldi	Total
ps13 [7]	0.17% (42)	1.41% (345)	0.04% (11)	0.26% (68)	0.94% (220)	0.23% (282)	0.34% (82)	0.39% (114)	0.59% (1064)
CIV [10]	0.10% (25)	0.10% (25)*	0.08% (20)	0.02% (6)	0.45% (110)*	0.33% (79)*	0.05% (13)	0.27% (65)	0.18% (343)
PKSpell	0.08% (19)	0.23% (56)	0.02% (5)	0.02% (5)	0.49% (121)	0.14% (35)	0.02% (4)	0.04% (11)	0.13% (256)

Table 1. Error rate and the number of errors (between parentheses) for different composers in Meredith’s Musedata pitch spelling dataset [7]. For ps13 and CIV, we took these values from the respective papers. The symbol * marks the results for the composers used to set the parameters for CIV. The best result for each composer is highlighted in bold.

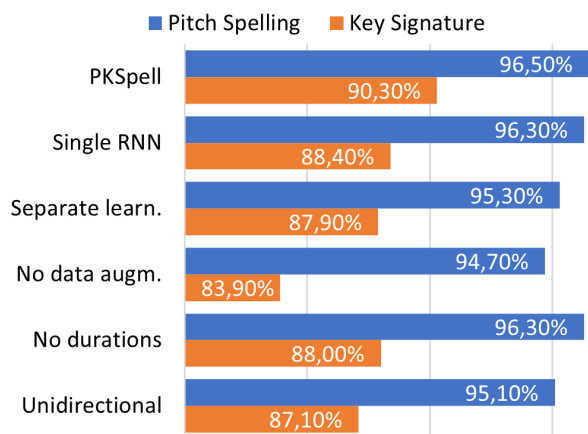


Figure 6. Accuracies for the ablation tests. Results are reported on the validation set of ASAP (33 pieces).

different tasks. This is the case in our system, as the accuracy for both PS and KS estimation improves when trained jointly compared to two distinct RNNs used separately.

No data augmentation. In theory, more data should improve the generalization of the model, provided that the augmented samples are representative of future observations. In our case, data augmentation provides a significant accuracy boost, especially for key signature estimation.

No durations. We use the duration of notes as an input feature for our model. While this is common in key estimation systems, multiple approaches to pitch spelling in the literature do not use this information [3, 7]. We find that durations improve our results, especially for KS estimation.

Unidirectional RNN. The RNN layers we consider can process a sequence either in one direction (usually left-to-right, LTR) or in a bidirectional manner, both LTR and RTL. We expect the bidirectional processing to perform better since the model also “sees” future notes. In our ablation experiment, both PS and KS accuracies increase by more than 1 point by using both directions (we keep the same number of parameters by dividing the hidden dimension by 2 when we use the bidirectional model). Note, however, that the unidirectional LTR model still works reasonably well and could be used for a real-time version of our system.

4. CONCLUSIONS AND PERSPECTIVES

We introduce PKSpell, a novel system for joint pitch spelling and key signature estimation that reaches new state-of-the-art performances on pitch spelling, is easy to integrate into a MIR pipeline, and works on any MIDI file, including human performances. To reach this goal, we perform multi-task learning with an RNN model inspired by musicological insights. We propose a data augmentation procedure and a preprocessing of note durations to generalize to different transpositions, tempos, and time signatures. We consider a minimal set of inputs (pitch-classes and note durations) that do not require high-level information such as time signature, voice separation, and downbeat position. The pre-trained model we provide can be used “as-is” and offers good performance for classical music of different centuries. Thanks to the ablation study, we directly observe the impact of the design choices of our system.

Possible future work concerns the evaluation of PKSpell on different styles of tonal music *e.g.*, pop, folk, and jazz. We expect it to perform well on pop and folk due to their low harmonic complexity. Jazz can be more challenging because of the extensive use of chord extensions and alterations. It is also interesting to study how non-strictly-tonal music in the ASAP dataset (*e.g.*, Debussy) impacts the training of our model. While concepts such as key signature and pitch spelling are less significant for non-tonal music, they are still used to write musical scores. Since PKSpell is not based on strong tonal principles, we expect it to be able to learn those rules as long as some coherent rules exist for pitch spelling and a large enough dataset is provided. A more in-depth analysis can be performed on the treatment of rhythmical information, by considering different ways to group duration values and by varying the number of groups. Other improvements may be done on the model architecture: the state of the art for sequence-to-sequence problems has shifted toward recurrent attentional models and transformers. However, the length of the sequences we are considering (more than 15 000 notes for certain pieces) poses a considerable challenge for full attentional mechanisms, whose memory requirements are quadratic with the input sequence length. Models that scale linearly have been recently proposed [32, 33] and could be a solution for this problem. Finally, it would be interesting to employ our model to perform local and global key estimation.

5. ACKNOWLEDGMENTS

We thank David Meredith for releasing the MuseData corpus and ps13 Lisp implementation. Additional thanks go to Tim Bradshaw for help running ps13 on current systems.

6. REFERENCES

- [1] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 1990.
- [2] R. Van Egmond and D. Butler, “Diatonic connotations of pitch-class sets,” *Music Perception*, vol. 15, no. 1, pp. 1–29, 1997.
- [3] E. Cambouropoulos, “Pitch spelling: A computational model,” *Music Perception*, vol. 20, no. 4, pp. 411–429, 2003.
- [4] N. N. López, L. Feisthauer, F. Levé, and I. Fujinaga, “On local keys, modulations, and tonicizations,” in *Digital Libraries for Musicology (DLfM)*, 2020.
- [5] D. Temperley, *The cognition of basic musical structures*. MIT Press, 2001.
- [6] E. Chew and Y.-C. Chen, “Real-time pitch spelling using the spiral array,” *Computer Music Journal*, vol. 29, no. 2, pp. 61–76, 2005.
- [7] D. Meredith, “The ps13 pitch spelling algorithm,” *Journal of New Music Research*, vol. 35, no. 2, pp. 121–159, 2006.
- [8] H. C. Longuet-Higgins and M. Steedman, “On interpreting Bach,” *Machine intelligence*, vol. 6, 1971.
- [9] B. Wetherfield, “The minimum cut pitch spelling algorithm,” in *International Conference of Technologies for Music Notation and Representation (TENOR)*. Hamburg University for Music and Theater, 2020, pp. 149–157.
- [10] G. Teodoru and C. Raphael, “Pitch Spelling with Conditionally Independent Voices,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2007, pp. 201–206.
- [11] A. K. Honingh, “Compactness in the Euler-lattice: A parsimonious pitch spelling model,” *Musicae Scientiae*, vol. 13, no. 1, pp. 117–138, 2009.
- [12] E. Chew, “The spiral array: An algorithm for determining key boundaries,” in *International Conference on Music and Artificial Intelligence*. Springer, 2002, pp. 18–31.
- [13] D. Temperley and E. W. Marvin, “Pitch-class distribution and the identification of key,” *Music Perception*, vol. 25, no. 3, pp. 193–212, 2008.
- [14] J. Albrecht and D. Shanahan, “The use of large corpora to train a new type of key-finding algorithm: An improved treatment of the minor mode,” *Music Perception: An Interdisciplinary Journal*, vol. 31, no. 1, pp. 59–67, 2013.
- [15] N. N. López, C. Arthur, and I. Fujinaga, “Key-finding based on a hidden Markov model and key profiles,” in *Digital Libraries for Musicology (DLfM)*, 2019, pp. 33–37.
- [16] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, “ASAP: a dataset of aligned scores and performances for piano transcription,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [17] A. Sherstinsky, “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network,” *Physica D: Nonlinear Phenomena*, vol. 404, pp. 132–306, 2020.
- [18] G. Micchi, M. Gotham, and M. Giraud, “Not all roads lead to Rome: Pitch representation and model architecture for automatic harmonic analysis,” *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 42–54, 2020.
- [19] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, “This time with feeling: Learning expressive musical performance,” *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.
- [20] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, “Coupled recurrent models for polyphonic music composition,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [21] D. Back, “Standard MIDI-file format specifications,” 1999, accessed May 5, 2021. [Online]. Available: <http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html>
- [22] A. Grønlund, K. G. Larsen, A. Mathiasen, J. S. Nielsen, S. Schneider, and M. Song, “Fast exact k-means, k-medians and Bregman divergence clustering in 1d,” *arXiv preprint arXiv:1701.07204*, 2017.
- [23] W. J. Dowling, “Scale and contour: Two components of a theory of memory for melodies,” *Psychological review*, vol. 85, no. 4, p. 341, 1978.
- [24] T.-P. Chen and L. Su, “Harmony transformer: Incorporating chord segmentation into harmony recognition,” *neural networks*, vol. 12, p. 15, 2019.
- [25] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.
- [26] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *International conference on Machine learning*, 2008, pp. 160–167.

- [27] L. Deng, G. Hinton, and B. Kingsbury, “New types of deep neural network learning for speech recognition and related applications: An overview,” in *IEEE international conference on acoustics, speech and signal processing*, 2013, pp. 8599–8603.
- [28] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [29] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.
- [30] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1724–1734.
- [31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [32] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, “Big bird: Transformers for longer sequences,” *Advances in Neural Information Processing Systems*, vol. 33, 2020.
- [33] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, “Nyströmformer: A Nyström-based algorithm for approximating self-attention,” 2021.

FILOSAX: A DATASET OF ANNOTATED JAZZ SAXOPHONE RECORDINGS

Dave Foster

Queen Mary University of London
d.foster@qmul.ac.uk

Simon Dixon

Queen Mary University of London
Centre For Digital Music

ABSTRACT

The Filosax dataset is a large collection of specially commissioned recordings of jazz saxophonists playing with commercially available backing tracks. Five participants each recorded themselves playing the melody, interpreting a transcribed solo and improvising on 48 tracks, giving a total of around 24 hours of audio data. The solos are annotated both as individual note events with physical timing, and as sheet music with a metrical interpretation of the timing. In this paper, we outline the criteria used for choosing and sourcing the repertoire, the recording process and the semi-automatic transcription pipeline. We demonstrate the use of the dataset to analyse musical phenomena such as swing timing and dynamics of typical musical figures, as well as for training a source activity detection system and predicting expressive characteristics. Other potential applications include the modelling of jazz improvisation, performer identification, automatic music transcription, source separation and music generation.

1. INTRODUCTION

The study of jazz improvisation has often focused on modelling *what* to play, most recently via deep learning techniques such as transformers [1], language models [2] and GANs [3]. Other recent work [4] has suggested that *how* to play is of equal importance when generating convincing synthesised performances. To properly examine the minutiae of how a performer plays, one requires a wealth of clean, isolated and consistent recordings, which are hard to come by when looking specifically at jazz music.

Another issue when researching the expressive nature of jazz performances is the difficulty of making pair-wise comparisons between performers. This is because there is very little overlap between the recorded corpora of any two musicians, especially when hoping for consistency of both key and tempo. Whereas classical music researchers can draw upon multiple recordings of the same pieces [5, 6], jazz music researchers have only sparse instances of duplicated “head” statements and common “licks” upon which to make their comparisons.

These were the motivations behind the commissioning and curation of the Filosax dataset, which was designed to provide both the isolated recordings and homogeneity of stimuli to allow for the analytical fidelity and inter-participant consistency that are required. The period 2020-2021 inadvertently proved a good period for enrolling willing participants, as COVID-related lockdowns meant that there was a dearth of live performing opportunities, and hence expert performers and improvisers had more time and inclination to take part than they might otherwise have done. The downside of the lockdown backdrop was that we were unable to record the musicians in exactly the same environment. We attempted to mitigate for this by collating an environment-agnostic recording kit, which was sent between the participants and meant that the recordings are as consistent as possible.

A novel aspect of the dataset is the dual note-level annotations that accompany the audio data. The first is a segmentation of the soloist audio into discrete note-events with pitch tags and precise timings, which allows the user to determine exactly *how* a note was played. The precise perturbations of pitch, amplitude and timbre can be measured and quantified, unencumbered by the requirement to filter out (or interpolate from) the accompaniment. The second is a sheet music representation (at a level of detail akin to the “Omnibook” series¹), where each note is assigned a place in the metrical grid by an expert human jazz transcriber. The mapping between the two annotation layers provides, perhaps, the most useful insight: given a sequence of notes, how does the performer play each one, given its position in the sequence?

We recognise that a set of annotated, source separated recordings will also be of use to researchers in related fields, some of which are discussed in section 7. Due to licensing issues, the full Filosax dataset cannot be made publicly available, but suitable researchers are welcome to apply to the authors to receive copies of the non-copyright material and annotations, along with instructions on how to purchase the copyrighted material and automatically reconstruct the full dataset.

2. RELATED WORK

The Weimar Jazz Database (WJD) [7] was a landmark in annotated jazz data, with 456 manually transcribed solos by 78 performers, and featuring music from a broad range



© D. Foster and S. Dixon. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Foster and S. Dixon, “Filosax: A Dataset of Annotated Jazz Saxophone Recordings”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ <https://www.halleonard.com/series/OMNIBK?subsiteid=65&&dt=item#products>

of eras (1925-2009). The inclusion of additional information about idiom, style, and form allow it to be used for many MIR purposes as well as ethnomusicological research. For our purposes, it is hampered by the medium (stereo mixes) and original choice of repertoire (chosen by the performers for artistic rather than scientific reasons) from which the authors could only take samples. There are only 3 pieces played by multiple tenor saxophonists (“Body and Soul” by 12 musicians, “Night and Day” and “U.M.M.G.” both by 2 musicians), and there are no readable (human corrected) score representations.

The “Dig That Lick” project developed pattern mining within the WJD [8] as well as the collation of the DTL1000 dataset [9], a set of automated transcriptions of 1750 solos from 1060 tracks. The algorithm used for transcribing the melody/lead line achieves a mean F1 score of 0.85, which the authors suggest is adequate for large-scale pattern mining, but implies that a significant amount of manual correction would be needed for the data to be of sufficient accuracy for note-level analysis.

The MedleyDB database [10] contains almost 200 tracks with full mixes and separate instrumental/vocal stems in a variety of styles and with pitch annotations for the lead line. Less than 10, however, are appropriate jazz recordings, representing approximately 15 minutes of data.

The best practices for the definition and presentation of an MIR dataset [11, 12] were useful in guiding the design and presentation of our dataset. The processes of compiling and annotating the DALI dataset [13] (of synchronised note/lyric annotations) were similar to the approaches that we have used, as is the means of addressing the issues around the distribution of copyrighted material.

3. DATASET CURATION

The diversity within the corpus of recorded jazz music is so broad that an attempt to capture the full variety of it within a database of this size would be futile. We chose to set a goal of a focusing on depth over breadth and, to this end, decided that we would record a single instrument (the tenor saxophone, possibly the most ubiquitous melodic instrument in recorded jazz) and a narrow scope of mid-tempo “standards” with a quasi-fixed tempo, in 4/4 time and a “swing” feel. We chose to engage five expert performers and improvisers on the instrument, to capture a variety of expressive approaches and to allow for inter-participant comparisons to be made.

3.1 Repertoire

To collect a sufficiently varied set of notes, whilst capturing sufficient repeated elements, we decided to base the dataset on a representative repertoire of pieces. Each participant would record themselves playing these pieces with the same accompaniment, and on each piece would play the melody (the “head”), interpret a transcribed solo, and improvise their own solo.

3.1.1 Accompaniment

A dataset which fully encompasses the jazz improvisation process would capture the interaction between the soloist and rhythm section. The Filofox dataset does not attempt to capture this, as to do so would sacrifice the comparisons which it allows to be made between the various performances under identical conditions. Hence, the interaction process is only in one direction: the soloist can respond to what is heard on the accompaniment, but there is no opportunity for the accompaniment to respond or for a feedback loop to be established.

We chose to use pre-recorded accompaniments from Jamey Aebersold², a commercial library of “play-along” tracks recorded between 1967 and the present day. The library consists of over 1000 tracks of jazz standards, recorded with different musicians but with a similar audio presentation of piano+drums in the left channel and bass+drums in the right channel. The use of commercial recordings for this purpose greatly extended the potential repertoire from which we could choose: the alternatives were to use freely available resources (of which there are very few), commission more recordings or to use synthesised accompaniments. Using any of the alternatives would be to the detriment of the range or quality of the data, or would require much more sophisticated recordings. The downside to using the commercial recordings is that we will be unable to distribute them with the dataset.

3.1.2 Solo transcriptions

For the transcribed solos, we sought a group of celebrated jazz artists, whose stylistic output was somewhat similar to each other. We selected 6 such musicians who met the following criteria:

- Made recordings in the 1950’s, 1960’s and 1970’s,
- Made recordings with a discernible and repeated chord sequence,
- The set of their recorded corpus intersects with the set of available backing tracks (in the same key and at a similar tempo).

The musicians chosen (who could broadly be described as performing within the “hard-bop” sub-genre) were: Stan Getz, Dexter Gordon, Tubby Hayes, Joe Henderson, Sonny Rollins and Ben Webster. Each is represented in the dataset by 8 extracts of their recorded solos, from the private collections of the authors, which were transcribed and typeset by the authors. The details of the original recordings were made available to participants, in case they felt it would be useful in developing their own interpretation.

3.1.3 Piece choice

All of the available Jamey Aebersold accompaniment tracks were examined and meta-data extracted (tempo, key, duration, number of choruses, time signature, rhythmic feel). Candidate pieces were found using the following criteria:

² <http://jazzbooks.com/jazz/JBIO>

- Entirely in 4/4 time, with a quasi-fixed tempo (allowing for gradual changes),
- With a “swing” feel throughout,
- A repeated chord sequence with sufficient harmonic movement to not be classed as “modal” (which we define as having multiple sections where a single chord is held for longer than 4 bars),
- Tempo in the range 100-240 beats per minute,
- An accompaniment consisting exclusively of piano, bass and drums.

This truncated list was cross-referenced with the discographies of the jazz artists listed in section 3.1.2 where, for a piece to be considered, the recorded version must be: in the same key, within 10% of the backing track tempo, and with at least 1 “chorus” of improvisation by the artist. When a selection (larger than 8) was available, a broad range of tempos, keys and modalities was sought. The full list of pieces is given on the dataset web page³.

3.2 Participants

The five musicians recruited to make the recordings were all known to the authors as expert performers and improvisers on the tenor saxophone. They were invited to participate on the understanding that they had access to an appropriate space in which to record themselves, a good quality instrument and a computer capable of making the recordings. On completion of the recordings, they were asked to take part in an informal follow-up session to review the transcriptions and to reflect on their experiences of making the recordings. Each read and agreed to the ethics, information and participation forms associated with the study, and was compensated with an Amazon gift voucher for £100 on completion of their recordings.

3.3 Recording

The recordings were made consecutively by the five participants, in their own homes, having been supplied with a set of materials, recording equipment and instructions.

3.3.1 Materials

For each of the 48 pieces, the participants were supplied with a printed copy of the sheet music and a copy of the corresponding digital audio workstation (DAW) file (segmented into bars and choruses) to record into. Access to the materials was given prior to receiving the recording equipment, in case they wanted to prepare.

3.3.2 Equipment

A flight case of equipment was shipped between the participants, containing an Aston Stealth microphone, a Focusrite Scarlett Solo USB audio interface, a reflection shield, closed-back headphones, microphone stand, XLR cable and USB cable. Directions on how to assemble and set up the equipment were given, where the settings for the microphone and audio interface were prescribed. Suggestions of how to choose and prepare a suitable room were included,

as well as the exact positioning of the microphone from the instrument.

3.3.3 Instructions

Participants received a document containing detailed information on the goals of the data collection, and how to approach their interpretation of the material. For the “head” section, they were told to perform this freely, as if in a live performance: adding, removing, changing or moving notes as they please (whilst still ensuring that it is identifiably the melody). For the interpretation of the recorded solo, they were asked to play accurately, to the best of their ability, without intentionally changing notes but with grace notes, articulation, slurs and “scoops” as they felt. For the improvised sections, they were asked to approach this more as a practice session than a concert: that is, to include repetition, longer notes, and to explore the full tessitura and dynamic range of the instrument, more than they might otherwise do in a concert setting.

4. ANNOTATION

The recordings in their entirety were annotated at two levels: firstly, an accurate list of note start times, end times, and homogenised pitch (semitone granularity); secondly, a simplified sheet music representation, where an interpretation of the intended rhythm was notated, using a pre-determined set of granularity assumptions. Figure 1 shows the semi-automatic transcription pipeline that was developed for the expedited and accurate annotation of the recordings. Commercially available software was used to aid with the rough segmentation of the audio into discrete note events, before standard MIR packages were used to obtain the absolute temporal boundaries of each event.

4.1 Initial Annotation

4.1.1 Bar / Beat Annotation

The annotation process began before the recordings were made. The backing track audio files were imported into the DAW Logic⁴, having first been normalised into the range $[-1, 1]$ in Audacity. The “Smart Tempo” function (a proprietary beat mapping algorithm) was used to annotate the file into bars and beats, and corrected by the authors where the downbeat or tempo scale was wrongly inferred. Choruses were duplicated or deleted at this stage, so that the duration of each piece was between 5:50 and 6:20 minutes. Finally, markers were added to correspond with the rehearsal marks on the sheet music, to visually guide the participants whilst they were recording.

4.1.2 Approximate note segmentation

When a participant returned a completed file, the note segmentation transcription could take place. This was initially done with the Logic “Flex Pitch” function (another proprietary algorithm, which performs pitch and onset detection), with human correction both before and after conversion to MIDI. The authors found that, on average, 9%

³<https://dave-foster.github.io/filosax/>

⁴<https://www.apple.com/uk/logic-pro/>

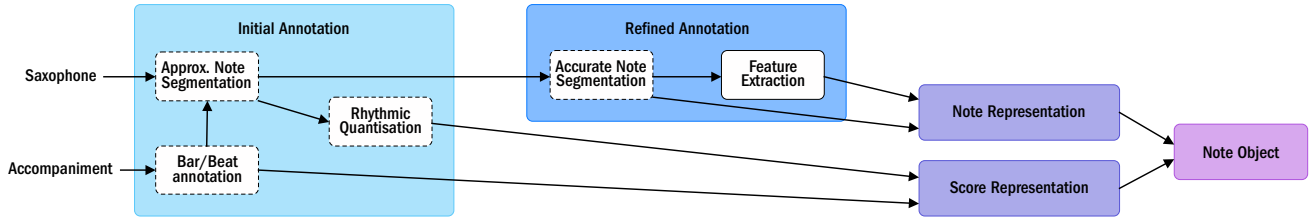


Figure 1: The annotation pipeline. Blocks with dashed lines signify those steps which are manually checked.

of notes in a file needed revising for pitch, timing, separation or concatenation. At this stage, the ground truths regarding the correct number (and order) of note events, their homogenised pitches and approximate timings were established, with the exact values being confirmed at a later stage, described in section 4.2. This workflow was found to maximise the utility of the human intervention (identifying pitches and approximate note boundaries), allowing the exact timings to be sought automatically.

4.1.3 Sheet music representation

The resulting MIDI track from the previous stage was duplicated for the basis of the sheet music representation, which was found to be of sufficient accuracy for this purpose. A granularity of triplet semiquavers was used, but only for notes which were as short as that (for example, in a rapid run of notes). In the main, we determined that a quantisation to a grid of quavers (for both onsets and offsets) would best match the idiomatic style of notation and, by doing so, adhere to the convention of notating off-beat quavers as being “on the grid”, regardless of the amount of “swing” that was used to delay them. After this initial quantisation, notes of a shorter duration were human-edited for rhythmic value on a case-to-case basis, where an ethos of readability over absolute temporal accuracy was employed in ambiguous cases.

Grace notes were consistently notated as acciaccatura, placed a triplet semiquaver ahead of the beat, for easier identification later in the pipeline. No slurs, dynamics or articulations are included in the annotations, which are left by the authors for possible future rounds of annotation, where the audio data could be used to generate suggestions in each case. In the sheet music which accompanies the dataset, there is a small exception, in that staccato crochets are used for readability whenever an on-the-beat quaver is followed by a quaver rest (similarly for staccato quavers).

4.2 Refined Annotation

4.2.1 Note boundaries

The three audio files (saxophone and split backing track) were exported from Logic, along with a MIDI file, containing the bar and beat timings and the two sets of note annotations. These were all processed by a Python script (using `mido`⁵ for MIDI functions), for finding accurate note timings and for serialising the data into the final database format. The former was performed by separating the audio into “phrases” (consecutive sequences of notes found in the

sheet music representation) by identifying gaps between note events in the score representation. These “phrases” may not correspond with the traditional definition, as they can contain just a single note.

The time values from the approximate note segmentation were used as estimates of the temporal mid-point of each note, so for a note N_k with approximate start and end times \hat{t}_k^s and \hat{t}_k^e , the mid-point $\hat{t}_k^m = (\hat{t}_k^s + \hat{t}_k^e)/2$.

We formally describe the process for refining the note boundaries as follows. Each performer’s interpretation of a piece is said to consist of a sequence of I phrases $(P_i)_{i=1}^I$, where each phrase P_i is a sequence of K_i sounding entities N_k with approximate start, mid-point and end (\hat{t}_k^s , \hat{t}_k^m and \hat{t}_k^e), start time t_k^s , end time t_k^e and pitch f_k . Hence,

$$P_i = (N_k)_{k=1}^{K_i}, \quad \text{where } N_k = (\hat{t}_k^s, \hat{t}_k^m, \hat{t}_k^e, t_k^s, t_k^e, f_k). \quad (1)$$

The phrases do not overlap, so $P_i(t_{K_i}^e) \leq P_{i+1}(t_1^s)$ for each i . Iterating by phrase, the corresponding audio was analysed using both the Madmom [14] “CNNOnsetProcessor” and Essentia [15] PYIN [16] implementation, giving a sequence of onsets $(O_j)_{j=1}^J$, a pitch curve $F(t)$ (a sequence of real or null values for each time frame $t \in [P_{i-1}(t_k^e), P_{i+1}(\hat{t}_1^s)]$) and a loudness curve $L(t)$ (a sequence of loudness values for each time frame t).

The start time t_k^s and end time t_k^e (in frames, where $\hat{t}_k^s < \hat{t}_k^m < \hat{t}_k^e$) for each N_k is determined by looking backward and then forward from the mid-point, so,

$$t_k^s = \max\{t_{k-1}^e, F_{null}^s + 1, L_{quiet}^s + 1\}, \quad (2)$$

$$t_k^e = \min\{F_{null}^e - 1, O_{first}^e, L_{quiet}^e\}, \quad (3)$$

where F_{null}^s and F_{null}^e are the time steps of the first null pitch value encountered, counting backwards and forwards from the mid-point \hat{t}_k^m respectively, O_{first}^e is the first onset encountered after the mid-point, and L_{quiet}^s and L_{quiet}^e are the first times (before and after the mid-point) when the loudness $L < L_{thresh}$, a threshold value. The notes are sequential and monophonic, hence $t_k^e \leq t_{k+1}^s$ for each k .

In the event of a continuous pitch curve and absence of a detected onset on or around the expected position of the boundary between two entities N_k and N_{k+1} , we define:

$$t_k^e = \begin{cases} \hat{t}_k^e, & \text{when } f_k = f_{k+1}, \\ \operatorname{argmin}_t \{F(t) > \frac{f_k + f_{k+1}}{2}\}, & \text{when } f_k < f_{k+1}, \\ \operatorname{argmin}_t \{F(t) < \frac{f_k + f_{k+1}}{2}\}, & \text{when } f_k > f_{k+1}. \end{cases} \quad (4)$$

Where an onset occurs before the first pitch value (likely due to a breath or key noise) or there is a gap in the pitch

⁵<https://github.com/mido>

curve between neighbouring notes ($t_k^e < t_{k+1}^s$), an unpitched entity object is recorded in the sequence. If K_i^P is the number of pitched entities and K_i^U the number of unpitched entities in each phrase P_i , then $K_i = K_i^U + K_i^P$ and $K_i^P \geq K_i^U$.

The output of this process is manually validated by the annotator by listening concurrently to the original recording and a synthesised version of the annotations.

4.2.2 Note attributes

With the exact timing of all the notes now known, the attributes of each note can be extracted. Another Python script is employed to automate this process, where it captures both the continuous curves and interpolates landmark values. The pitch curve is extracted (again with the Essentia PYIN function, and constrained to the vicinity of the determined pitch), and the average pitch, time to average pitch, and average vibrato rate and extent (using Essentia functions) are all estimated. Similar curves and features are extracted for amplitude, spectral centroid and spectral flux. It is these attributes that we identified as crucial to the initial goal of studying expressive performance: other use cases for the dataset may require different features, the extraction of which could be automated in a similar fashion. The chord(s) over which the note is played is derived from the published chord sequence, and used to derive the scale degree(s) relative to the chord root.

4.3 Dataset structure

The dataset D is presented as an ordered set of uniquely identifiable sounding entities (pitched and unpitched), which have the following attributes:

- start_time,
- end_time,
- musician_number,
- piece_number,
- bar_number,
- bar_type (head, written solo or improvisation),
- tempo.

In addition, pitched entities have the following attributes:

- MIDI_pitch,
- score_start_time,
- score_end_time,
- score_rhythmic_position,
- score_rhythmic_duration,
- is_grace_note,
- chord_changes (an array, of length 1 for short notes, and possibly > 1 for longer notes),
- scale_degrees (an array).

The entities are collected sequentially in a JSON file (conforming to the JAMS specifications [17]), allowing for easy searching, analysis and n-gram construction. The data is also made available as a set of both MIDI and MusicXML files (the latter just containing the sheet music representation), although the entities themselves contain all the information needed to reconstruct both of these formats. The attributes described in section 4.2.2 are also contained in the JSON file, as are the corresponding full curve values.

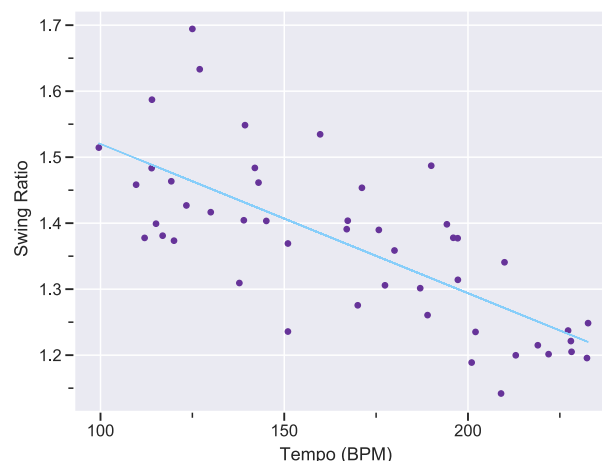


Figure 2: The ‘swing ratio’ of a single participant as a function of tempo. Each point represents the mean tempo and ratio of quavers played in a single piece.

5. ANALYSIS

We present the results of two analysis studies of the Filosax dataset, making comparisons with previous analysis of similar data, as demonstrations of how the dataset can be used.

5.1 Swing Ratio

The presentation of the pitched entities (described in section 4.3) allows for the “groove” or “swing” of a note to be instantly calculated, by comparing the `score_start_time` attribute (the time at which the note starts in the score representation) to the `start_time` attribute (when the note actually starts).

Figure 2 shows the “swing ratio”, the duration of the first of a pair of quavers divided by the duration of the second, as a function of the current speed of the piece. The blue line-of-best-fit shows the same negative correlation found in other analyses of swing rhythm [18–20].

5.2 Enclosed notes

“Enclosing” notes is a device used in the construction of “bebop” phrases, where a chord tone is preceded by both a note above and below (diatonically or chromatically). To show how the dataset can be used for deriving performance parameters, we search the dataset for instances of these 3-grams of consecutive quavers.

Figure 3 shows the range of loudness values where the third note is an off-beat chord tone, and the preceding notes are above and then below that pitch. The graph on the left is derived from instances where one or both of the preceding notes are greater than a tone away, and on the right where both preceding notes are within a tone (“enclosed”).

In the first case, the notes are given almost equal emphasis, whereas the second case shows a characteristic emphasis on the first and (to a lesser extent) third notes, by means of “ghosting” (placing less emphasis on) the second. The ranges derived could be used as probability distributions for generating phrase-level performance instructions.

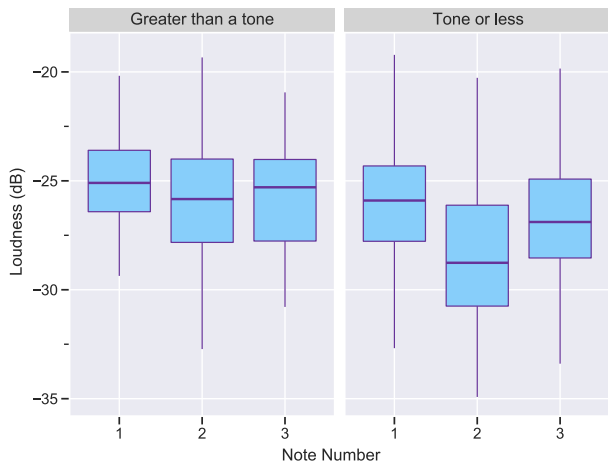


Figure 3: Relative loudness of quavers approaching an off-beat chord tone (3) from above (1) and below (2).

6. EXPERIMENTS

6.1 Activity detection

The Filosax dataset was used to train a system for saxophone activity detection from mixed recordings, that outputs a binary value for each time frame. The solo saxophone recordings were used to generate the ground truth, and mixes of the three stems were used as input. A variation of the U-Net architecture used for the similar task of vocal detection [21] was employed, and various input representations were experimented with.

The best performing combination used a CQT input (achieving an AUROC mean of 0.933), slightly higher than was achieved for vocal activity detection in the original paper (AUROC mean of 0.924). Neither the use of joint training (separation and detection), HCQT input nor HCQT with phase input yielded any improvement in results.

6.2 Expressive Timing

A sequence-to-sequence language model was used to predict performance parameters by framing the problem as a translation task: a “sentence” of “words” (a phrase of notes, using attributes from the dataset) was “translated” into a sequence of expressive instructions (timing, loudness and articulation, all derived from the dataset) via word embeddings and a context vector.

This preliminary system was able to learn the fundamentals of “swing” rhythm (despite it not being explicitly encoded in the input representation), but the rendered output, in its current state, is not of a standard that will impact the bookings of any human jazz musicians. Refinement of this system will form the basis of future research.

7. POTENTIAL APPLICATIONS

We outline several potential applications for the Filosax dataset, outside the field of expressive performance.

7.1 Jazz Improvisation

The dataset contains multiple hours of improvised jazz solos, with corresponding chord and form annotations, which those researching *what* to play could use to train their systems, either as a standalone resource or augmenting another dataset. The range of performances of the “head” of each piece could aid in the study of melody interpretation, and the performances of the transcribed solo could inform research in jazz education.

7.2 Predominant Melody Extraction

Predominant melody extraction in jazz has been restricted to using note annotations with the ensemble recording source [22, 23]. With the Filosax dataset, this type of system could be trained with various mixes of soloist/accompaniment, potentially leading to a system which is more robust to variations in melodic prominence.

7.3 Source Separation

Similar to the previous use case, the 3 distinct audio stems could be leveraged to develop jazz-specific source separation architectures, or existing architectures could be trained with the data. This could lead to improved methods for isolating the solo instrument on jazz recordings which, in turn, could inform more accurate automated data collection from ensemble recordings.

8. DISTRIBUTION

The backing tracks and the melodies are all under copyright, so the Filosax dataset cannot be made public. To ensure reproducibility and to facilitate the adoption of the dataset, we will allow researchers (on application) to access the saxophone recordings, annotations and sample notebooks on the Zenodo repository⁶. We will also provide the list of backing tracks required, and a Python module for checking, normalising and segmenting (see section 4.1.1) the files, in order to accurately reconstruct the data. The module is part of *mirdata* [24], an open-source tool for the distribution of datasets and corresponding annotations, which ensures that the user has the canonical version of all the components.

9. CONCLUSION

No jazz musician ever decided the programme of their concert or album based on what might be useful to future scientists, nor intentionally played an identical chorus to that played by one of their peers because it would provide useful data. We propose that the introduction of the Filosax dataset somewhat tackles these issues, and does so without unduly compromising the stylistic or artistic credibility of the music. The data has already proved to be an invaluable resource for our ongoing research, and we share our rationale, methodology and the data itself in the hope that it may also be for others.

⁶<https://zenodo.org>

10. ACKNOWLEDGMENTS

The first author is a research student at the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, supported by UK Research and Innovation [grant number EP/S022694/1].

11. REFERENCES

- [1] S.-L. Wu and Y.-H. Yang, “The jazz transformer on the front line,” in *21st International Society for Music Information Retrieval Conference*. ISMIR, 2020.
- [2] S. H. Hakimi, N. Bhonker, and R. El-Yaniv, “Bebop-net: Deep neural models for personalized jazz improvisations,” in *21st International Society for Music Information Retrieval Conference*. ISMIR, 2020.
- [3] N. Trieu and R. Keller, “JazzGAN: Improvising with generative adversarial networks,” in *Proc. Int. Workshop on Musical Metacreation*, 2018.
- [4] K. Frieler and W.-G. Zaddach, “Evaluating an analysis-by-synthesis model for jazz improvisation,” 2021, under review.
- [5] K. Kosta, O. F. Bandtlow, and E. Chew, “Mazurkabl: Score-aligned loudness, beat, expressive markings data for 2000 Chopin Mazurka recordings,” in *Proceedings of the Fourth International Conference on Technologies for Music Notation and Representation (TENOR)*, 2018, pp. 85–94.
- [6] T. Gadermaier and G. Widmer, “A study of annotation and alignment accuracy for performance comparison in complex orchestral music,” in *20th International Society for Music Information Retrieval Conference*. ISMIR, 2019.
- [7] M. Pfeleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, *Inside the Jazzomat: New Perspectives for Jazz Research*. Schott Music GmbH, 2017.
- [8] K. Frieler, F. Höger, M. Pfeleiderer, and S. Dixon, “Two web applications for exploring melodic patterns in jazz solos,” in *19th International Society for Music Information Retrieval Conference*. ISMIR, 2018, pp. 777–783.
- [9] L. Henry, K. Frieler, G. Solis, M. Pfeleiderer, S. Dixon, F. Höger, T. Weyde, and H.-C. Crayencour, “Dig that lick: Exploring patterns in jazz with computational methods,” *Jazzforschung/Jazz Research*, Vol. 50, 2020.
- [10] R. M. Bittner, J. Wilkins, H. Yip, and J. P. Bello, “MedleyDB 2.0: New data and a system for sustainable data collection,” in *ISMIR Late Breaking and Demo Papers*. ISMIR, 2016.
- [11] G. Peeters and K. Fort, “Towards a (better) definition of the description of annotated MIR corpora,” in *13th International Society for Music Information Retrieval Conference*. ISMIR, 2012, pp. 25–30.
- [12] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, “Open-source practices for music signal processing research,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 128–137, 2018.
- [13] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, “Creating DALI, a large dataset of synchronized audio, lyrics, and notes,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.
- [14] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new python audio and music signal processing library,” in *Proceedings of the 24th ACM International Conference on Multimedia*, 2016, pp. 1174–1178.
- [15] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, “Essentia: An open-source library for sound and music analysis,” in *Proceedings of the 21st ACM International Conference on Multimedia*, 2013, pp. 855–858.
- [16] M. Mauch and S. Dixon, “pYIN: A fundamental frequency estimator using probabilistic threshold distributions,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 659–663.
- [17] E. J. Humphrey, J. Salamon, O. Nieto, J. Forsyth, R. M. Bittner, and J. P. Bello, “JAMS: A JSON annotated music specification for reproducible MIR research,” in *15th International Society for Music Information Retrieval Conference*. ISMIR, 2014, pp. 591–596.
- [18] A. Friberg and A. Sundström, “Swing ratios and ensemble timing in jazz performance,” *Music Perception: An Interdisciplinary Journal*, vol. 19, no. 3, pp. 333–349, 2002.
- [19] C. Dittmar, M. Pfeleiderer, S. Balke, and M. Müller, “A swingogram representation for tracking micro-rhythmic variation in jazz performances,” *Journal of New Music Research*, vol. 47, no. 2, pp. 97–113, 2018.
- [20] C. Corcoran and K. Frieler, “Playing it straight: Analyzing jazz soloists’ swing eighth-note distributions with the weimar jazz database,” *Music Perception: An Interdisciplinary Journal*, vol. 38, no. 4, pp. 372–385, 2021.
- [21] D. Stoller, S. Ewert, and S. Dixon, “Jointly detecting and separating singing voice: A multi-task approach,” in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 329–339.
- [22] S. Balke, C. Dittmar, J. Abeßer, and M. Müller, “Data-driven solo voice enhancement for jazz music retrieval,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 196–200.

- [23] J. Abeßer, K. Frieler, E. Cano, M. Pfeiderer, and W.-G. Zaddach, “Score-informed analysis of tuning, intonation, pitch modulation, and dynamics in jazz solos,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 168–177, 2017.
- [24] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, “mirdata: Software for reproducible usage of datasets.” in *19th International Society for Music Information Retrieval Conference*. ISMIR, 2019, pp. 99–106.

AN INTERPRETABLE MUSIC SIMILARITY MEASURE BASED ON PATH INTERESTINGNESS

Giovanni Gabbolini

Insight Centre for Data Analytics
School of Computer Science & IT
University College Cork, Ireland

giovanni.gabbolini@insight-centre.org

Derek Bridge

Insight Centre for Data Analytics
School of Computer Science & IT
University College Cork, Ireland

d.bridge@cs.ucc.ie

ABSTRACT

We introduce a novel and interpretable path-based music similarity measure. Our similarity measure assumes that items, such as songs and artists, and information about those items are represented in a knowledge graph. We find paths in the graph between a seed and a target item; we score those paths based on their interestingness; and we aggregate those scores to determine the similarity between the seed and the target. A distinguishing feature of our similarity measure is its interpretability. In particular, we can translate the most interesting paths into natural language, so that the causes of the similarity judgements can be readily understood by humans. We compare the accuracy of our similarity measure with other competitive path-based similarity baselines in two experimental settings and with four datasets. The results highlight the validity of our approach to music similarity, and demonstrate that path interestingness scores can be the basis of an accurate and interpretable similarity measure.

1. INTRODUCTION

The concept of similarity is central to music information retrieval, as it underpins important applications like recommender systems and visualisation in user interfaces [1]. In recent years, there has been a surge in the use of knowledge graphs to solve various information retrieval tasks. For example, knowledge graphs have been applied to recommender systems [2] and question answering [3]. In this work, we use knowledge graphs to gauge music similarity. We introduce a novel path-based similarity measure. We represent items, such as songs and their artists, and information about those items in a knowledge graph; we find paths in the graph between a seed and a target item; we score those paths; and we aggregate the scores to determine the similarity between the seed and the target.

We propose to score paths based on interestingness. Interestingness is introduced in [4] as a way of distinguishing

Path (in natural language)	Int.
George Jones was married to Tammy Wynette.	0.60
Tammy Wynette was the parent of the artist Georgette Jones and George Jones was also the parent of the same artist.	0.56
Tammy Wynette wrote “Our Private Life” and George Jones also wrote the same song.	0.45
Tammy Wynette has made country music, and so did George Jones.	0.34
Tammy Wynette was based in United States and George Jones was based in the same country.	0.31
Tammy Wynette was a solo artist, and George Jones was a solo artist.	0.29

Table 1. An example of the working of our proposed similarity measure. In this example, it is given the artists Tammy Wynette and George Jones as seed and target items.

more interesting from less interesting paths in a knowledge graph. In [4], the authors use what they call *segues* to connect a song to the next song, e.g. in a playlist. Segues are translations into natural language of interesting paths in a knowledge graph. We also employ the path-to-text module from [4], useful to translate paths to natural language. In this paper, we are extending [4] by showing that these paths and their interestingness scores can be the basis of an accurate and interpretable similarity measure.

For example, given the artists Tammy Wynette and George Jones as seed and target, our algorithm finds six paths between the two items, then it assigns an interestingness score to the paths, and it aggregates the scores to determine the similarity. It can convert the paths to natural language for display to a user. We report in Table 1 its natural language translations of the paths (Path column) and the interestingness scores (Int. column).

A distinguishing feature of our similarity measure is its interpretability. Miller defines interpretability as the degree to which a human can understand the cause of a decision [5]. According to this definition, our similarity measure is interpretable. Users can understand the causes of a similarity score by looking at natural language translations of the paths used to compute it, ranked by their interestingness. For example, a user can understand why our al-



gorithm thinks that Tammy Wynette and George Jones are similar, when presented with the ranked list of explanations in Table 1. Notice how the interestingness score intuitively reflects the interestingness of the paths. For example, two artists being married has higher score than two artists both being solo singers.

Music similarity measures are usually not interpretable, e.g. see [6]. This may be related to a system-centric research protocol, which is common in the field, and consists of laboratory experiments conducted without end-user involvement, e.g. the evaluation of algorithms on digital databases [7]. This protocol is pragmatic but it may have contributed to widening the “semantic gap” in music similarity measures, which sets a bound on user satisfaction [1]. The term “semantic gap” refers to the discrepancy between high-level perception of similarity by humans and the low-level numerical data used by algorithms [8]. For example, a user might perceive two songs as similar because of their lyrics, while an algorithm might be limited to using abstract statistical descriptors operating on the audio signals. One way of alleviating the semantic gap is the introduction of mid-level features. Mid-level features either combine low-level features or extend them to ones that incorporate additional, higher-level knowledge [1]. Another way of reducing the gap is the adoption of interpretable similarity measures. An interpretable similarity measure can narrow the gap between the low perceptual level of an algorithm, which makes a decision about similarity, and the high perceptual level of a human, by providing an explanation for the human to understand the decision.

The remainder of the paper is organised as follows: in Section 2, we frame our similarity measure in the literature of the subject; in Section 3, we formalise our similarity measure; and, in Section 4, we validate our similarity measure in four experimental conditions. The source code supporting this study is freely available.¹

2. RELATED WORK

The concept of similarity is central to Music Information Retrieval. Over the years, researchers have proposed many ways of measuring music similarity, which can be categorised based on the data they use. We might, for example, categorise them into two: content-based approaches that use the audio signal; and context-based approaches that use information about artists and the pieces of music themselves, e.g. their lyrics, listening logs, and so on [9]. We refer the reader to the book by Knees & Schedl [6] for an extensive review of content-based and context-based approaches. The similarity measure that we propose in this paper is context-based.

Some of the context-based approaches make use of knowledge graphs. These approaches represent entities from the music domain and information about those entities as nodes in a graph, and they use graph structure to gauge similarity. (We define knowledge graphs more formally in Section 3.1.)

Many knowledge graph-based approaches to the measurement of similarity are path-based. They use the existence of paths between the seed and the target entities to determine similarity. Path-based algorithms usually score paths based on some criteria, and then aggregate path scores to determine similarity. For example, in [10] Leal et al. describe Shakti. Shakti scores each path by summing hand-crafted node weights and edge weights, and then dividing by the cube of the path length. Shakti only considers paths whose length falls below a fixed threshold. In [11], Strobin & Niewiadomski extend Shakti by setting node and edge weights using a genetic algorithm. This extension evolves different sets of weights indexed by path length, and considers paths of any length. They adjust node weights by considering centrality, i.e. the number of in-going and out-going edges of a node.

Passant proposes another path-based algorithm, called LDSO [12]. LDSO considers only paths of length up to two. It assigns low scores to paths whose type is common in the context of the seed and target entities. Piao & Breslin extend LDSO in [13]. They propose four variants of LDSO, which correct the path scoring mechanism with global information, such as the frequency of path types in the whole knowledge graph. In a comparative study, Piao et al. show that LDSO largely outperforms Shakti in accuracy [14].

The similarity measure that we propose in this paper is also path-based. It is different from the works in the literature as we employ a novel scoring mechanism for paths, based on their interestingness.

As well as path-based approaches to the measurement of similarity, there are also embedding-based approaches. These work by representing the knowledge graph in a dense, low-dimensional feature space, which tries to preserve as much of the graph’s structural information as possible. There are several ways to learn knowledge-graph embeddings. We refer the reader to [15] for a survey. Once the embeddings are learned, the similarity between entities can be computed on their vector representations using, for example, cosine similarity.

Path-based and embedding-based approaches to the measurement of similarity differ in their interpretability. In [16], Du et al. provide a characterization of approaches based on their potential for interpretability. They introduce the concept of *intrinsic interpretability*, which is achieved by constructing self-explanatory models. Path-based approaches typically exhibit intrinsic interpretability, since the paths used to gauge similarity can be considered to be self-explanatory. Embedding-based approaches do not exhibit intrinsic interpretability, since the embeddings used to gauge similarity cannot be considered to be self-explanatory. As a consequence, path-based approaches are interpretable after the translation of the paths to natural language, while embedding-based approaches can become interpretable only after the introduction of a post-hoc model, built to generate explanations [16].

Path-based approaches are not the only category of similarity measures to feature intrinsic interpretability. For ex-

¹ <https://github.com/GiovanniGabbolini/ipsim>

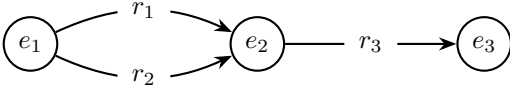


Figure 1. A simple knowledge graph.

ample, if items are described by sets of tags, then Jaccard similarity over tags also exhibits intrinsic interpretability. The similarity can be explained by showing the tags shared between the seed and the target. Even so, we find a scarcity of interpretable similarity measures in the literature; the book [6] contains very few, for example.

We mention that Passant’s LDS is also interpretable, since Passant implements a mechanism to translate paths to natural language [12].

3. METHOD

As we have already indicated, our approach to the measurement of similarity of entities in knowledge graphs is path-based. It builds on work described in [4], where the authors introduce interestingness as a way of distinguishing more interesting from less interesting paths in a knowledge graph.

In the following, we first introduce some preliminary concepts, then we review [4]’s definition of interestingness, and finally we introduce our similarity measure.

3.1 Preliminaries

Both our work and [4] make use of a knowledge graph G as an abstract representation for items and information about those items. In the experiments that we report later in this paper, items are artists; in [4], items are songs. But the approach is domain-independent: items can be any entities of interest.

A **knowledge graph** is a set of triples $G = \{(e, r, e') \mid e, e' \in E, r \in R\}$, where E and R denote, respectively, the sets of entities (nodes) and relationships (edges). For example, the knowledge graph of Figure 1 contains three triples, $\{(e_1, r_1, e_2), (e_1, r_2, e_2), (e_2, r_3, e_3)\}$. A special subset of entities $I \subseteq E$, are the items. Every entity has a *type* and a *value*. For example, if an entity represents an artist, then its type is “artist” and its value is the URI of that artist. Every edge (relationship) has a *type* also. For example, an edge that connects two artists who have collaborated will have “collaborated with” as its type.

A **path** p in G is an ordered list of entities and edges in G , $p = [e_1, r_1, \dots, r_{n-1}, e_n]$ where each triple in p must be in G . For example, with reference to the knowledge graph of Figure 1: $[e_1, r_1, e_2, r_3, e_3]$ is a path; $[e_1, r_1, e_2, r_1, e_3]$ is not a path, as the triple (e_2, r_1, e_3) is not in the knowledge graph. The *type* of p is the ordered concatenation of the entity and edge types in p .

The function $paths(i_1, i_2)$ finds the paths in G from the item $i_1 \in I$ to the item $i_2 \in I$, without visiting any other item in I and without cycles. In other words, if items are songs, for example, then the function finds paths between pairs of songs, where the paths are not allowed to contain any intermediate songs.

3.2 Interestingness

In [4], the interestingness of a path in a knowledge graph is defined as a weighted combination of three heuristics:

Rarity Let T be the set of all path types in G ; and let $f(t)$ be the number of paths in G that are of type t . The *rarity* of a path p is defined as:

$$rarity(p) = 1 - \frac{f(type(p))}{\max_{t \in T} f(t)}$$

Unpopularity Let $edgeset(e)$ be the set of in-going and out-going edges to and from an entity $e \in E$ in G . The centrality of an entity e is:

$$centrality(e) = \min \left(1, \frac{|edgeset(e)|}{\text{median}_{e' \in E} |edgeset(e')|} \right),$$

given $type(e') = type(e)$

The *unpopularity* of a path p is defined as:

$$unpopularity(p) = 1 - \min_{e \in p \cap E} (centrality(e))$$

Shortness If we define $length(p)$ as the number of edges in path p , then the *shortness* of a path is given by:

$$shortness(p) = \frac{1}{length(p)}$$

The *interestingness* of a path p in G is given by:

$$interestingness(p) = w_1 rarity(p) + w_2 unpopularity(p) + w_3 shortness(p)$$

Its values range from zero to one. w_1, w_2, w_3 are parameters to be tuned, subject to $w_1 + w_2 + w_3 = 1$.

In [4], the authors run a user trial where participants are asked to evaluate segues. They find that *interestingness* positively correlates with human perceptions of quality: *interestingness* is high for segues perceived as high-quality, and vice versa. This lends credence to the idea of using their definition of interestingness in the similarity measure that we are proposing in this paper.

3.3 Similarity

We define the similarity between two items $i_1 \in I$ and $i_2 \in I$ as an aggregate of the interestingness of the paths between them:

$$sim(i_1, i_2) = \sum_{p \in paths(i_1, i_2)} interestingness(p) \mathcal{I}(p, n)$$

where \mathcal{I} is an indicator function: $\mathcal{I}(p, n) = 1$ if $\text{length}(p) \leq n$, and 0 otherwise. n is a parameter whose value determines whether the aggregation restricts attention to shorter paths or whether it considers all paths ($n = \infty$). It is not clear *a priori* whether it is beneficial to put a limit on path length and, if so, what limit to use. For example, [12] use only paths of length up to three, while [11] uses all paths. In our case, we leave the choice open by introducing the parameter n to be tuned. The interestingness weights w_1, w_2, w_3 are also parameters to be tuned.

In the following, we refer to our similarity measure as **IPSIM (Interesting Path Similarity)**. IPSIM is domain-independent. In the rest of this paper, we focus on the music domain, and in particular the case where items are music artists. The reason we focus on artist similarity is that it is a common choice in the literature and there is an amount of data to use in offline experiments.

4. EXPERIMENTS

We provide an evaluation for IPSIM in the case of music artists. In this evaluation, we are particularly interested in evaluating the accuracy that IPSIM is able to achieve.

4.1 Knowledge graph

We represent an artist with their MusicBrainz URI.² This representation can be easily changed with only minor modifications to the rest of our implementation, e.g. if artists were instead represented by their Spotify URIs.

Our implementation uses a knowledge graph with 30 distinct node types and 205 distinct edge types. It is built with data that we harvest from two resources:

MusicBrainz We use MusicBrainz as the main source of factual data. We exploit the MusicBrainz APIs.³ They allow us to navigate the MusicBrainz database, and offer entity-linking functionalities. We mine different sorts of factual data, ranging from the birth places of the artists to the genres that they play.

Wikidata We use Wikidata as an additional source of factual data.⁴ There exists a mapping from MusicBrainz URIs to Wikidata URIs, making it easy to use both resources. From Wikidata, we mine biographical data about artists that is not available in MusicBrainz, e.g. the awards that an artist has won.

We provide a complete description of entities and relationships that build up the knowledge graph that we use in this paper in the additional materials.⁵

4.2 Experiment design

As highlighted by Knees & Schedl [1], there is no agreement upon the best method to evaluate the quality of mea-

asures of music similarity. Every method has its own advantages and disadvantages. In this paper, we assess music similarity measures using two common experimental settings. The first uses a similar-artists ground-truth, and the second uses user-artist interaction histories.

4.2.1 Using a ground-truth

This evaluation procedure replicates the one in [17]. The procedure uses datasets that are composed of tuples of the form (*seed artist, similar-artists ground-truth*). In other words, for each seed artist, it gives a list of artists that have been independently judged to be similar to the seed, these artists being regarded as the ground-truth. We employ two datasets:

MIREX: This dataset comprises 188 seed artists. The ground truth is based on similarity judgements expressed by experts during the MIREX Audio Music Similarity and Retrieval Task.

LastFM-g: This dataset comprises 2336 seed artists. The ground truth is based on similarity judgements gathered from the Last.fm APIs.⁶

See [17] for additional details on these datasets.

To evaluate a similarity measure on a given dataset, we do the following. For every seed artist, we use the similarity measure to score the other artists and then to rank them by decreasing similarity with the seed. Then, we compare this ranking @ N with the ground-truth from the dataset, according to the standard accuracy metrics *nDCG* and *Precision*, or *Pr* for short.

4.2.2 Using interaction histories

This protocol uses a dataset that records how users have interacted with artists. We denote with U, I the sets of users and artists resp. and with $|U|, |I|$ their cardinalities. Lower case letters u, i refer to $u \in U, i \in I$. In this case, the dataset is organised as a matrix $R \in \mathbb{R}^{|U| \times |I|}$. Each cell $r_{ui} > 0$ of R accounts for a user-artist interaction. If a user u is not known to have interacted with an artist i , then r_{ui} is zero.

We employ two datasets of this kind:

LastFM-h: In this Last.fm dataset (which is different from the one described earlier), r_{ui} is the listening count by user u of artist i [18].⁷ We filter out users who have listened to fewer than five artists, and artists listened to by fewer than five users. We end up with 1877 users and 2828 artists. We convert listening counts to ratings on a 1–5 scale following the procedure given in [19]. For every user, we hold-out at random 40% of her interactions, and we keep the other interactions as training data. We further split the held-out interactions into two equal-sized parts, to form validation and test data. Finally, we remove the interactions where $r_{ui} < 4$ from the validation and test data so that they only contain artists that the users like.

² <https://musicbrainz.org/>

³ <https://python-musicbrainzngs.readthedocs.io/en/v0.7.1/>

⁴ <https://wikidata.org/>

⁵ <https://doi.org/10.5281/zenodo.5121460>

⁶ last.fm

⁷ <https://grouplens.org/datasets/hetrec-2011/>

Facebook: This dataset, from the Second Linked Open Data-enabled Recommender Systems Challenge, contains items liked by users in three domains: movies, books and music.⁸ We focus on music artists, by keeping only items of types: *music_artist* and *music_band*. We filter out artists liked by fewer than five users, and those whose MusicBrainz URI cannot be resolved. We end up with 52069 users and 4435 artists. In this case, $r_{ui} = 1$ if the user u likes the artist i . For every user, we hold-out 40% of her interactions at random, and we keep the others as training data. We further split the held-out interactions into two equal-sized parts, to form validation and test data.

For these two datasets, the evaluation procedure uses the similarity measure as a core component within a recommender system. The design of the recommender system is inspired by [13], chosen because of its heavy reliance on similarity, which is what we are trying to evaluate. We define the user profile of a user u as the artists she has rated (in the training set):

$$P(u) = \{j | r_{uj} \neq 0\}.$$

Let $NN(k, i, P(u))$ be the set of k most similar items in $P(u)$ to an artist i . The predicted relevance score for a user u and an artist i , indicated as $\overline{r_{ui}}$, is computed as:

$$\overline{r_{ui}} = \begin{cases} 0 & \text{if } i \in P(u) \\ \sum_{j \in NN(k, i, P(u))} r_{uj} \text{ sim}(i, j) & \text{if } i \notin P(u) \end{cases}$$

For every user, we compute the relevance score $\overline{r_{ui}}$ for every artist $i \in I$, and rank the artists by decreasing score. Then, we compare this ranking @ N and the held-out test data (or validation data, as appropriate), according to the standard accuracy metrics *nDCG* and *Precision*, or *Pr* for short. We take higher recommendation accuracy to be an indication of a higher quality similarity measure.

4.3 Baseline similarity measures

We include in our experiments three baselines. The first is a random algorithm (RND), useful to set a lower bound on the performance. It rates the similarity between two items to be a random number between zero and one. The other two (COUNT and LDSD) are path-based algorithms. By comparing our path-based approach to other path-based similarities, we reduce the number of confounders, so we can investigate whether our idea of using interestingness as a scoring mechanism for paths is a good one. In fact, the path-based similarity measures that we use in this experiment all use the same knowledge graph and the same paths, and differ only in the way the paths are scored and how the scores are aggregated. COUNT is a simple path-based algorithm that works by counting the number of paths between items. It is equivalent to substituting 1 for the argument to the sum in the definition of *sim* given earlier. LDSD is an accurate path-based algorithm that proposes a carefully-designed weighting heuristic for paths [12].

⁸<https://lists.w3.org/Archives/Public/public-vocabs/2015Feb/0046.html>

	w_1	w_2	w_3	n	k
<i>MIREX</i>	0.3	0.1	0.6	2	NA
<i>LastFM-g</i>	0.0	0.1	0.9	2	NA
<i>LastFM-h</i>	0.0	0.0	1.0	2	40
<i>Facebook</i>	0.9	0.0	0.1	∞	∞

Table 2. Optimal parameters found for IPSIM.

4.4 Parameter tuning

IPSIM has some parameters, as indicated in Section 3.3. Additionally, the experiment using listening histories introduces another parameter, i.e. the number of neighbours k used by the recommender system, as described in Section 4.2.2. We set the parameters with a grid-search, optimizing *nDCG@10* on the validation data. For the experiment using a ground-truth, we tune the parameters of IPSIM. For the experiment using listening histories, we tune the parameters of IPSIM and we tune k for each of RND, COUNT, LDSD and IPSIM. We report the optimal parameters found for IPSIM in Table 2. The optimal values of the parameter k for RND, COUNT and LDSD are, respectively, 5, 10 & 40 in *LastFM-h* and 1, 1 & 7 in *Facebook*.

The optimal parameter configurations change from dataset to dataset. This might be due to the different nature of the datasets. *MIREX* features human judgements of artist similarity; the algorithmic judgments to be found in *LastFM-g* might not completely agree with these. Schedl et al. [7] suggest that human perception of similarity is influenced by user factors, such as user context (e.g. mood) and user properties (e.g. musical training). User factors might have directly influenced the people who annotated *MIREX*, but not the algorithm used to construct *LastFM-g*. The remaining two datasets, *LastFM-h* and *Facebook*, gather another kind of data, i.e. interaction histories, but they are gathered in different ways. In *LastFM-h*, unless the user intervenes, the next recommended song is played automatically, whereas, in *Facebook*, users are interacting with artist fan pages in what is usually a more deliberate, conscious and considered way.

In three of the four datasets, at least one of the three interestingness heuristics of IPSIM receives a weight of zero. We may find an explanation for this in the median number of paths between artists: 19 in *MIREX*, 13 in *LastFM-g*, and seven in both *LastFM-h* and *Facebook*. *MIREX* is the only dataset where all three weights are non-zero, and is also the dataset with the highest median number of paths between artists. In general, the use of a more complex model (with non-zero weights for each component) might be beneficial only when the dataset refers to artists in the knowledge graphs for whom there exists a rich variety of paths, e.g. when considering popular artists.

4.5 Results

We conduct the experiments described in Section 4.2. We compare IPSIM against the baselines described in Section 4.3 (RND, COUNT, LDSD). We set the parameters of the similarity measures and the recommender as described in

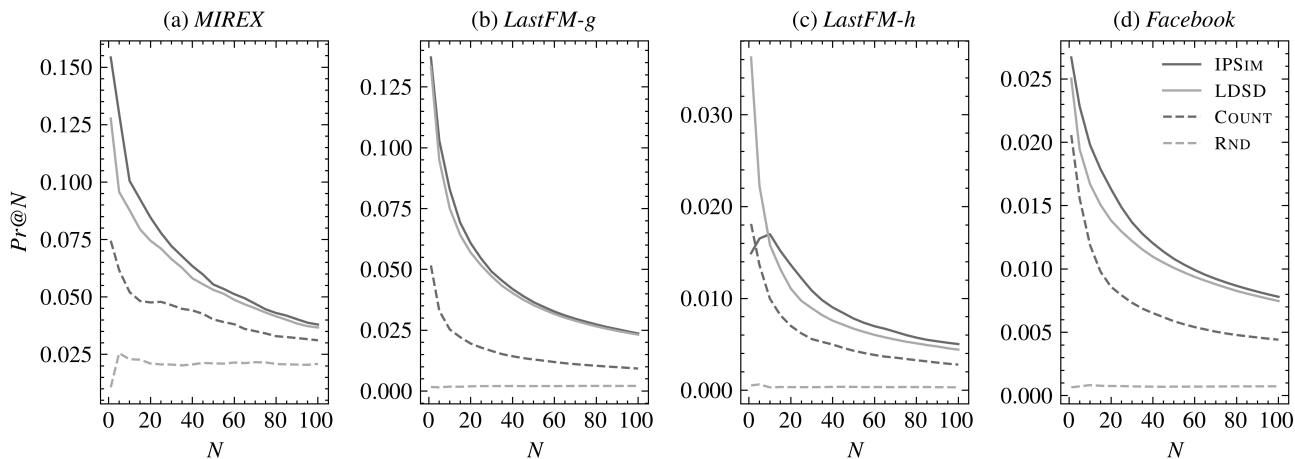


Figure 2. Accuracy of similarity measures on the *MIREX*, *LastFM-g*, *LastFM-h* and *Facebook* datasets, as measured by *Precision* at cutoffs from one to 100.

	<i>nDCG@5</i>	<i>nDCG@10</i>	<i>Pr@5</i>	<i>Pr@10</i>
RND	0.023	0.022	0.026	0.023
COUNT	0.065	0.057	0.062	0.052
LDSD	0.102	0.094	0.096	0.088
IPSIM	0.136***	0.114*	0.130***	0.101*

Table 3. Accuracy of similarity measures on the *MIREX* dataset, as measured by *nDCG* and *Precision* at cutoffs five and ten. *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$.

	<i>nDCG@5</i>	<i>nDCG@10</i>	<i>Pr@5</i>	<i>Pr@10</i>
RND	0.002	0.002	0.002	0.002
COUNT	0.037	0.030	0.033	0.025
LDSD	0.103	0.086	0.095	0.075
IPSIM	0.111**	0.094***	0.103**	0.083***

Table 4. Accuracy of similarity measures on the *LastFM-g* dataset, as measured by *nDCG* and *Precision* at cutoffs five and ten. *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$.

	<i>nDCG@5</i>	<i>nDCG@10</i>	<i>Pr@5</i>	<i>Pr@10</i>
RND	0.001	0.000	0.001	0.000
COUNT	0.015	0.012	0.014	0.010
LDSD	0.025***	0.020**	0.022***	0.016
IPSIM	0.016	0.017	0.017	0.017

Table 5. Accuracy of similarity measures on the *LastFM-h* dataset, as measured by *nDCG* and *Precision* at cutoffs five and ten. *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$.

	<i>nDCG@5</i>	<i>nDCG@10</i>	<i>Pr@5</i>	<i>Pr@10</i>
RND	0.001	0.001	0.001	0.001
COUNT	0.017	0.014	0.016	0.012
LDSD	0.021	0.018	0.019	0.017
IPSIM	0.024***	0.021***	0.023***	0.020***

Table 6. Accuracy of similarity measures on the *Facebook* dataset, as measured by *nDCG* and *Precision* at cutoffs five and ten. *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$.

Section 4.4 using the validation data. We report the results on the test data. We verify the significance of differences between IPSIM and LDSD with Wilcoxon signed-rank test.

Tables 3 & 4 report on the experiment that uses a ground-truth. IPSIM outperforms all the baselines in both the *MIREX* and *LastFM-g* datasets. We notice that IPSIM scores accuracy at least double that of COUNT in both datasets. The increase in performance with respect to LDSD is statistically significant in both datasets.

Tables 5 & 6 report on the experiment that uses listening histories. IPSIM is always more accurate than COUNT. IPSIM always outperforms LDSD in *Facebook* but outperforms LDSD in only one case in *LastFM-g*, and the difference in this case is not statistically significant.

We investigate more deeply how accuracy varies with the cutoff in Figure 2. For the four datasets, the figure shows the *Precision* of the similarity measures as a function of the cutoff. The figure confirms that IPSIM has higher precision than LDSD on *LastFM-h* for cutoffs greater than ten, but the precision of IPSIM drops for cutoffs < 10 . We also notice that, on *MIREX*, *LastFM-g* and *Facebook*, IPSIM outperforms LDSD for every cutoff ranging from one to 100.

The results provide evidence that interestingness can be the basis of an accurate similarity measure.

5. CONCLUSIONS AND FUTURE WORK

The results of the experiments highlight the validity of our approach to music similarity (at least in the case of music artists), and demonstrate that the interestingness scores of [4] can be the basis of an accurate and interpretable similarity measure. Future work might include a comparison of the performance of our similarity measure against an even wider range of music similarity measures. We are also interested in using the proposed similarity measure in a related-item recommender system, that can meaningfully guide users in their exploration of the items by means of the interpretability of the similarity measure.

6. ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 12/RC/2289-P2 which is co-funded under the European Regional Development Fund. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

7. REFERENCES

- [1] P. Knees and M. Schedl, "Introduction to music similarity and retrieval," in *Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 1–30. [Online]. Available: https://doi.org/10.1007/978-3-662-49722-7_1
- [2] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.
- [3] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 615–620. [Online]. Available: <https://www.aclweb.org/anthology/D14-1067>
- [4] G. Gabbolini and D. Bridge, "Generating interesting song-to-song segues with dave," in *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 98–107. [Online]. Available: <https://doi.org/10.1145/3450613.3456819>
- [5] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0004370218305988>
- [6] P. Knees and M. Schedl, in *Music Similarity and Retrieval: An Introduction to Audio- and Web-based Strategies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016. [Online]. Available: <https://doi.org/10.1007/978-3-662-49722-7>
- [7] M. Schedl, A. Flexer, and J. Urbano, "The neglected user in music information retrieval research," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 523–539, 2013. [Online]. Available: <https://doi.org/10.1007/s10844-013-0247-6>
- [8] D. Ponceleón and M. Slaney, "Multimedia information retrieval," in *Modern Information Retrieval: The Concepts and Technology Behind Search*. Addison-Wesley Professional, 2011, pp. 589–590.
- [9] P. Knees and M. Schedl, "A survey of music similarity and recommendation from music context data," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 10, no. 1, Dec. 2013. [Online]. Available: <https://doi.org/10.1145/2542205.2542206>
- [10] J. Leal, V. Rodrigues, and R. Queirós, "Computing semantic relatedness using DBPedia," *Symposium on Languages, Applications and Technologies, 1st*, no. January, pp. 133–147, 2012.
- [11] Ł. Strobin and A. Niewiadomski, "Evaluating semantic similarity with a new method of path analysis in RDF using genetic algorithms," *Journal of Applied Computer Science*, vol. 21, no. 2, pp. 137–152, 2013.
- [12] A. Passant, "Measuring semantic distance on linking data and using it for resources recommendations," *AAAI Spring Symposium - Technical Report*, vol. SS-10-07, pp. 93–98, 2010.
- [13] G. Piao and J. G. Breslin, "Measuring semantic distance for linked open data-enabled recommender systems," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, ser. SAC '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 315–320. [Online]. Available: <https://doi.org/10.1145/2851613.2851839>
- [14] G. Piao, S. S. Ara, and J. G. Breslin, "Computing the semantic similarity of resources in dbpedia for recommendation purposes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9544, pp. 185–200, 2016.
- [15] Y. Dai, S. Wang, N. N. Xiong, and W. Guo, "A survey on knowledge graph embedding: Approaches, applications and benchmarks," *Electronics*, vol. 9, no. 5, 2020. [Online]. Available: <https://www.mdpi.com/2079-9292/9/5/750>
- [16] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, p. 68–77, Dec. 2019. [Online]. Available: <https://doi.org/10.1145/3359786>
- [17] S. Oramas, M. Sordo, L. Espinosa-Anke, and X. Serra, "A semantic-based approach for artist similarity," in *16th International Society for Music Information Retrieval Conference*, Málaga, Spain, 26/10/2015 2015, pp. 100–106. [Online]. Available: <http://hdl.handle.net/10230/26278>
- [18] I. Cantador, P. Brusilovsky, and T. Kuflik, "2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011)," in *Proceedings of the 5th ACM conference on Recommender systems*, ser. RecSys 2011. New York, NY, USA: ACM, 2011.
- [19] Ò. Celma, "Music recommendation and discovery in the long tail," Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, 2008.

LEVERAGING HIERARCHICAL STRUCTURES FOR FEW-SHOT MUSICAL INSTRUMENT RECOGNITION

Hugo Flores Garcia, Aldo Aguilar, Ethan Manilow, Bryan Pardo

{hugofg@u., aldoa@u., ethanm@u., pardo}@northwestern.edu

Interactive Audio Lab, Northwestern University, Evanston, IL, USA

ABSTRACT

Deep learning work on musical instrument recognition has generally focused on instrument classes for which we have abundant data. In this work, we exploit hierarchical relationships between instruments in a few-shot learning setup to enable classification of a wider set of musical instruments, given a few examples at inference. We apply a hierarchical loss function to the training of prototypical networks, combined with a method to aggregate prototypes hierarchically, mirroring the structure of a predefined musical instrument hierarchy. These extensions require no changes to the network architecture and new levels can be easily added or removed. Compared to a non-hierarchical few-shot baseline, our method leads to a significant increase in classification accuracy and significant decrease in mistake severity on instrument classes unseen in training.

1. INTRODUCTION

Musical instrument recognition is a machine learning task that aims to label audio recordings of musical instruments, typically at a fine temporal granularity (second by second) [1–3]. Musical instrument recognition can be viewed as a subtask of Sound Event Detection (SED), which consists of identifying and locating any type of sound event (e.g., car horn, dog bark) in an audio recording [4–6].

Labelling audio tracks is extremely important for organizing the dozens of tracks in a typical Digital Audio Workstation (DAW) recording session [7,8], but manual labelling is a tedious process. Automated musical instrument recognition could enable automated track labeling. Automated second-by-second labeling could go further, enabling navigation through recording projects by traversing musical instrument *labels*, rather than waveform visualizations. This would be especially helpful for audio engineers with low or no vision, as existing interfaces leave accessibility as an afterthought [9] and navigating by visually examining waveforms is not a viable option for them [10].

A barrier to incorporating instrument recognition into DAWs is that most existing deep learning techniques must

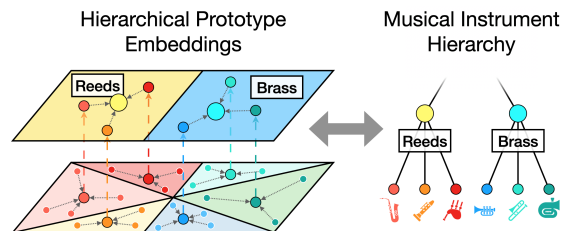


Figure 1. Overview of our method. Prototypes from a set of embedded support examples at a fine-grained level (bottom left) are aggregated to make a set of *metaprototypes* at a coarser-grained level (top left). In this way, we learn a hierarchical set prototypes that corresponds to a musical instrument hierarchy (right).

be trained on instruments that have abundant labeled training data. The datasets that support these systems only focus on the limited set of instrument classes that have sufficient data [11–17]. However, the vast diversity of musical instrument sounds necessitates supporting a broader set of instrument classes [18]. While expanding current datasets with more diverse coverage can ameliorate this issue, collecting human annotations for a large number of audio files is a tedious, time consuming task [19, 20], and there will always be unanticipated sound categories that an end-user would like to automatically label.

Therefore, musical instrument recognition systems should be able to dynamically expand their vocabularies after deployment, to conform to end-user needs. This requires an approach that lets a system learn a new sound category given only a few examples that can be provided by an end user, *a la* few-shot learning.

Using a hierarchical system, like the widely-used Hornbostel-Sachs hierarchy [21], to organize and classify musical instruments has broad precedent in many human cultures [22]. We can take advantage of a musical instrument hierarchy, like the widely-used Hornbostel-Sachs hierarchy [21], to improve few-shot learning. A system could learn a feature space meaningful for unseen classes that share hierarchical ancestry with the classes seen during training. For example, the Chinese zhongruan is a plucked string instrument that shares ancestry with other chordophones in the Hornbostel-Sachs hierarchy (like the guitar), which might be more common in datasets of Western instruments. A model could leverage the hierarchical relationship between an instrument it has never been trained on (e.g. the zhongruan) and more common instru-



ments seen during training (*e.g.* the guitar) to produce a meaningful representation of the new instrument with only a few support examples.

In this work, we propose a simple extension to prototypical networks [23] that imposes a hierarchical structure on the learned embedding space (Figure 1). We first create prototypes from an initial set of embedded support examples at the most granular level. We then aggregate these initial prototypes into new prototypes corresponding to a coarser hierarchical level, in a manner reminiscent of agglomerative clustering [24]. Repeating this process lets our system represent classes at many granularities of a predefined instrument hierarchy. We also propose a weighted, hierarchical extension of cross-entropy loss to ensure the network learns the hierarchy. Compared to a non-hierarchical few-shot baseline [25], our method shows a significant increase in classification accuracy and significant decrease in mistake severity on unseen instrument classes.

2. RELATED WORK

Musical instrument recognition can be performed in single-source contexts [26–29], where only a single sound source may be active at any given time, as well as in multi-source contexts [13–15, 30, 31], where multiple sound sources may be active at the same time. We consider the single-source case, as the vast majority of audio in a studio music production workflow is single-source.

Hierarchical structures have shown to be effective for many machine learning tasks, such as text classification [32] and image classification [33, 34]. In fact, Bertinetto *et al.* [35] propose a hierarchical image classification approach that uses a similar exponentially weighed hierarchical loss function to the one proposed here, although they do not focus on a few-shot setting, as we do, and they favor learning broader classes, whereas we are also interested in finer classes. Hierarchical structure was explored for musical instrument recognition by using fixed signal processing feature extraction techniques [29, 36, 37]. Here, we use deep learning methods to flexibly learn a feature space that mirrors musical instrument hierarchies.

Recent work has studied how hierarchical structures can be incorporated into neural network models for different tasks. In the automatic speech recognition (ASR) domain, CTC-based hierarchical ASR models [38–40] employ hierarchical multitask learning techniques, particularly by using intermediate representations output by the model to perform intermediate predictions in a coarse-to-fine scheme. Manilow *et al.* [41] have shown that hierarchical priors can have significant benefits for performing source separation of musical mixtures. None of these systems, however, were designed for few-shot learning.

Previous deep learning systems have been proposed for multilevel audio classification [42–44]. However, none of these systems work in a few-shot setting and they require either specialized network architectures or complex data pipelines to learn a hierarchy. Our approach is a simple extension to incorporate hierarchy into an established few-shot learning paradigm.

Recent work in audio tagging and sound event detection tasks has explored few-shot learning in the audio domain [19, 25, 45–47], though none of this work assumed any hierarchical structure.

Here, we propose a method for hierarchical representation learning in a few-shot setting, leveraging the increased flexibility of both hierarchy and few-shot methods for musical instrument recognition.

3. BACKGROUND

3.1 Few-shot Learning

In a few-shot classification setting, we consider a target class $k \in \mathcal{K}$ for a set of target classes, \mathcal{K} , of size $|\mathcal{K}|$. Let x_s be a single support example drawn from a set of examples \mathcal{S} , called the support set. Assume N labeled support examples (*i.e.*, shots) per class k , totalling $N \times |\mathcal{K}|$ labeled examples. We define \mathcal{S}_k as the subset of \mathcal{S} containing the examples of class k .

We are provided an unlabeled query set \mathcal{Q} of M unlabeled examples. The goal of the task is to label each query example $x_q \in \mathcal{Q}$ with a target class $k \in \mathcal{K}$. A neural network model f_θ projects both the support and query sets into a discriminative embedding space. The query is assigned to the class of the support set it is closest to, according to distance metric d .

3.2 Prototypical Networks

Prototypical networks [23] compute an embedding vector for each instance in \mathcal{S}_k . The prototype, c_k , for class k is the mean vector of all the support embeddings belonging to class k :

$$c_k = \frac{1}{|\mathcal{S}_k|} \sum_{x_s \in \mathcal{S}_k} f_\theta(x_s). \quad (1)$$

Using a distance function d , we can produce a probability distribution over the set of classes \mathcal{K} for a given query x_q by applying a softmax over the negated distances from the query to each class prototype:

$$p(\hat{y}_q = k | x_q) = \frac{\exp(-d(f_\theta(x_q), c_k))}{\sum_{c'_k} \exp(-d(f_\theta(x_q), c'_k))}. \quad (2)$$

We use the Euclidean distance as d in this work.

4. METHOD

Musicologists have long categorized musical instruments into hierarchical taxonomies, such as the Hornbostel-Sachs system [21], which classifies musical instruments into a hierarchy corresponding to their sound producing mechanisms. We can improve upon existing few-shot models by leveraging the hierarchical structure intrinsic to musical instrument taxonomies. To do this, we extend prototypical networks by training on a multitask scenario composed of multiple classification tasks, one for each level of a class tree, where the prototype for a parent node in the class tree is defined as the mean of the prototypes for each of the parent node’s children.

We impose hierarchical structure on our few-shot task by constructing a tree, T , with height H , starting from a set of leaf nodes. We define the leaf nodes as the same set of classes, \mathcal{K} , that we defined for our standard few-shot setup in Sec. 3.1. We then define the parents of the leaf nodes by aggregating classes, $k \in \mathcal{K}$. For musical instrument recognition, we aggregate classes according to a predefined instrument hierarchy (e.g., Hornbostel-Sachs). We iteratively aggregate child classes up to the max height of the tree H . We index the tree as $T_{i,h}$, where $i \in \mathcal{K}_i$ indexes over the set of sibling classes at level h , for $h = 0, \dots, H$, with level 0 containing the most specific classes and level H containing the broadest. In our notation $H = 0$ describes a tree with no hierarchy and is equivalent to the non-hierarchical prototypical network defined in Sec. 3.2. $H = 1$ has two levels, and so on.

4.1 Hierarchical Prototypical Networks

We define our proposed hierarchical prototypical network by extending typical prototypical networks [23] to a hierarchical multitask learning scenario, where we wish to label each query example, $x_q \in \mathcal{Q}$, at multiple levels of our class tree, T . Here, labeling at each level is a separate task.

Like a normal prototypical network, we use a network f_θ to produce embeddings for every example in the support set. The mean of these embedded support examples creates an initial set of prototypes (Eq. 1). We deviate from the typical setup by considering this initial set of prototypes as the lowest level of our tree, T , and aggregating these initial prototypes *again* to make another set of prototypes representing the next level. The prototypes at this higher level are, thus, prototypes of prototypes, or *metaprototypes*, and define a hierarchy according to the structure of our tree, T . We continue to iteratively aggregate prototypes in this fashion for all levels of our tree. The prototype for each parent class at level $h+1$ is notated $c_{T_{i,h+1}}$ and is the mean of the members of its support set $\mathcal{S}_{T_{i,h}}$. For levels $h > 0$, each example \hat{x}_s , is itself a prototype:

$$c_{T_{i,h+1}} = \frac{1}{|\mathcal{S}_{T_{i,h}}|} \sum_{\hat{x}_s \in \mathcal{S}_{T_{i,h}}} f_\theta(\hat{x}_s), \quad (3)$$

This process is shown in Figure 1.

Given a query example x_q , we use the network to create an embedding $f_\theta(x_q)$ and measure its distance to each class prototype or metaprototype $c_{T_{i,h}}$ at a given level h . Given these distances, we output H probability distributions, one for each level in our class tree:

$$p(T_{i,h}|x_q) = \frac{\exp(-d(f_\theta(x_q), c_{T_{i,h}}))}{\sum_{c'_{T_{i,h}}} \exp(-d(f_\theta(x_q), c'_{T_{i,h}}))}. \quad (4)$$

We note that Eqs. 1 and 2 are special cases of the proposed Eqs. 3 and 4, evaluated at $h = 0$. Our generalization allows multi-task few-shot classification at multiple levels of a hierarchical class tree.

Our proposed method does not require any specific network architecture. Instead, it provides a hierarchical la-

bel structure for support examples x_s to be aggregated together, forming fine-to-coarse representations (i.e., $c_{T_{i,h}}$) that we can leverage and optimize with. This exposes the potential for a model to be trained with multiple concurrent hierarchies, a direction for future work.

4.2 Multi-Task Hierarchical Loss

We now set up a learning objective, where we minimize the cross-entropy loss between the predicted distribution and the ground truth class for each level in the class tree. The intuition behind our approach is that we can use a hierarchically structured objective to encourage our model to produce an embedding space with discriminative properties at both coarse and fine granularities, allowing some of these coarse features to generalize beyond the training set of fine grained leaf classes to their unseen siblings in the class tree. We use an exponentially decaying sum of loss terms for each level in the hierarchy [35]:

$$\mathcal{L}_{\text{hierarchical}} = \sum_{h=0}^H e^{-\alpha \cdot h} \mathcal{L}_{CE}^{(h)}, \quad (5)$$

where $\mathcal{L}_{CE}^{(h)}$ denotes the cross-entropy loss for the classification task at height h , and α is a hyperparameter that determines the decay of each loss term w.r.t height. Setting $\alpha > 0$ places more weight on finer-grained tasks, $\alpha < 0$ places more weight on coarser-grained tasks, and $\alpha = 0$ weighs all tasks equally. We note that $H = 0$ reduces to the non-hierarchical (baseline) definition of the problem, where we only optimize for the fine-grained task.

5. EXPERIMENTAL DESIGN

We evaluated our proposed hierarchical prototypical approach using a non-hierarchical prototypical method [25] as a baseline. We evaluated all models on a few-shot musical instrument recognition task, measuring standard classification metrics (F1) as well as mistake severity. We conducted ablations for class tree height, choice of class hierarchy, and proposed loss function.

5.1 Datasets

For all experiments, we trained and evaluated using isolated tracks from the MedleyDB [48] and MedleyDB 2.0 [49] datasets. MedleyDB contains multi-track recordings of musical instruments and vocals. We excluded recordings that do not have fine-grained instrument labels (e.g., "brass" was excluded because the audio could be of trumpets, trombones, etc.). Additionally, we considered sections of a single instrument to be the same class as the instrument itself (e.g. "violin section" and "violin" both belong to the class "violin"). Altogether, the dataset consists of 63 different instruments, with 790 tracks in total.

For training and evaluation, we removed the silent regions of each audio track. We then split the remainder of the track into 1 second segments with a hop size of 0.5 seconds, where each 1 second segment is an input example to the model. All audio was downsampled to 16kHz. For

each example, we compute a 128-bin log-Mel spectrogram with a 32ms window and an 8ms hop. After preprocessing, our training and evaluation datasets contained 539k and 56k 1-second examples, respectively. We performed silence removal using `pysox` [50].

5.2 Network Architecture

The backbone network architecture used in all experiments was based on the prototypical network described in Wang *et al.* [47]. It uses a log-Mel spectrogram as input, and consists of four CNN blocks, where each convolutional filter has a kernel size of 3×3 , followed by a batch normalization layer, a ReLU activation, and a 2×2 maxpooling layer. After the last convolutional block, we applied maxpooling over the time dimension, to obtain a 1024-dimensional embedding. Finally, we added a linear projection layer that reduces the 1024-dimensional embedding to 128 dimensions.

5.3 Hornbostel-Sachs Class Tree

We used a musical instrument hierarchy inspired by the Hornbostel-Sachs [21] taxonomy,¹ (maximum height of 4) which is organized by the sound production mechanisms of each instrument. Since similar sound production mechanisms can lead to similar sounds, we believe this is a natural organization that our model can leverage to learn discriminative features at different levels of a class hierarchy.

5.4 Episodic Training and Evaluation

We have a musical instrument hierarchy tree, where individual instrument classes are leaf nodes (e.g. violin, guitar). Nodes at higher levels ($h > 0$) are instrument families, (e.g. bowed strings, plucked strings). Our goal is to observe classification performance on previously-unseen leaf classes (e.g. zhongruan, erhu). Therefore, we created a data split of 70% train, 30% evaluation, with no overlap between train and evaluation classes at the leaf instrument level ($h = 0$). We further added the constraint that the classes in both testing and evaluation sets be distributed evenly among the instrument families ($h > 0$). This avoids a problem where, for example, the train set consists only of percussion and the evaluation set consists only of chordophones. All experiments shared a train/evaluation split.

For each experiment, we trained every model in a few-shot learning scenario using episodic training. Each model was presented with a unique $|\mathcal{K}|$ -way, N -shot learning task (an episode) with M queries per leaf class at each training step. We constructed an episode by sampling a set of $|\mathcal{K}|$ instrument classes from the training data. For each of these $|\mathcal{K}|$ classes, we sampled $N + M$ audio examples. Here, for each class k , $N = |S_k|$ is the number of "shots" in the support set and M is the size of the query set.

We trained all models using the same random initialization for a maximum of 60,000 steps with early stopping after the evaluation loss stopped improving for 4500 steps,

using the Adam optimizer and a learning rate of 0.03. During training, we set $|\mathcal{K}| = 12$, $N = 4$, and $M = 12$. We evaluated each trained model on episodes constructed from the test data. For each evaluation, we made 100 episodes, with $|\mathcal{K}| = 12$, $M = 120$. All hyperparameters were fixed except those we ablated, as described below.

5.5 Evaluation Metrics

We used the F1-score as our primary classification metric, reporting the distribution of F1 scores computed for each episode, evaluated for predictions made at the finest level of the hierarchy.

Similar to Bertinetto *et al.* [35], we used the hierarchical distance of a mistake as a metric indicative of a model's mistake severity. Given a class tree, the hierarchical distance of mistake is defined as the height of the lowest common ancestor (LCA) between the prediction node and ground truth node when the input is misclassified (that is, when the model makes a mistake). We report the average hierarchical distance of a mistake over all evaluation episodes.

For all hierarchical models, we measured mistake severity with respect to its own hierarchy. For the non-hierarchical model, we evaluated with respect to our proposed 4-level version of the Hornbostel-Sachs hierarchy, as we believe that its organization is meaningful.

6. EXPERIMENTS

We now describe specific experiments to measure the effects of different design choices. We trained and evaluated all models using the procedure described in Section 5. Our experiment code is available online².

6.1 Tree Height

To observe the effect of tree height on classification, we constructed shorter trees from the Hornbostel-Sachs class tree by removing every leaf node's parent until the desired max height of the tree is met. We trained and evaluated five models using our proposed class tree, shortened to different heights $H \in \{0, 1, 2, 3, 4\}$, where $H = 0$ is the baseline, non-hierarchical case inspired by Wang *et al.* [25]. Each model was trained with $\alpha = 1$ and evaluated with $N = 8$ support examples per class, at inference.

Results are shown in Figure 2. All variations of the proposed model achieved a better classification performance than the baseline. The best F1 score was seen at $H = 1$, with a mean value of .8111 over all evaluation episodes. Compared to the baseline mean score of .7792, this is a 4% improvement. A Wilcoxon signed-rank test showed that all of our proposed models achieve a statistically significant improvement when compared to the baseline, with $p < 10^{-7}$ for all hierarchies. These results show that incorporating our method into a prototypical network can lead to statistically significant improvements in classification performance under few-shot learning conditions.

¹ See: <https://en.wikipedia.org/wiki/Hornbostel-Sachs>

² <https://github.com/hugofloresgarcia/music-trees>

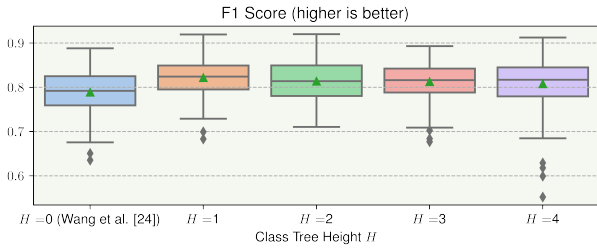


Figure 2. F1 scores for models trained with class trees of varying height H , evaluated over 100 episodes. Means are shown as green triangles. Note that $H = 0$ is our baseline model (Wang *et al.* [25]), as it is trained without a class tree.

Surprisingly, a shallow tree with only the coarsest categories and the leaf nodes ($H = 1$) achieved the highest increase in performance. We believe this is due to the small number of classes encountered in a training episode (in our case, 12). At a given level of the tree, at least 2 of the classes in the support set need to have a parent node in common for our method to be able to compute a meaningful metaprototype that can be leveraged by our loss. As a class tree gets deeper, the number of nodes at a given level can grow exponentially, meaning that our support set of 12 classes has a lower chance of finding meaningful groupings at deeper levels. This indicates that loss terms for levels closer to the leaf nodes are more likely to be identical to the non-hierarchical loss. Though the loss term for the coarsest level is still present in these deeper trees, it has a smaller impact on the gradient of the primary loss function, as loss terms are weighted to decay exponentially as the height increases. We believe training with a higher $|\mathcal{K}|$ can help leverage deeper hierarchies better. However, we leave this for future work.

6.2 Number of Support Examples

We evaluated our best proposed model ($H = 1$, $\alpha = 1$) as well as our baseline model by varying the number of support examples N provided to the model, where $N \in \{1, 4, 8, 16\}$. Results are shown in Figure 3 (left). We notice that increases in performance are greater when more support examples are provided, with the smallest increase (+2.17% in the mean relative to baseline) occurring when $N = 1$. Our model achieved a statistically significant improvement on all test cases ($p < 10^{-4}$ for all N).

As shown in Figure 3 (right), our model achieved a lower hierarchical distance of a mistake, on average. A Wilcoxon signed-rank test indicates that all improvements are statistically significant ($p < .0005$). This means that, when making incorrect predictions, our method was more likely to make predictions that are closer to the ground truth in terms of the class hierarchy (*i.e.*, lower mistake severity). We believe it is fair to assume that mistake severity from a sound production perspective (as in our class hierarchy) is related to mistake severity in predictions made by humans. That is, a human is more likely to confuse a viola for a violin than to confuse a viola for a drum.

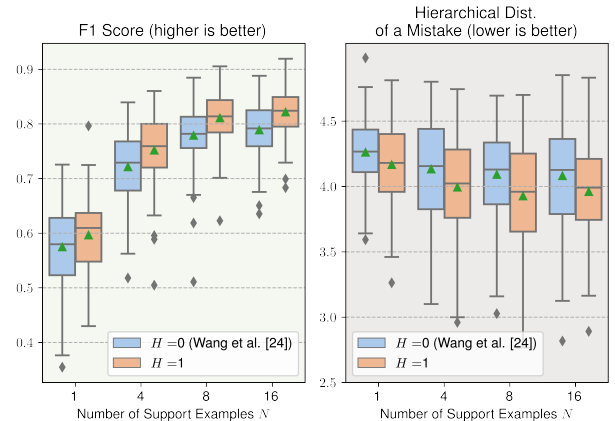


Figure 3. Model comparison between the baseline model and our best proposed model ($H = 1$), evaluated under conditions with a different number of shots (support examples) provided during inference.

6.3 Arbitrary Class Trees

To understand how the choice of hierarchy affects the results of our model, we evaluated the same prototypical network architecture trained using the Hornbostel-Sachs hierarchy and also 10 randomly generated class trees. We generated each tree by performing random pairwise swaps between leaf nodes in our original class tree, doing so 1000 times for each node. For this experiment, all trees were trained with ($H = 3$, $\alpha = 1$), and evaluated with $N = 16$.

Results for our evaluation of random class hierarchies are shown in Figure 5. Our best performing random hierarchy in terms of classification performance ("random-best") achieves an F1 score comparable to our proposed hierarchy ($p > 0.05$) though with a larger spread. Additionally, "random-best" obtains much worse mistake severity relative to the hierarchy it was trained on. This indicates that the model was not able to generalize the hierarchical structure it was trained on to out-of-distribution classes. On the other hand, our worst performing random hierarchy, "random-worst", caused a statistically significant deterioration in both classification performance and mistake severity compared to the baseline ($p < 0.005$). Even though the random-best model fairs comparably to Hornbostel-Sachs model, it is impossible to know *a priori* whether any random tree will produce good results, therefore for practical uses (*i.e.*, within a DAW), we find Hornbostel-Sachs to be a suitable choice.

6.4 Hierarchical Loss Functions

To measure the impact of our proposed multi-task hierarchical loss, we compared it to a reasonable baseline "flat" loss. As our baseline approach, we treated hierarchical classification as a single-task, multilabel classification problem, where the ground truth is a multi-hot vector, with 1s for the leaf ground truth node and all of its ancestors in the tree, and 0s otherwise. Furthermore, we minimized the binary cross entropy between each individual predicted node and ground truth node. Note that this required us to

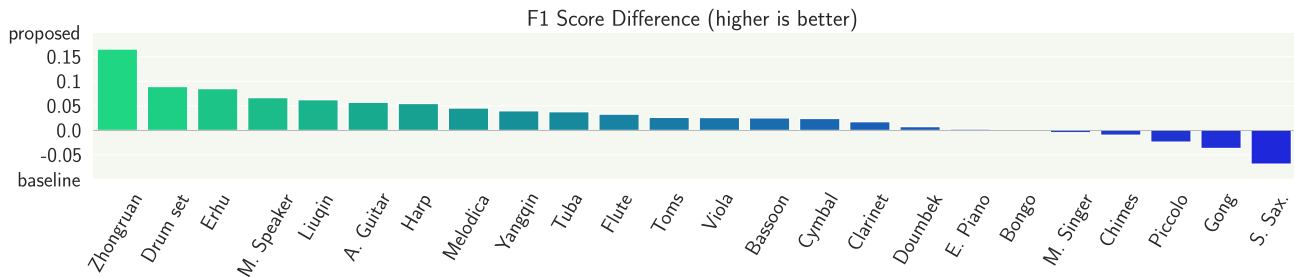


Figure 4. Difference in F1-score between our best proposed model ($H = 1$, $\alpha = 1$) and the baseline (Wang *et al.* [25]) on all instruments in the test set. Both models were evaluated with $N = 8$.

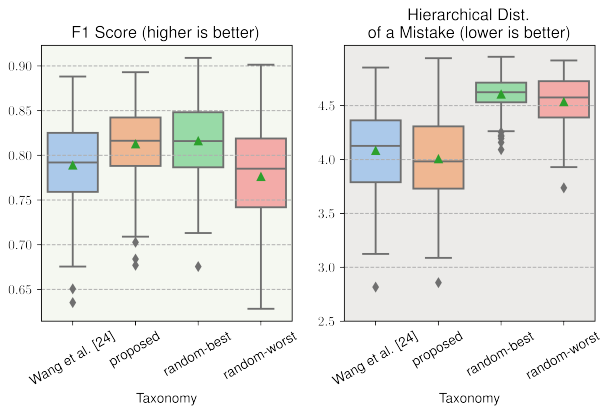


Figure 5. Comparison between the best and worst performing models trained on random hierarchies. The hierarchical distance of a mistake is calculated using the hierarchy the model was trained on. For the baseline (Wang *et al.* [25]), we calculated the hierarchical distance of a mistake using the Hornbostel-Sachs hierarchy.

use a sigmoid function instead of Eq. 2, which uses a softmax function. Additionally, we performed a hyperparameter search to find the best value of the α parameter for our proposed loss function (Section 6.4) using the search space $\alpha \in \{-1, -0.5, 0, 0.5, 1\}$. For this experiment, all trees were trained with $H = 4$ and evaluated with $N = 16$.

Results are shown in Figure 6. We observe that only the models with $\alpha > 0$ cause an improvement over Wang *et al.* [25]. Moreover, the flat loss causes a severe degradation in classification performance. This may be because training prototypical networks using a binary, one-vs-all formulation could yield a much less discriminative embedding space. Wang *et al.* [25] found a similar result: training prototypical networks with a binary formulation did not yield performance improvements.

6.5 Examining All Instrument Classes

In Figure 4, we examine the classification performance of every instrument in our test set. We compare our best model ($H = 1$, $\alpha = 1$) to the baseline model from Wang *et al.* [25], evaluated with $N = 8$. For clarity, we report the difference in F1 Score between the models. Our model beats the baseline on 18 of the 24 classes in the test set. In particular, our model shows a substantial improvement (+16.56%) in F1 Score when classifying *zhongruan*, which may be rarely seen in a dataset composed of Western

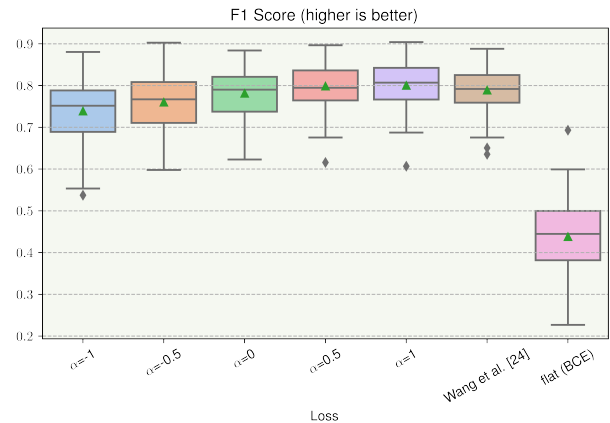


Figure 6. Evaluating the loss function. We vary α in our proposed hierarchical loss from negative (emphasize loss on broader categories) to positive (emphasize loss on finer categories) and additionally compare to a "flat" binary cross entropy (BCE) baseline.

music. Figure 4 demonstrates that, overall, our hierarchical few-shot model is better at identifying a wider range of instrument classes than the baseline. This is important if we desire to make systems that are more robust to biases in the training data and, thus, can classify more a diverse set of instrument types.

7. CONCLUSION

We presented an approach for incorporating hierarchical structures in a few-shot learning model for the purpose of improving classification performance on classes outside of the training distribution. Our method builds on top of prototypical networks by computing prototypical representations at fine and coarse granularities, as defined by a class hierarchy. We showed that our proposed method yields statistically significant increases in classification performance and significant decreases mistake severity when evaluated on a classification task composed of unseen musical instruments. Moreover, we found that the choice of hierarchical structure is not arbitrary, and using a hierarchy based on the sound production mechanisms of musical instruments had the best results. We hope our work enables users with diverse cultural backgrounds with the ability to classify diverse collections of musical instruments. Future directions include examining new types of hierarchies, learning multiple hierarchies simultaneously, and the unsupervised discovery of hierarchies from unlabeled data.

8. ACKNOWLEDGEMENTS

This work was funded, in part, by USA National Science Foundation Award 1901456.

9. REFERENCES

- [1] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 303–319, 2010.
- [2] A. Eronen and A. Klapuri, "Musical instrument recognition using cepstral coefficients and temporal features," in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, vol. 2. IEEE, 2000, pp. II753–II756.
- [3] A. Krishna and T. V. Sreenivas, "Music instrument recognition: from isolated notes to solo phrases," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4. IEEE, 2004, pp. iv–iv.
- [4] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.
- [5] —, "Tut database for acoustic scene classification and sound event detection," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1128–1132.
- [6] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.
- [7] S. Savage, *The art of digital audio recording: A practical guide for home and studio*. Oxford University Press, 2011.
- [8] B. Owsinski, *The mixing engineer's handbook*. Nelson Education, 2013.
- [9] A. Saha and A. M. Piper, "Understanding audio production practices of people with vision impairments," in *The 22nd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '20), October 26–28, 2020, Virtual Event, Greece*. IEEE, 2020.
- [10] A. Tanaka and A. Parkinson, "Haptic wave: A cross-modal interface for visually impaired audio producers," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 2150–2161.
- [11] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera, "A comparison of sound segregation techniques for predominant instrument recognition in musical audio signals," in *International Society for Music Information Retrieval (ISMIR) Conference*. Citeseer, 2012, pp. 559–564.
- [12] E. Humphrey, S. Durand, and B. McFee, "Openmic-2018: An open data-set for multiple instrument recognition," in *International Society for Music Information Retrieval (ISMIR) Conference*, Paris, France, 2018, pp. 438–444.
- [13] Y.-N. Hung and Y. Yang, "Frame-level instrument recognition by timbre and pitch," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018.
- [14] S. Gururani, M. Sharma, and A. Lerch, "An attention mechanism for musical instrument recognition," in *International Society for Music Information Retrieval (ISMIR) Conference*, 2019.
- [15] Y. Hung, Y. Chen, and Y. Yang, "Multitask learning for frame-level instrument recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 381–385.
- [16] M. Taenzer, J. Abeßer, S. I. Mimitakis, C. Weiß, M. Müller, and H. Lukashevich, "Investigating cnn-based instrument family recognition for western classical music recordings," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. ISMIR, 2019, pp. 612–619.
- [17] A. Kratimenos, K. Avramidis, C. Garoufis, A. Zlatintsi, and P. Maragos, "Augmentation methods on monophonic audio for instrument classification in polyphonic music," in *2020 28th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 156–160.
- [18] V. Lostanlen, J. Andén, and M. Lagrange, "Extended playing techniques: the next milestone in musical instrument recognition," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, 2018, pp. 1–10.
- [19] B. Kim and B. Pardo, "I-sed: An interactive sound event detector," in *Proceedings of the 22nd International Conference on Intelligent User Interfaces*, ser. IUI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 553–557.
- [20] M. Cartwright, G. Dove, A. E. Méndez Méndez, J. P. Bello, and O. Nov, "Crowdsourcing multi-label audio annotation tasks with citizen scientists," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–11.
- [21] E. M. von Hornbostel and C. Sachs, "Classification of musical instruments: Translated from the original german by anthony baines and klaus p. wachsmann," *The Galpin Society Journal*, vol. 14, pp. 3–29, 1961.
- [22] M. J. Kartomi, *On concepts and classifications of musical instruments*. University of Chicago Press Chicago, 1990.

- [23] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.
- [24] O. Maimon and R. Lior, *Data Mining and Knowledge Discovery Handbook*. Springer, 2006, ch. Clustering methods.
- [25] Y. Wang, J. Salamon, N. J. Bryan, and J. Pablo Bello, “Few-shot sound event detection,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 81–85.
- [26] E. Benetos, M. Kotti, and C. Kotropoulos, “Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection,” in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, 2006, pp. V–V.
- [27] A. Eronen and A. Klapuri, “Musical instrument recognition using cepstral coefficients and temporal features,” in *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, vol. 2, 2000, pp. II753–II756 vol.2.
- [28] V. Lostanlen and C.-E. Cella, “Deep convolutional networks on the pitch spiral for music instrument recognition,” in *International Society for Music Information Retrieval (ISMIR) Conference*, 2016.
- [29] S. Essid, G. Richard, and B. David, “Hierarchical classification of musical instruments on solo recordings,” in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5, 2006, pp. V–V.
- [30] Y. Han, J. Kim, K. Lee, Y. Han, J. Kim, and K. Lee, “Deep convolutional neural networks for predominant instrument recognition in polyphonic music,” *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 25, no. 1, p. 208–221, Jan. 2017.
- [31] S. Gururani, C. Summers, and A. Lerch, “Instrument activity detection in polyphonic music using deep neural networks,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018.
- [32] R. A. Stein, P. A. Jaques, and J. F. Valiati, “An analysis of hierarchical text classification using word embeddings,” *Information Sciences*, vol. 471, pp. 216–232, 2019.
- [33] A. Dhall, A. Makarova, O. Ganea, D. Pavllo, M. Greff, and A. Krause, “Hierarchical image classification using entailment cone embeddings,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 836–837.
- [34] S. Sun, Q. Sun, K. Zhou, and T. Lv, “Hierarchical attention prototypical networks for few-shot text classification,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 476–485.
- [35] L. Bertinetto, R. Mueller, K. Tertikas, S. Samangooei, and N. A. Lord, “Making better mistakes: Leveraging class hierarchies with deep networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 506–12 515.
- [36] S. Essid, G. Richard, and B. David, “Instrument recognition in polyphonic music based on automatic taxonomies,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 68–80, 2005.
- [37] T. Kitahara, M. Goto, and H. G. Okuno, “Category-level identification of non-registered musical instrument sounds,” in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, 2004, pp. iv–iv.
- [38] S. Fernández, A. Graves, and J. Schmidhuber, “Sequence labelling in structured domains with hierarchical recurrent neural networks,” in *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, 2007.
- [39] R. Sanabria and F. Metze, “Hierarchical multitask learning with ctc,” in *2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 485–490.
- [40] K. Krishna, S. Toshniwal, and K. Livescu, “Hierarchical multitask learning for ctc-based speech recognition,” *arXiv preprint arXiv:1807.06234*, 2018.
- [41] E. Manilow, G. Wichern, and J. Le Roux, “Hierarchical musical instrument separation,” in *International Society for Music Information Retrieval (ISMIR) Conference*, Oct. 2020, pp. 376–383.
- [42] Y. Xu, Q. Huang, W. Wang, and M. D. Plumbley, “Hierarchical learning for dnn-based acoustic scene classification,” *arXiv preprint arXiv:1607.03682*, 2016.
- [43] A. Jati, N. Kumar, R. Chen, and P. Georgiou, “Hierarchy-aware loss function on a tree structured label space for audio event detection,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6–10.
- [44] J. Cramer, V. Lostanlen, A. Farnsworth, J. Salamon, and J. P. Bello, “Chirping up the right tree: Incorporating biological taxonomies into deep bioacoustic classifiers,” in *ICASSP 2020-2020 IEEE International*

Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020, pp. 901–905.

- [45] K. Cheng, S. Chou, and Y. Yang, “Multi-label few-shot learning for sound event recognition,” in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, 2019, pp. 1–5.
- [46] B. Shi, M. Sun, K. C. Puvvada, C.-C. Kao, S. Matsoukas, and C. Wang, “Few-shot acoustic event detection via meta learning,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 76–80.
- [47] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, “Few-shot drum transcription in polyphonic music,” in *International Society for Music Information Retrieval (ISMIR) Conference*, 2020.
- [48] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, “Medleydb: A multitrack dataset for annotation-intensive mir research,” in *15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- [49] R. Bittner, J. Wilkins, H. Yip, and J. P. Bello, “Medleydb 2.0 audio,” Aug. 2016. [Online]. Available: <https://doi.org/10.5281/zenodo.1715175>
- [50] R. Bittner, E. Humphrey, and J. Bello, “Pysox: Leveraging the audio signal processing power of sox in python,” in *Proceedings of the International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2016.

WHAT IF THE ‘WHEN’ IMPLIES THE ‘WHAT’?: HUMAN HARMONIC ANALYSIS DATASETS CLARIFY THE RELATIVE ROLE OF THE SEPARATE STEPS IN AUTOMATIC TONAL ANALYSIS

Mark Gotham^{1,2}, Rainer Kleinertz², Christof Weiß³, Meinard Müller³, Stephanie Klauk²

¹ Institut für Musik und Musikwissenschaft, Technische Universität Dortmund, Germany

² Institut für Musikwissenschaft, Saarland University, Germany

³ International Audio Laboratories Erlangen, Germany

ABSTRACT

This paper uses the emerging provision of human harmonic analyses to assess how reliably we can map from knowing only *when* chords and keys change to a full identification of *what* those chords and keys are. We do this with a simple implementation of pitch class profile matching methods, partly to provide a benchmark score against which to judge the performance of less readily interpretable machine learning systems, many of which explicitly separate these *when* and *what* tasks and provide performance evaluation for these separate stages. Additionally, as this ‘oracle’-style, ‘perfect’ segmentation information will not usually be available in practice, we test the sensitivity of these methods to slight modifications in the position of segment boundaries by introducing deliberate errors. This study examines several corpora. The focus is on symbolic data, though we include one audio dataset for comparison. The code and corpora (of symbolic scores and analyses) are available within: <https://github.com/MarkGotham/When-in-Rome>

1. INTRODUCTION

Since the pioneering work of Carol Krumhansl and colleagues in the 1980s, [1,2] there have been several proposals for prototypical key profiles in tonal music based on the relative importance of the constituent pitches in a key (examples follow in context below). The motivations and data for this approach derive usually from either psychological tests (asking ‘how well does this pitch fit this key context?’, for instance), empirical usage data (‘how often is it used?’), or a combination of the two. The working hypothesis is that there exists a link between this pair: that

substantial past exposure to the statistical regularities of a musical style forms a mental representation which affects our expectations when listening.

The main task for these ‘prototypical’ profiles in the empirical domain is automatic key finding, either for an entire work (‘what is *the* key of this piece’), or for passages (*keys* plural) with the latter often approached in terms of key matching for the usage within ranges delimited by a moving window [3,4].

The idea is that if these *prototypical* pitch profiles give us a strong sense of the relative usage of each pitch in a key, and we also have data for the *actual* pitch usage in a score or audio source of interest, then we can simply compare the source with the prototypes for each key and find the one that fits best.

Several more sophisticated algorithms have been proposed to replace the whole practice of matching profiles [5,6], or enhance that practice in place [7], but there is an enduring attraction to the clarity and simplicity of this approach. That clarity and simplicity could have a particular significance now for evaluating the more opaque machine learning approaches that increasingly dominate this field. While these architectures can be hard to interpret, they often separate the constituent tasks (notably here segmentation from identification) such that their performance can be compared with simpler techniques.¹

1.1 Prototype Profiles: Comparing Like with Like

At their simplest, prototype profiles consist of a binary separation with (typically) a value of 1 for membership of the collection, and 0 otherwise across the pitch classes (0–11), discounting the differences of octave or enharmonic spelling. For instance, such a ‘binary’ profile for the *chord* of C-major (CEG) would be given as:²

[1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0]

while the *key* of C-major (CDEFGAB) is represented by:

[1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]

¹ On segmentation as part of chord analysis with machine learning in symbolic data, see especially [8,9].

² Note that we speak only of ‘representation’ here: there is clearly more to both chords and keys than these simple pitch class profiles.



‘Rotations’ of these profiles cover all the transposition-equivalent sets, in this cases encompassing all major triads and all major keys respectively.

Alternatively, profiles can be constructed from the empirical evidence of either psychological ‘goodness of fit’ studies or from musical practice, for instance, by taking the pitch class profile (hereafter, PCP) of all events in a dataset (‘symbolic’ or audio) considered relevant.

These more subtly weighted profiles help to address vexed issues like scale degrees 6 and 7 in the minor mode. They can also help to introduce other, potentially important contextual sensitivity in response to variables that limit the effectiveness of a one-size-fits-all approach to the profile. Literature on symbolic datasets of Western classical music specifically include accounts of:³

- **Repertoire.** [11] demonstrates changing PCPs for repertoire created during the historical period in which there is a move from modality to tonality.⁴
- **Specific keys.** [14] demonstrate a small but significant effect of key on the resulting usage profiles, reinforcing the received wisdom that tonal composers do not regard transposition as a neutral change.
- **Partial pieces.** [11, 14, 15] and others demonstrate that drawing prototype profiles from a short passage at the beginning and/or end improves performance.

1.2 The Part versus the Whole

The last of these points is significant for our purposes. Given that common practice tonal music almost always starts and ends unequivocally in the same, main key, it makes perfect sense that these regions would be more tonally-stable, and thus provide a better indicator of what the PCP for within-key music looks like.

And while the benefit may be only slight for identifying the key of an overall piece, we can realistically expect a greater improvement for shorter, partial piece comparisons. This is because the whole practice of profile matching depends on comparing like with like.

In the extreme case, if we used a chord template for key matching or vice versa, we should expect the system to perform worse on average. More realistically, this applies to the vast grey area between chord and key. For instance, what could be a clearer statement of key than a simple V7-I progression? Assuming the two chords have equal length and exactly one instance of each constituent pitch, this equates to an overall PCP in C major of:

[1, 0, 1, 0, 1, 1, 0, 2, 0, 0, 0, 1].

Several of the main PCPs in the literature would confuse this short passage in C major for one in G major despite the presence of F and lack of F♯: the double weighting of the pitch G is enough to tip the balance. This is not to criticise

³ See [10] for a recent overview of the audio literature on these topics that has no overlap with the symbolic papers cited here.

⁴ There has also been a notable, recent return to probe-tone psychological study of repertoire difference. See [12] (after [13]) on distinguishing ‘classical’ and ‘rock’ styles.

any specific scholarship or PCP, but simply to illustrate the problem with applying a key-matching profile to a passage that is too short.

In general, to build PCPs that perform well for the task of identifying keys within a piece effectively, we may have most success by creating those profiles from passages that are firmly attested to be in a given key, rather than assumed to be so. The barrier to generating these PCP models in this way is that it requires us to know what the keys are and where they change in the first place, yet that is the problem that we are trying to solve [14, p.532]. One solution is to build up corpora of human annotations in other, relevant (‘similar’) music, which can then serve as a model for an automatic approach to new cases that do not come with a manual analysis.

1.3 Human analyses for keys and chords

Fortunately, recent years have seen the creation of several human harmonic analysis corpora.⁵ Equipped with this data, we can do better than taking entire works or even shorter spans of an arbitrary length as the basis for our PCPs, focussing instead on passages corresponding exactly to these human-defined segmentations, and building up profiles from passages more robustly ‘known’ to be within-key.⁶ Profiles from individual passages can then be combined with other, comparable passages according to the user’s priorities (e.g., repertoire, length, key, and position in work) to yield new models for profile matching.

Moreover, these full harmonic analysis datasets enable us to apply the same logic to the equivalent task for chords. While the field of automatic chordal analysis is at least as established as the equivalent for key, repertoire-based PCP models for chord recognition are rarely available, at least for ‘classical’ music and symbolic data. This is entirely understandable given that chord-level analysis is much more fine-grained than key-only: they take longer to create, and are harder to manage when alignment issues are concerned (e.g., for multiple sources).

In both cases, the human analyses provide both full details of *what* the chords and keys are, but also *when* they change. This allows us to take the *when* information alone, segment the corpora into short segments for each chord or key, and compare the PCP of that passage to a reference profile to assess how reliably the *when* implies the *what*.

This paper undertakes that comparison for the case of both chords and keys, and across several corpora. Further, we consider how dependent this process is on the exact segmentation by introducing systematic errors to see how deleteriously this affects the results. In all cases, we at least start with simple, highly interpretable conditions: ℓ^1 -normalisation and the Manhattan comparison metric; simple (e.g., binary) reference profiles; and a clear separation of ‘known’ information (the ‘*when*’ of segmentation) from the ‘tested’ part (the ‘*what*’ of identification).

⁵ Datasets with full (chord and key) analyses of Western classical music include [16–20].

⁶ This does not, of course, account for inter-analyst disagreement. We should avoid the term ‘ground truth’ for human annotations.

2. CORPORA

For comparison, this study draws together a range of sources across **repertoires** (within the symbolic domain) and **data types** (both audio and symbolic representations of one repertoire). Specifically, for the audio-symbolic comparison, we use the **Beethoven sonata first movements** with score and audio data from [21] and analyses originally from [17]. The audio-analysis alignment includes key- but not chord-level section data, so this is currently limited to the key-level study.

The corresponding score-analysis alignment is as detailed at the ‘When in Rome’ repository,⁷ and includes both keys and chord. ‘When in Rome’ provides a single, consistent, human- and computer-readable format for all publicly-shared, encoded corpora of Roman numeral harmonic analyses of notated works.⁸ First reported in [19], the repository continues to grow and currently includes ca.450 analyses of works by 100 composers (unevenly distributed) for a total of ca.100,000 Roman numerals.

The largest of the new datasets within this framework comprises over 150 analyses of 19th-century songs from the **OpenScore Lieder Corpus**.⁹ This provides a useful and interesting counterpoint for across-repertoire comparisons, balancing similarity and difference. The songs are from a similar time period to the Beethoven sonata movements (overlapping, though mostly later), for similar forces (the solo piano now joined by a voice part), and generally slightly shorter (but not always).

The Beethoven sonatas and lieder provide the main two symbolic corpora studies here, supplemented in one case by the **Bach Well Tempered Clavier** collection, also from the ‘When in Rome’ meta-corpus, and as reported in [19].

2.1 Data preparations for across-domain comparison

Despite the self-evident differences between audio and symbolic data, we seek to make the representations as similar and comparable as possible. To that effect, as well as taking a frame-by-frame approach to audio (sampling rate 10Hz), we approach the symbolic data in a comparable way, converting encodings via musicXML to a similar ‘slice’ representation that encodes a new data point for each change of pitch in the score.¹⁰

In both cases, given segment timing information, frames and slices within the corresponding segment can be combined into single PCPs and compared (via the same normalisation) to reference profiles. For the symbolic data, we provide the full set of these ‘slice’ files as well as another set of files recording the pitch-class profiles for each section asserted to be in one key at the ‘When in Rome’ repository. This makes processing faster and less dependent on external libraries, which in turn makes replication studies more practical.

Specifically, for every symbolic source, these files provide the metadata (including title and composer) and record for each key-section at least the:

- **key:** tonic pitch (such as E \flat) and mode (major or minor, indicated by case, e.g. ‘F \sharp ’ versus ‘f \sharp ’);
- **profile:** the raw (not normalised) PCP of usage;
- **start and end ‘offset’:** as measured from the start of the piece in ‘quarter note’ symbolic values as well as the ‘quarter length’ recording the difference;
- **start, end, and length in measures:** the equivalent measurements using symbolic ‘measures’.

For the audio sources, we use absolute duration in seconds as the primary measurement of time.

From this point, it is easy and computationally inexpensive to build new model PCPs from the entries relevant to a specific use case. For instance, to assess the best-fit minor key for passages of 20 measures’ duration, we may want to build and use a model profile from all minor-key entries of between, say, 15 and 25 measures, ignoring shorter or longer passages, and perhaps also restricting the sample to the composer and / or genre in question. Alternatively, these ‘typical’ ranges could also be used to inform parameter setting for variable window size.

2.2 Two qualifications: subjectivity and similarity

First, it bears repeating that human analysis datasets – valuable as they are – are naturally and necessarily subjective. While we often see strong agreement for simple cases, analysts differ greatly in their view of more complex passages. Then again, that is exactly the object of this research area. If there were strict, comprehensive rules mapping from a score or audio source to a single, ‘correct’ analysis, there would be no need for either the corpora or the studies presented by this paper and the wider field.

Second, while it is expedient initially to work with simple, True/False data for the presence/absence of a match between human and computational key choice (as we do here), chords and keys really exist in a relative proximity relation. For more on the *kinds* of ‘errors’ that are typical, see [26]’s early examination of the tendencies of certain reference profiles and [20, 27] on discrepancies between a computer reading and several manual annotations of local key in Schubert’s *Winterreise* song cycle.

3. CHORD-LEVEL SEGMENTATION

We begin with the case of chord identification, a problem operationally defined here as the selection from among 9 distinct chord types in any of the 12 transpositions, making for 108 options in total. In these studies we test chord and key identification from the corresponding segmentation separately, so chords are defined in ‘absolute’ terms

⁷ <https://github.com/MarkGotham/When-in-Rome>

⁸ In addition to [17], this includes [16, 18] and more.

⁹ Originally reported in [22]; now released as an MIR dataset in [23].

¹⁰ See [24] (after [25]) for more details and code.

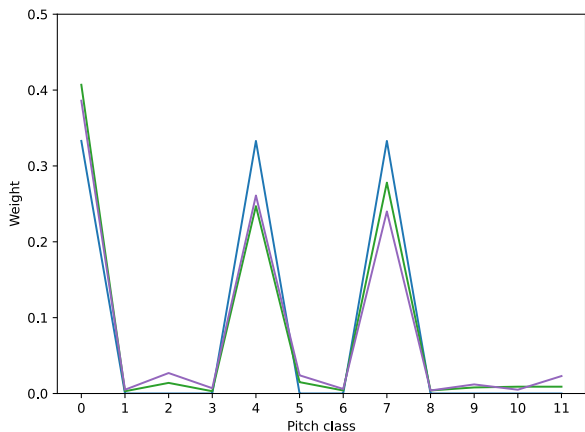


Figure 1. A comparison of ℓ^1 -normalised profiles for the major triad with binary values (blue), alongside values from the Lieder (green) and Beethoven (purple) corpora.

(e.g. the *triad* C major) rather than in relation to a key (‘tonic’ or ‘I’ in the *key* of C-major).¹¹

3.1 Included v.s. excluded chords by type and %

The 9 chords types in question are the four triads (major, minor, diminished, augmented) and five of the most common sevenths (dominant, major, minor, diminished, half-diminished). Around 95% of all chords in the corpora are accounted for by one of these. The remaining ca.5% of cases deemed outside the scope for the current study include: additional seventh types such as sevenths built on augmented triads (relatively rare); all further tertian chords (i.e., 9ths: there are no 11th or 13th chords in the corpora); some chromatic chords like augmented sixths; and detailed entries using RomanText’s syntax for supporting missing, added, and altered tones.

To support other approaches to this problem, we provide code for simplifying these chords to what might be considered the ‘nearest’ corresponding member of the canonical 9 types. For instance, this means removing the 9th of a 9th chord to yield a 7th chord, and ‘completing’ incomplete triads. All the same, we operationally exclude these cases in the present comparisons as they do not directly reflect the analyst’s stated view. The ‘N/A’ values on Table 1 provide exact numbers of these cases, corpus by corpus.

3.2 Repertoire-specific profiles

Complementing the binary profiles, we create and use a new set of PCPs extracted from the corpora at hand. Given robustly aligned data, the method for this is straightforward: for each chord (in the analysis), identify the triad or seventh type, extract the PCP (from the corresponding range of the score), rotate it to place the chord’s root on C (pitch class 0), add each PCP usage value to running totals for the relevant triad or seventh type.

¹¹ This testing of key and chord separately accounts for most of the relevant considerations, though it is worth noting that the Roman numeral encoding includes other, intermediary information such as ‘secondary’ key tonicizations.

Repertoire	Matches analysis			Total	% True from True+False
	True	False	N/A		
Binary reference profiles:					
Bach	1149	865	95	2109	57.051
Beethoven	3782	2058	61	5901	64.760
Lieder	8395	2897	505	11797	74.345
<i>Winterreise</i>	1899	660	96	2655	74.209
Profiles from Beethoven:					
Bach	1182	832	95	2109	58.689
Beethoven	3880	1960	61	5901	66.438
Lieder	8187	3105	505	11797	72.503
<i>Winterreise</i>	1874	685	96	2655	73.232
Profiles from Lieder:					
Bach	1237	777	95	2109	61.420
Beethoven	4101	1739	61	5901	70.223
Lieder	8695	2597	505	11797	77.001
<i>Winterreise</i>	2013	546	96	2655	78.664

Table 1. Chord-level segmentation to chord identification, separating values for correct (‘True’), incorrect (‘False’) and out of scope (‘N/A’).

An alternative strategy would keep separate PCPs without transposition. We decide against that approach here. First, the datasets are not that large, and some chord types (such as augmented triads) are rather rare: this suggests erring on the side of caution, creating fewer distinct chord type PCPs from a greater number of repertoire instances. Second, the lieder dataset is central here, and it is common practice to transpose those songs to a variety of keys depending on the vocal range of the performer. Composers are aware of this, and it stands to reason that key-specific writing is rarer here than in symphonies, say. Finally, and related, it is not obviously better to keep chords separate by absolute triad (e.g., recording a PCP for C major) than by within-key, functional status (e.g., for all tonic major triads). We anticipate further investigation of these areas as the provision of analysis corpora grows and matures.

For illustration, Figure 1 plots the ℓ^1 -normalised profiles for the major triad as extracted from the Beethoven and lieder corpora along with the (also normalised) binary profile. Note how the binary profile (blue line) preserves equal weighting of C, E, and G (pitch classes 0, 4, and 7), while the repertoire cases place greater emphasis on the tonic, C, less on E and G, and include some use of the non-chord tone pitch classes.

3.3 Results

Table 1 provides comparative data for this task across the corpora and prototypes. Specifically, we begin with the binary profiles discussed above, applying these to the Bach, Beethoven and Lieder corpora, as well as single collection from within the corpus (Schubert’s *Winterreise*) for comparison. We then apply the same method with new chord PCPs extracted from the Beethoven and Lieder corpora.

In all cases, we ℓ^1 -normalise both the source PCP and the 108 reference profiles, take the Manhattan distance between the two, and return the top-choice from among those 108 options. Each individual case is a match if and only if the top-choice is the same as that given by the analyst. The

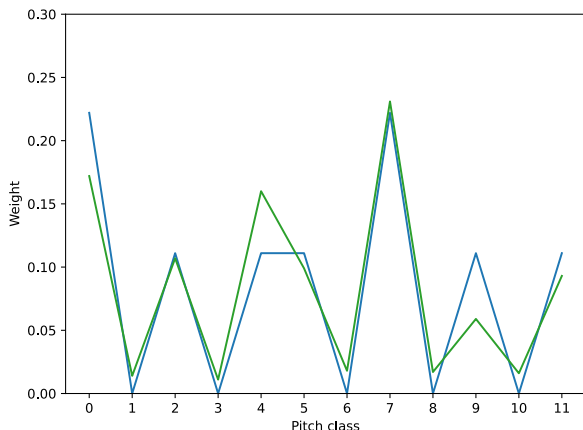


Figure 2. A comparison of ℓ^1 -normalised profiles for the major mode with values from C.S. (blue) and Q.W. (green).

final score is given by the percentage of ‘correct’ responses (computational process matches human judgement) from the total cases (excluding those out of scope).

These results suggest two main observations. First, the success rate of ca.70% across the board is relatively high, considering the simplicity of the algorithm, and the fact that any divergence counts as a failure, including the often slight difference between V and V7. This appears to indicate that the *where* of segmentation is a significant part of the problem: once ‘solved’, even simple algorithms perform very well in determining the *what* of identification.

Secondly, we should consider the effect of using repertoire-specific PCPs. For these tasks, the lieder PCPs substantially improve results on all corpora, while the Beethoven PCPs are more mixed: they perform better than the binary PCPs on the Bach and Beethoven corpora, but worse for the lieder. The success of the lieder PCPs may speak to the benefit of using PCPs that reflect averages across a more diverse corpora. This has significant ramifications for the field, given that most corpora still pursue a single, focussed repertoire of works by one composer.

4. LOCAL-KEY LEVEL SEGMENTATION

Turning to the equivalent task for local key, there are several, existing PCPs available. Figure 2 illustrates the difference between two contrasting profiles from the literature used here. First, **Craig Sapp (C.S.)** [26] provides deliberately simple (nearly binary) profiles. Specifically, Sapp starts with values of 1 for within-key and 0 for chromatic pitches, but adds ‘an additional value of 1’ to the tonic and dominant pitch classes (0 and 7) to mitigate ‘modal confusion between relative major and (natural) minor keys’.

This profile provides a clear, discrete point of comparison for more fine-tuned, continuous alternatives, notably **Quinn and White (Q.W.)** [14] which distinguishes itself as the only set of profiles to go beyond transposition-equivalence and offer key-specific usage profiles.¹² As

¹² Note that although Figure 2 provides a major-key composite for direct comparison, the present study uses their key-specific profiles.

Reference Profile	Manhattan to human	Euclidean to human	Manhattan to Euclidean
Lieder			
C.S.	74.640	73.583	92.123
A.S.	76.177	74.928	92.315
Q.W.	77.618	76.081	93.756
Beethoven (symbolic)			
C.S.	70.529	73.300	87.657
A.S.	79.345	80.353	92.191
Q.W.	85.39	82.620	93.199
Beethoven (audio)			
C.S.	51.263	51.136	82.828
A.S.	63.763	61.490	84.722
Q.W.	68.813	67.298	87.879

Table 2. The percentage of segments for which the comparison metrics match the human judgement.

part of the ‘When in Rome’ repository, we provide these along with all published profiles, enabling others to experiment with the full range. As Figure 2 shows, the two distributions differ primarily in their handling of scale degrees 3 and 6 (pitch classes 4 and 9).

For this test, we expand the corpora to include the Beethoven audio, and add a third reference profile from **Albrecht and Shanahan (A.S.)** [15]. We also include an additional comparison between using the ℓ^1 -normalisation with Manhattan distance metric, and the ℓ^2 -normalisation with the Euclidean distance, comparing each to the human-asserted key and additionally to each other.

Table 2 sets out the results in the form of percentages of segments determined by analysts to be in a single key for which the comparison metrics yield a match, choosing from among the 24 keys (12 major, 12 minor). In all cases, the Q.W. profile, ℓ^1 -normalisation and Manhattan distance metric perform best (as highlighted in bold on the Table). The reference profiles are particularly compelling (the normalisation/distance metric paints a more mixed picture). It is perhaps also reassuring that the symbolic and audio corpora follow the same trend.

5. SENSITIVITY TO SEGMENTATION ERROR

While it is useful to know how profile matching performs given ‘perfect’ segmentation data, for most prospective use cases, we need to know the effect of ‘near-misses’ in the segmentation. This is important given the subjectivity of human analysis annotation in general, and the fact that segmentation appears to be a particularly variable element.

To that effect, let us return to the Beethoven audio corpus as a test case for considering the effect of segmentation ‘error’. For this final test, we introduce systematic segmentation errors across the range of likely ‘near misses’. This means varying both the *length* of the segment (making it longer or shorter) as well as the *position* of that modification: adjusting the start of the segment, the end, or both (sharing the length change equally between start and end, centring the new span form on the original range).

The dataset comprises 792 key-segments with a mean duration of 17.8 seconds and a standard deviation of 24.2

Extra length in seconds (total)	Applying adjustment to:		
	the end	the start	both (shared)
-8 sec.	35.859	39.394	35.101
-4 sec.	49.369	51.136	49.116
-2 sec.	58.838	57.323	57.828
-1 sec.	65.404	61.869	61.995
0 sec.	68.813	—	—
1 sec.	66.162	68.182	66.793
2 sec.	61.742	62.626	63.889
4 sec.	53.914	49.495	52.146
8 sec.	45.202	35.606	38.510

Table 3. The percentage of segments yielding a match when deliberately diverging from the analyst-defined section length (using Q.W.’s PCPs). The central duration of 0 seconds refers to the ‘correct’ (analyst defined) length; negative values are shorter; positive are longer.

due to a long tail (many segments last more than a minute), and we adjust the length by $\pm 1, 2, 4,$ and 8 seconds per segment. In the original, segment boundaries create contiguous blocks: the end of one segment is the start of the next. As such, increases (positive length errors) mean overlapping segments such that frames originally near the boundary will be considered as part of both the foregoing and following segments. In this scenario, the first and last segments of each movement also extend into the preceding and following silence.

Decreases (negative length error) mean introducing gaps such that there are boundary passages between segments that are not considered at all. In a few cases (of short segments subject to a large change), the segment may be shortened by more than its total length, thus creating a segment with a (musically meaningless) negative duration.

To operationalise an approach to these situations, both the silent *frames* and the negative-duration (non-existent) *segments* return a flat profile with equal weighting for each pitch class. This, in turn, always makes the same choice of best key (an arbitrary one that is usually wrong).

Table 3 and Figure 3 set out the results. Perhaps the most notable outcome here is the asymmetry: shortening a segment is typically more damaging than extending it, particularly in the ‘start’ condition and the most relevant range of small timing errors. While some of this effect may be due to the handling of negative length described above, that would affect start and end conditions equally and almost never have a bearing on small changes of ± 1 second. Instead, a start error of $+1$ second leads to a drop in performance of only 0.631% , while the equivalent error of -1 yields more than 10 times the drop: 6.944% .

On the one hand this is surprising. Reducing the segment length means the new passage is still within the range defined by the analyst to be in-key. In this case, we have lost some of the relevant, within-key material, but not added anything from the neighbouring sections in different keys. We might expect this to be barely any more difficult than no change, and certainly less problematic than extending into another segment in a different key.

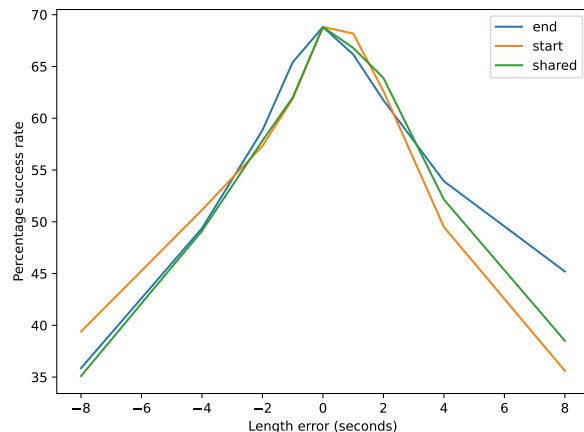


Figure 3. The effect of segmentation errors on the Beethoven audio corpus by error size (x -axis, seconds) and the position of the error (start of segment, end, or shared).

On the other hand, the moments at which we enter a new key area often announce themselves with material characteristic of the new key such as a dominant seventh chord. According to that view, starting late means missing important material. Viewed this way, it is unsurprising at least that *late starts* adversely affect key recognition.

6. CONCLUSION

This paper has sought to demonstrate both the utility of human analyses for evaluating automatic key- and chord-detection in general and specifically how the very simple information contained therein for *when* chords and keys change can be significant for determining the *what* of full harmonic analyses. We demonstrate this with very simple algorithms that are fully transparent, interpretable, and computationally lightweight.

At a minimum, the results provide important benchmark values for the equivalent task within machine learning architectures that have become popular tools for this field. It may also suggest more efficient work-flows for producing human analyses by separating the tasks which computational processes can perform well from those for which we really need expert annotators.

Having demonstrated the relatively high performance of such simple methods for exact matches, a final section considers the effect of small errors in segmentation. There appears to be an asymmetrical effect of error type by length and position, with late starts being notably damaging for even the shortest adjustments. As these artificial errors emulate a more realistic scenario for many data-driven processes that do not have segmentation information available, this result may have significant implications, for instance in setting window size and tolerance thresholds.

In short, the *what* is highly interrelated with the *when*, at least in the ‘idealized’ case of full, manual, human harmonic analysis. Segmentation may not be all we need, but it certainly does contribute a great deal, especially relative to the simplicity of the information it encodes.

Acknowledgements: This work was supported by the German Research Foundation (DFG MU 2686/7-2, KL 864/4-2). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

7. REFERENCES

- [1] C. L. Krumhansl and E. J. Kessler, “Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys,” *Psychological Review*, vol. 89, no. 4, pp. 334–368, 1982.
- [2] C. L. Krumhansl, *Cognitive Foundations of Musical Pitch*. Oxford University Press, 1990, 2001.
- [3] C. S. Sapp, “Visual hierarchical key analysis,” *Computers in Entertainment (CIE)*, vol. 3, no. 4, pp. 1–19, Oct. 2005. [Online]. Available: <http://doi.acm.org/10.1145/1095534.1095544>
- [4] C. Weiß and J. Habryka, “Chroma-based scale matching for audio tonality analysis,” in *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, Berlin, Germany, 2014, pp. 168–173.
- [5] I. Quinn, “Are pitch-class profiles really “key for key”?” *Zeitschrift der Gesellschaft für Musiktheorie [Journal of the German-Speaking Society of Music Theory]*, vol. 7, 01 2010.
- [6] L. Feisthauer, L. Bigo, M. Giraud, and F. Levé, “Estimating keys and modulations in musical pieces,” in *Proceedings of the 17th Sound and Music Computing Conference*, 2020, pp. 323–330.
- [7] N. Nápoles López, C. Arthur, and I. Fujinaga, “Key-finding based on a hidden markov model and key profiles,” in *Proceedings of the 6th International Conference on Digital Libraries for Musicology*, ser. DLfM '19. New York, NY, USA: ACM, 2019. [Online]. Available: <http://doi.acm.org/10.1145/3358664.3358675>
- [8] T.-P. Chen and L. Su, “Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 259–267. [Online]. Available: <https://doi.org/10.5281/zenodo.3527794>
- [9] A. McLeod and M. Rohrmeier, “A modular system for the harmonic analysis of musical scores using a large vocabulary,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2021, forthcoming.
- [10] J. Pauwels, K. O’Hanlon, E. Gomez, and M. B. Sandler, “20 years of automatic chord recognition from audio,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 54–63. [Online]. Available: <https://doi.org/10.5281/zenodo.3527739>
- [11] J. D. Albrecht and D. Huron, “A Statistical Approach to Tracing the Historical Development of Major and Minor Pitch Distributions, 1400-1750,” *Music Perception*, vol. 31, no. 3, pp. 223–243, 02 2014. [Online]. Available: <https://doi.org/10.1525/mp.2014.31.3.223>
- [12] D. T. Vuvan and B. Hughes, “Probe tone paradigm reveals less differentiated tonal hierarchy in rock music,” *Music Perception: An Interdisciplinary Journal*, vol. 38, no. 5, pp. 425–434, 2021.
- [13] T. de Clercq and D. Temperley, “A corpus analysis of rock harmony,” *Popular Music*, vol. 30, no. 1, pp. 47–70, 001 2011.
- [14] I. Quinn and C. W. White, “Corpus-Derived Key Profiles Are Not Transpositionally Equivalent,” *Music Perception*, vol. 34, no. 5, pp. 531–540, 06 2017. [Online]. Available: <https://doi.org/10.1525/mp.2017.34.5.531>
- [15] J. Albrecht and D. Shanahan, “The use of large corpora to train a new type of key-finding algorithm,” *Music Perception: An Interdisciplinary Journal*, vol. 31, no. 1, pp. 59–67, 2013.
- [16] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, “Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, 2015, pp. 728–734. [Online]. Available: http://ismir2015.uma.es/articles/261_Paper.pdf
- [17] T. Chen and L. Su, “Functional harmony recognition of symbolic music data with multi-task recurrent neural networks,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018, pp. 90–97. [Online]. Available: http://ismir2018.ircam.fr/doc/pdfs/178_Paper.pdf
- [18] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, “The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets,” *Frontiers in Digital Humanities*, vol. 5, no. 16, 2018. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fdigh.2018.00016>
- [19] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, “The Romantext format: A flexible and standard method for representing Roman numeral analyses,” in *Proceedings of the 20th International*

Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019, A. Flexer, G. Peeters, J. Urbano, and A. Volk, Eds., 2019, pp. 123–129. [Online]. Available: <http://archives.ismir.net/ismir2019/paper/000012.pdf>

- [20] C. Weiß, F. Zalkow, V. Arifi-Müller, M. Müller, H. V. Koops, A. Volk, and H. G. Grohgan, “Schubert Winterreise dataset: A multimodal scenario for music analysis,” *Journal on Computing and Cultural Heritage*, vol. 14, no. 2, May 2021. [Online]. Available: <https://doi.org/10.1145/3429743>
- [21] C. Weiß, S. Klauk, M. Gotham, M. Müller, and R. Kleinertz, “Discourse not dualism: An interdisciplinary dialogue on sonata form in Beethoven’s early piano sonatas,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020, pp. 199–206.
- [22] M. Gotham, P. Jonas, B. Bower, W. Bosworth, D. Rootham, and L. VanHandel, “Scores of scores: An OpenScore project to encode and share sheet music,” in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, ser. DLfM ’18. New York, NY, USA: ACM, 2018, pp. 87–95. [Online]. Available: <http://doi.acm.org/10.1145/3273024.3273026>
- [23] M. Gotham and P. Jonas, “The OpenScore Lieder Corpus,” in *Music Encoding Conference*, ser. MEC ’21. MEC, 2021, forthcoming.
- [24] M. Gotham, “Moments musicaux,” in *6th International Conference on Digital Libraries for Musicology*, ser. DLfM ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 70–78. [Online]. Available: <https://doi.org/10.1145/3358664.3358676>
- [25] C. W. White and I. Quinn, “The Yale-Classical Archives Corpus,” *Empirical Musicology Review*, vol. 11, no. 1, 2016.
- [26] C. S. Sapp, “Computational methods for the analysis of musical structure,” Ph.D. dissertation, Stanford University, 2011.
- [27] C. Weiß, H. Schreiber, and M. Müller, “Local key estimation in music recordings: A case study across songs, versions, and annotators,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2020.

LET'S AGREE TO DISAGREE: CONSENSUS ENTROPY ACTIVE LEARNING FOR PERSONALIZED MUSIC EMOTION RECOGNITION

Juan Sebastián Gómez-Cañón¹ Estefanía Cano² Yi-Hsuan Yang³
Perfecto Herrera¹ Emilia Gómez^{4,1}

¹ Music Technology Group, Universitat Pompeu Fabra, Spain

² Songquito UG, Erlangen, Germany

³ Academia Sinica, Taiwan

⁴ Joint Research Centre, European Commission, Seville, Spain

juansebastian.gomez@upf.edu

ABSTRACT

Previous research in music emotion recognition (MER) has tackled the inherent problem of subjectivity through the use of personalized models – models which predict the emotions that a particular user would perceive from music. Personalized models are trained in a supervised manner, and are tested exclusively with the annotations provided by a specific user. While past research has focused on model adaptation or reducing the amount of annotations required from a given user, we propose a methodology based on uncertainty sampling and query-by-committee, adopting prior knowledge from the agreement of human annotations as an oracle for active learning (AL). We assume that our disagreements define our personal opinions and should be considered for personalization. We use the DEAM dataset, the current benchmark dataset for MER, to pre-train our models. We then use the AMG1608 dataset, the largest MER dataset containing multiple annotations per musical excerpt, to re-train diverse machine learning models using AL and evaluate personalization. Our results suggest that our methodology can be beneficial to produce personalized classification models that exhibit different results depending on the algorithms' complexity.

1. INTRODUCTION

Historically, the field of MER has mainly focused on extracting meaningful acoustic features from audio and associating them to possible emotions that music can convey [1]. Machine learning algorithms are trained with these features and then linked with the emotional judgements that annotators report to perceive or feel when listening to the music [2] – ultimately presented as "ground truth" to the algorithms. One of the key issues to MER

is the difference between perceived and induced emotions: perceived emotions are the listeners' judgements with respect to musical properties (e.g., key, tempo, timbre), while induced (or felt) emotions are those that the music may arouse within the listener. Despite the effort from the field of music cognition to better understand the psychological differences between these emotions [3–6], their contrast poses a fundamental obstacle to the field of MER. Namely, the construction of the needed "ground truth" results questionable: (1) listeners are commonly confused between perceived and induced emotions when reporting their emotional judgements, (2) the inherent subjectivity from the annotation task is typically addressed by averaging the several annotations into a common "ground truth", and (3) the annotation procedure is a highly demanding task resulting in small datasets with few annotations per music excerpt. Nonetheless, researchers from the field of MER have tackled this problem by: (1) identifying whether the listener's response is based on the judgement of perceived or induced emotions [7] and attempting to train listeners [8], (2) using exclusively the emotion reports from a particular listener or group of listeners to produce personalized and group-based models [9], and (3) introducing AL methods to reduce the amount of annotations required to train such algorithms (see Section 2). Given the importance the construction of a "ground truth" for MER, we address two research questions in this paper:

RQ1 - Can we exploit human agreement in music emotion annotations as input for AL methodologies to produce personalized models?

RQ2 - What is the impact of the choice of classification algorithm on personalization of MER systems?

The rest of this paper is structured as follows: Section 2 reviews basic definitions and previous work, in Section 3 we detail the methodology of our study, including the proposed consensus entropy methods and classification schemes. Section 4 provides results of our study which are later discussed in Section 5.

2. RELATED WORK

Individual differences of listeners have a significant impact on the performance of a MER algorithm. To this



extent researchers have proposed two distinct solutions to tackle the subjectivity issue [1,9]: group-based and personalized MER models. Group-based MER assembles annotators according to individual factors (e.g., sex, age, music experience) and create a common "ground truth" for this group. Personalized MER uses the annotations from a specific user to train a machine learning model. Yang et al. [9] tested both approaches for the regression task and found that: (1) group-based methods do not outperform general models (i.e., models which are trained with the common "ground truth" from the complete set of users), and (2) personalized algorithms largely outperform general models. However, Gómez-Cañón et al. [10] studied the influence of native language and self-reported lyrics comprehension on the agreement of annotations and their impact on group-based MER classification. The authors found substantial differences in the annotations of users with different mother tongues (consistent with findings from [11, 12]), and a direct impact of individual differences (i.e., familiarity, preference, and lyrics comprehension) on the agreement of these annotations. The authors also reported that group-based MER algorithms trained on the annotations of users that reported understanding the lyrics, consistently outperformed general models for a small dataset with a large amount of annotations per excerpt, contradicting results by Yang et al. [9]. More research is needed on the topic of group-based MER, hence in this paper we focus on the need of personalization strategies.

Su and Fung [13] proposed using AL (i.e., uncertainty sampling) in order to achieve personalization – which is the focus of this paper. The aim of AL is to minimize the annotation cost by cleverly choosing unlabeled data instances, such that machine learning algorithms perform better with less training [14]. Sarasúa et al. [15] used it to reduce the amount of training instances and achieve better classification performance for MER. Uncertainty sampling uses the posterior probability from a classification model to assess the most difficult/uncertain unlabeled data instances (e.g., consider an output probability of 0.5 for binary classification).¹ Su and Fung [13] used two sampling methods to select training instances: (1) using the most *informative* instances – with highest uncertainty, and (2) using the most *representative* instances – with least uncertainty. Their results suggested that AL can reduce the annotation task up to 80% without decreasing performance of classification. However, the performance of AL as a personalization strategy appears to be hindered by low quality annotations – a problem known as the "noisy oracle" issue: low reliability in annotations results in poor training instances, in turn resulting in poor classification performance. In this direction, multi-oracle AL [17, 20–23] has been proposed to exploit multiple annotators by estimating the importance of both unlabeled instances and the expertise of each annotator – ultimately improving label quality. More recently, Chen et al. [24, 25] proposed model adaptation to achieve personalization. Their approach relied on developing a

general MER regression model (namely, Gaussian Mixture Models) and progressively tying the Gaussian components to adapt the models based on the maximum a posteriori (MAP) linear regression. Results evidence that only 10-20 personal annotations are necessary to obtain the same level of accuracy as a baseline model (50 annotations). However, they found no statistically significant difference between the proposed tying methods. Overall, we find two limitations in the MER personalization literature: (1) the evaluation of different AL strategies and (2) the definition of best algorithms for effective personalization.

3. METHODOLOGY

The main contribution of our work is to address open questions by proposing query strategies that involve collective judgement for personalization and evaluating diverse algorithms, later introduced in Section 3.3. We use a different query strategy to build upon the work by Su and Fung [13], and propose a novel method to account for the collective judgement – differing from traditional instance selection for AL. Our work is also motivated by the multi-oracle AL paradigm [20–22] in order to exploit this judgement. However, instead of picking an expert/confident annotator, we select instances which are ambiguous to the crowd – different to those ambiguous to the algorithms. We introduce *consensus entropy* [26] to AL for MER with a three-fold perspective: (1) analyzing the agreement achieved by a committee of pre-trained models (*machine consensus - MC*), (2) analyzing the agreement from a committee of annotators (*human consensus - HC*), and (3) taking into account both committees (*hybrid consensus - MIX*). Our work differs from [25] since we obtain personalization by sampling informative instances and re-training the algorithms, instead of progressively adapting model parameters. Our main assumption is that prior knowledge about the uncertainty of an excerpt with respect to the collective judgement (i.e., human consensus), results in the particular instances which could be indicative of classification boundaries across individual listeners. Music excerpts on which we disagree upon define our personal opinions and should be taken into account for personalization. Secondly, we assume that the confusion between perceived and induced emotions is mainly static and will not vary over time (see [27] for a study on intra-rater agreement), hence personalization could lead to models that can predict both types of emotion and work must be done to determine the type of emotion [7, 8]. To the best of our knowledge, the use of the collective judgement as a personalization strategy has never been explored in MER so far.²

3.1 Data

Despite the complexity and difficulty of obtaining music emotion annotations, researchers in MER have made great efforts to create open datasets.³ To pre-train our classifiers, we used the DEAM dataset [28]. The benchmark

¹ We refer the reader to [14, 16–19] for a comprehensive overview of AL methods.

² <https://github.com/juansgomez87/consensus-entropy>

³ Data from the study in [13] is not openly available.

dataset for MER, DEAM was constructed across several MediaEval contests (2013–2015), and contains 1802 music excerpts and dynamic arousal and valence annotations (introduced by Russell [29]). We discretized annotations into four quadrants for classification, following [30]: Q1 (positive valence and arousal, A+V+), Q2 (positive arousal and negative valence, A+V-), Q3 (negative valence and arousal, A-V-), Q4 (negative arousal and positive valence, A-V+).⁴ To test personalization, we used the AMG1608 dataset [35]. This dataset was previously used for personalization purposes [24,25], and is composed of 1608 music excerpts rated with static arousal-valence annotations from 665 listeners (22 annotators from the campus of the National Taiwan University and 643 from Amazon Mechanical Turk). From the pool of annotators, we use the subset of 46 annotators that rated more than 150 songs (from which 10 belong to the campus subset).

We used two feature sets depending on the classification algorithm (see Section 3.3): (1) low-level, emotionally-relevant features for classic machine learning algorithms, and (2) mel-spectrograms for novel convolutional neural network architectures. As to (1), the IS13 ComParE feature set [36] has been widely used for sound, speech, and music emotion recognition. We extracted 260 features (mean and standard deviation of 65 low-level music descriptors and their first order derivatives) from segments of 1 second [28], with 50% overlap, and standardize across features – using OpenSMILE [37]. In order to test our approach on novel deep learning architectures we extracted mel-spectrograms, based on [38]: we downsampled audio to 16kHz, performed a Short-Time Fourier Transform (window size: 512 samples \sim 23ms; hop size: 256 \sim 12ms), and extracted a mel-scale spectrogram with 128 mel-bands – using Librosa [39].

3.2 Consensus entropy

Consensus entropy is a combination of uncertainty sampling and query-by-committee methods as follows [16,26]: (1) a committee of classifiers predicts the output probabilities of unlabeled data, (2) probabilities are averaged across the committee of classifiers, (3) uncertainty is calculated as Shannon’s entropy across classes for each instance, (4) q instances with highest entropy are selected to be annotated by the oracle, and (5) classifiers are re-trained with the provided annotations. For example, full disagreement from a committee of four classifiers results when each one predicts a different quadrant with 100% probability. This yields average probabilities per quadrant $p_{avg} = \{Q1 : 0.25, Q2 : 0.25, Q3 : 0.25, Q4 : 0.25\}$ and high inter-class entropy/uncertainty of 1.386. We refer to this approach as *machine consensus (MC)*. Secondly, studies have shown evidence of the impact of inter-rater agreement on the performance of MER algorithms [10,12]. Hence, we propose *human consensus (HC)* as a variation from classical consensus entropy: we calculate entropy on the normalized annotation histogram per song. For exam-

ple, given 6 annotators for song i , we obtain a relative frequency $f_i = \{Q1 : 1/6, Q2 : 2/6, Q3 : 3/6, Q4 : 0/6\}$. Thirdly, we combine the strategies for a *hybrid consensus (MIX)* by stacking the probabilities and relative frequencies, and calculating the overall entropy.

The proposed method is summarized in Algorithm 1: let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^m$ represent the pre-training data (DEAM) consisting of m labeled instances (1802 excerpts) and $\mathcal{U} = \{(x_i)\}_{i=m+1}^n$ represent the "unlabeled" data for personalization (AGM1608). Since $\{(y_i)\}_{i=m+1}^n$ for \mathcal{U} are already present in the dataset, we query q excerpts and fine-tune with their annotations. We consider x_i an input feature (low-level feature vector or mel-spectrogram), and $y_i \in C = \{Q1, Q2, Q3, Q4\}$ as the annotated quadrant. Finally, we split annotated data by each user $u_j = \{0 \dots 45\}$ with more than 150 annotations: 85% for training and 15% for testing, with no overlapping music excerpts. We denote $P_{\mathcal{L}, M_k}(y_i|x_i)$ as the conditional probability of y given x according to a classifier M_k trained on \mathcal{L} . Notice that $P_{\mathcal{L}}(y_i|x_i)$ is the probability averaged across all models M_k . We performed 10 iterations and queried $q = 10$ instances per iteration.⁵ Following Chen et al. [25], we used random selection as a baseline.

Algorithm 1: Consensus entropy for MER.

```

input : Labeled data  $\mathcal{L}$ , unlabeled data  $\mathcal{U}$ 
Pre-train each model  $M_k$  on  $\mathcal{L}$ ;
for each iteration  $it = \{0 \dots 9\}$  do
    for each user  $u_j$  do
        Calculate  $P_{\mathcal{L}, M_k}(y_i|x_i)$  for each  $x_i \in \mathcal{U}$ ;
        if MC then
            Average  $P_{\mathcal{L}}(y_i|x_i)$  across frames and  $M_k$  models;
            Select  $q$  excerpts with highest entropy;
        else if HC then
            Calculate relative frequency  $f_i$  per music excerpt;
            Select  $q$  excerpts with highest entropy;
        else if MIX then
            Calculate and stack  $P_{\mathcal{L}}(y_i|x_i)$  and  $f_i$ ;
            Select  $q$  excerpts with highest entropy;
        else
            Select  $q$  random excerpts;
        Annotate  $q$  instances by  $u_j$ ;
        for each model  $M_k$  do
            for each  $(x_i, y_i) \in q$  do
                Re-train  $M_k$  on  $\mathcal{L} \cup (x_i, y_i)$ ;
                Compute metrics on test data;
                Update  $\mathcal{L} \leftarrow \mathcal{L} \cup (x_i, y_i)$  and  $\mathcal{U} \leftarrow \mathcal{U} \setminus (x_i, y_i)$ ;
            end
        end
    end
end
    
```

⁴ We refer the reader to [10] for a concise explanation of music emotion taxonomies. See also [31–34] for in-depth theory.

⁵ Tests using $q = 15$ and $q = 40$ reduced the amount of available users (i.e., each user annotated a different amount of excerpts). We chose $q = 10$ to match the study in [25]: 46 users with over 150 annotations.

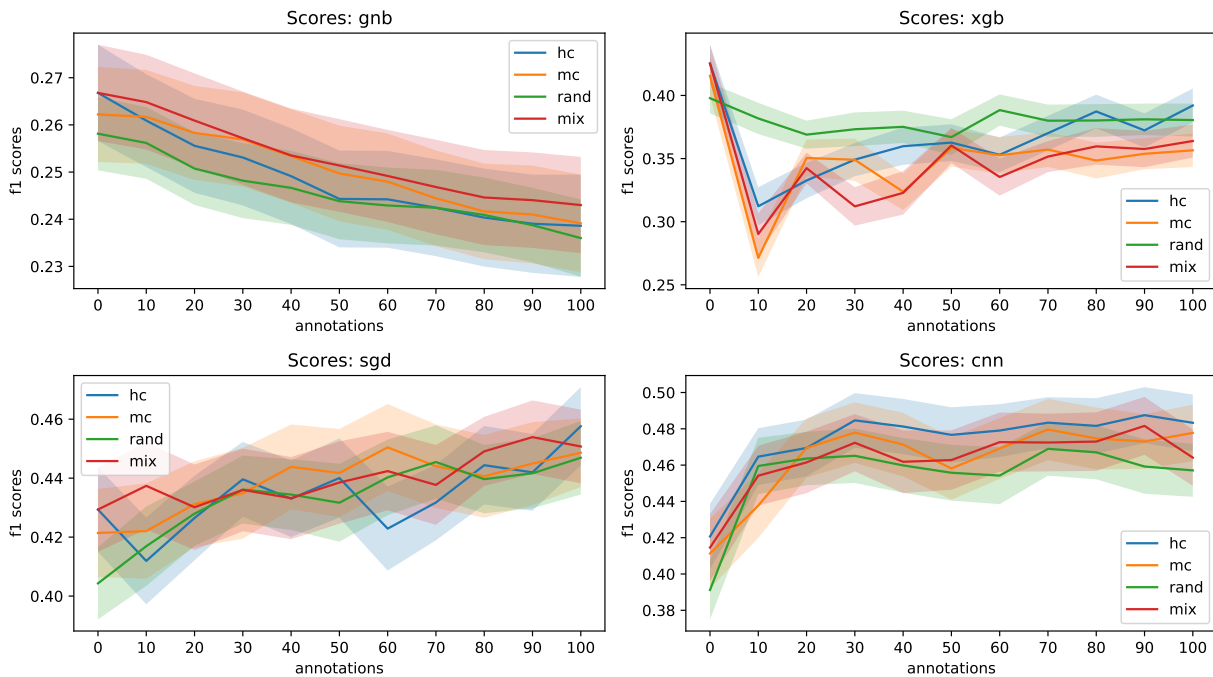


Figure 1. Average results of weight-averaged F1-scores for each type of model, across 46 users and 5 classifiers (shaded area corresponds to $CI = 95\%$, $n = 230$). HC stands for Human Consensus, MC for machine consensus, MIX for hybrid consensus and RAND for random selection.

3.3 Algorithms

Since the query strategy requires a committee of classifiers, we pre-trained all the following models with the DEAM dataset using 5-fold cross validation – each classifier is pre-trained on general annotations while still resulting in diverse predictions, in order to analyze agreement amongst classifiers. For each algorithm $m_k = \{0 \dots 4\}$, we obtained 5 classifiers for a total of 20 models per user. We used four algorithms in this study – based on (1) computational efficiency and low memory cost, and (2) well-established and novel approaches in the state-of-the-art – and introduce them as follows:

Gaussian Naive Bayes (GNB). These algorithms are based on the "naive" assumption of independence between pairs of features, given a class label [40]. Bayes' theorem relates the conditional probability of the output y and the dependent feature vectors x_i . The likelihood of the features is assumed to be normal-distributed, hence the models are Gaussian. Priors are adjusted according to the data and variance smoothing is set to $1e-9$.

Extreme Gradient Boosting (XGB). This widely used machine learning method is based on the idea of gradient tree boosting: an ensemble of weak learners (i.e., regression trees) is optimized to minimize a given loss function [41]. In contrast to other gradient boosting algorithms, XGB is well-established given its scalability and training speed. We performed a minor change to allow re-fitting the algorithm and set parameters empirically during pre-training: the maximum depth of 5 for each decision tree.

Logistic Regression (SGD). We used a model that optimizes a log-loss function with L2 regularization to output

class probabilities – obtaining a Logistic Regression classifier fitted using Stochastic Gradient Descent (SGD). SGD is an optimization method to fit linear classifiers using convex loss functions [42].

Short-chunk Convolutional Neural Network (CNN). In the field of automatic audio tagging, Won et al. [38] have recently proposed a 7-layer 2D convolutional neural network that processes chunks of 3.69s of audio and (2×2) max-pooling layers to summarize the chunk into a single dimension. A mixture of scheduled Adam [43] and SGD are used as optimization methods, following [44]. We pre-trained models for 200 epochs and re-trained for 100 epochs – best models were selected when the validation loss improved.

4. RESULTS

Figure 1 shows the weighted-average F1-scores on the test data averaged across 46 users, averaging across each algorithm for a total of 3680 trained classifiers ($46 \text{ users} \times 4 \text{ algorithms} \times 5 \text{ models per pre-training split} \times 4 \text{ consensus entropy methods}$). We report weighted average scores since datasets are class-imbalanced.

4.1 Algorithms and consensus entropy methods

Firstly, we use pairwise, one-sided t-tests ($d.f. = 229$, statistical significance $p < 0.05$) in order to evaluate differences among consensus entropy methods (i.e., HC, MC, MIX, and RAND) for each particular model after 100 annotations (at least 150 annotations are available per user), as evaluated by Chen et al. [25]. We do not perform other

statistical tests (i.e., McNemar’s Test or Wilcoxon signed-rank test), as proposed by Demšar [45], since each user has annotated different songs (i.e., the training and testing data is not the same between users).

Gaussian Naive Bayes (GNB). These classifiers appear to diminish their performance with more annotations which is expected of naive bayesian models (i.e., limited generalization to new data) – MIX appears to outperform the random baseline by ~ 1 percent point. However, none of the comparisons between methods is statistically significant ($p > 0.147$).

Extreme Gradient Boosting (XGB). These classifiers display an expected behavior: random selection results in limited variation throughout 100 annotations, while other methods (MC, HC, and MIX) suffer a significant fall with the initial re-training data and increasingly improve with the amount of annotations. In this case, HC is significantly better than MC ($p = 0.0001$) and than MIX ($p = 0.0017$), but does not outperform RAND.

Logistic Regression (SGD). These classifiers exhibit increasing performance with more data – the HC method outperforms the random baseline by ~ 1 percent points. Again, none of the pairwise comparisons show significant differences across cross entropy methods ($p > 0.125$).

Short-chunk Convolutional Neural Network (CNN). Classifiers exhibit a significant increase with initial re-training data – the HC method again outperforms the random baseline by ~ 2 percent points. Interestingly, these classifiers display the best performance across all models with cases of high f1-scores (approximately 0.7-0.8 for particular users – see Figure 2). HC is significantly better than RAND ($p = 0.00811$) and than MIX ($p = 0.044$), while MC is better than RAND ($p = 0.0291$). Similar to results reported by Chen et al. [25], these classifiers appear to improve after 20-30 annotations and plateau.

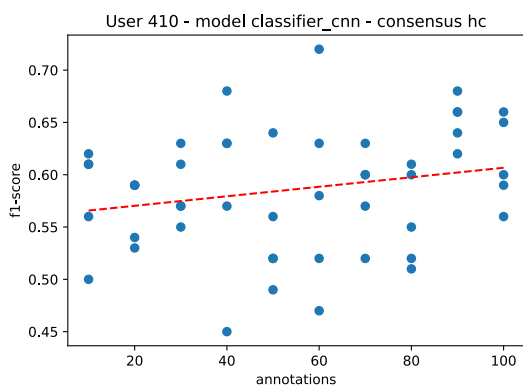


Figure 2. F1-scores of five CNN classifiers from user 410 using the HC consensus entropy method. Each point represents the F1-score for each classifier on the user’s test data.

4.2 Campus subset

Given the impact of agreement on classification performance, Chen et al. [25] split the users into two groups:

the general pool of 46 annotators and subset of 10 annotators from campus (as mentioned in Section 3.1). We perform the same analysis for this subset of annotators (*d.f.* = 49, statistical significance $p < 0.05$). With respect to the subset of campus annotators, the general tendencies mentioned in Section 4.1 appear to hold, yet the difference of performance between the proposed methods and the random baseline appears to narrow.⁶ For the XGB model, HC significantly outperforms MIX ($p = 0.0149$). For the CNN model, HC significantly outperforms RAND ($p = 0.0254$) and MIX ($p = 0.0152$).

4.3 Effective personalization

Although the proposed methods marginally outperform the random selection baseline in the general behavior across all users, we observe diverse behaviors when analyzing each user: (1) XGB and SGD classifiers exhibit less variation across each algorithm than GNB and CNN – XGB and SGD classifiers appear to be more stable with respect to each re-training iteration, and (2) models do not necessarily improve with more annotations – it is likely that the annotations of a particular user are not producing personalization.⁷ Thus, we tested evaluating each user’s algorithms as seen in Figure 2 and fitted a linear regression (using Ordinary Least Squares) to estimate if the average metrics from the ensemble of classifiers indeed improved as more personal annotations are presented. Namely, when the slope of the lineal regressor is positive, we assume that "effective" personalization has been achieved – as more personal annotations are presented, the algorithm improves performance on the test data. Figure 3 summarizes the results of the amount of personalized models following this assumption: (1) GNB classifiers are rarely producing personalized models, (2) SGD classifiers appear to produce the same number of personalized models regardless the consensus entropy method (slightly more personalization is achieved with HC), and (3) for both XGB and CNN classifiers all proposed consensus entropy methods appear to produce more personalized models than RAND.

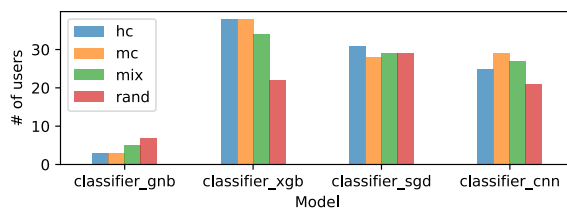


Figure 3. Number of users with effective personalization per algorithm from a total of 46 users.

5. DISCUSSION AND CONCLUSIONS

5.1 Discussion

Our study is inspired by the work from Su and Fung [13], in which AL was used to progressively re-train MER mod-

⁶ Refer to Figure 1 from supplementary material.
⁷ Refer to Figures 2-4 from supplementary material.

els with informative and representative data instances to produce such personalized models. We also are encouraged by studies from Bullard et al. [46] in which the traditional approach of AL is challenged in favor of "realistic human interaction": instead of providing the algorithms an *optimal query strategy* [14], we attempt to *put the human-in-the-loop* by grounding emotion concepts based on community judgements and simple inter-rater agreement. We aimed at using the collective judgement of the pool of annotators as prior knowledge for AL – our main assumption is that highly uncertain instances in the collective judgement reflect individual boundaries of classification that should be used to personalize MER models. Thus, we propose two consensus entropy methods for AL based on the classical uncertainty sampling and query-by-committee strategies: (1) *human consensus (HC)* that uses the pool of annotators as the committee to obtain informative samples, and (2) *hybrid consensus (MIX)* that considers possible complementary advantages from HC and MC.

Regarding *RQ1* - Can we exploit human agreement in music emotion annotations as input to AL methodologies to produce personalized models? Our findings suggest that our proposed methods appear to improve personalization with respect to a baseline that presents random instances for re-training. Particularly, the proposed HC method outperforms the methods presented by Su and Fung [13], which rely on using uncertainty sampling to compare most informative (highest entropy - MC) data instances for personalization for 8 users. Their study reports average F1-scores of $\mu = 0.35, \sigma = 0.30$ after using AL. Our study shows the following F1-scores from 46 users: **CNN** – $\mu = 0.48, \sigma = 0.12$, **XGB** – $\mu = 0.39, \sigma = 0.10$, **SGD** – $\mu = 0.457, \sigma = 0.10$, **GNB** – $\mu = 0.238, \sigma = 0.08$. Additionally, the MIX method marginally outperforms the random baseline, showing similar performance to the MC method – the MIX method is likely querying similar instances as the MC method for each iteration.

With respect to *RQ2* - What is the impact of the choice of classification algorithm on personalization of MER systems? We tested our method on four types of algorithms, which display different behaviours: (1) Gaussian Naive Bayes classifiers (GNB) appear not to generalize to new data or work for personalization – Naive Bayes assumes independence of predictors which is not likely the case for overlapping emotionally-relevant features, (2) Logistic Regression (SGD) appears to produce personalized models but there is no significant difference across the consensus entropy methods – the assumption of linearity between features and annotations is not likely to capture more complex relationships from features, (3) Extreme Gradient Boosting classifiers (XGB) appear to produce the highest amount of personalized models – however, results suggest that these models require more annotations in order to eventually surpass the performance metrics from the random selection baseline, and (4) Short-chunk Convolutional Neural Network (CNN) appears to produce the best classification performance and the HC method appears to produce more personalized models – yet the "black box" nature of neural

networks might hinder the interpretability and explainability of using these models.

In addition to the fact that our findings are limited by the datasets and the methodologies used to build and annotate them, we present three main limitations to be considered:

Inter-rater agreement. Previous studies [10,11,27,47] have evaluated inter-rater agreement as defined by Krippendorff's coefficient α [48]. However, it is not possible to use this coefficient to assess agreement for the HC method since the annotations are categorical. Only one coefficient can be calculated for arousal or valence over the complete dataset (or dataset subset). Nonetheless, the relative frequency (HC) can be interpreted as an empirical probability that is informative with respect to simple agreement.

Interpretability of the MC method. The lack of agreement between the classifiers might be due to other factors different than the difficulty of the "ground truth". In this sense, acoustic properties and the impact of the features on predictions might produce confounding factors for the classifiers and will be considered as future work.

Stasis of the HC method. HC is mainly static as opposed to MC approach: the method is restricted by the amount of annotated songs and number of users. Thus, the songs that result from each query will be the same for all users, as opposed to the MC approach. In the case of MC, every time a model is re-trained the classification boundaries are adjusted along with the uncertainty of new particular instances. Although the underlying principles of HC and MC are quite different, the expectation of complementary advantages over each other was not met.

5.2 Conclusions

To the extent of our knowledge, the proposed methodology has not been used for the MER task or other MIR use cases, since the classic aim of AL is to make the data collection less burdensome (i.e., reduce the workload of the annotation procedure). In the context of producing user-centric MIR [49], we argue that using knowledge about the collective consensus could be beneficial for other tasks with low inter-rater agreement: music auto-tagging [50], music similarity [11, 27], automatic chord estimation [51], and beat tracking [52]. Indeed, current streaming services and social media constantly produce high amounts of diverse responses in tagging environments – which could be beneficial to test our methodology on other tasks. For the particular field of MER, building a collective "ground truth" by merely averaging ratings across annotators might be oversimplifying what has recently been questioned in neuroscience research on emotions by Barrett [53]:

"One instance of [an emotion] need not look or feel like another, nor will it be caused by the same neurons [in the brain]. *Variation is the norm.* Your range of [an emotion] is not necessarily the same as mine, although if we were raised in similar circumstances, we will likely have some overlap."

For once, we could simply agree to disagree.

6. ACKNOWLEDGEMENTS

The research work conducted in the Music Technology Group at the Universitat Pompeu Fabra is partially supported by the European Commission under the TROMPA project (H2020 770376). We would like to thank anonymous reviewers that helped us improve the paper with constructive feedback.

7. REFERENCES

- [1] Y.-H. Yang and H. H. Chen, *Music Emotion Recognition*. CRC Press, 2011.
- [2] N. N. Vempala and F. A. Russo, “Modeling music emotion judgments using machine learning methods,” *Frontiers in Psychology*, vol. 8, 2018.
- [3] A. Gabrielsson, “Emotion perceived and emotion felt: Same and different,” *Musicae Scientiae*, vol. 10, no. 2, pp. 191–213, 2006.
- [4] J. Cespedes-Guevara and T. Eerola, “Music communicates affects, not basic emotions - A constructionist account of attribution of emotional meanings to music,” *Frontiers in Psychology*, vol. 9, no. FEB, pp. 1–19, 2018.
- [5] L. A. Warrenburg, “Comparing musical and psychological emotion theories,” *Psychomusicology: Music, Mind, and Brain*, vol. 30, no. 1, pp. 1–19, 2020.
- [6] —, “Choosing the right tune: A review of music stimuli used in emotion research,” *Music Perception*, vol. 37, no. 3, pp. 240–258, 2020.
- [7] J. S. Gómez-Cañón, N. Gutiérrez-Páez, L. Porcaro, A. Gkiokas, P. Herrera, and E. Gómez, “Improving emotion annotation of music using citizen science,” in *Proceedings of the 16th International Conference on Music Perception and Cognition (ICMPC/ESCOM)*, Sheffield, United Kingdom (virtual), 2021.
- [8] N. Gutiérrez-Páez, J. S. Gómez-Cañón, L. Porcaro, P. Santos, D. Hernandez-Leo, and E. Gómez, “Emotion annotation of music: a citizen science approach,” in *Proceedings of the 27th International Conference on Collaboration Technologies and Social Computing (CollabTech)*, Trier, Germany (virtual), 2021.
- [9] Y.-H. Yang, Y.-F. Su, Y.-C. Lin, and H. H. Chen, “Music Emotion Recognition: The Role of Individuality,” in *Proceedings of the International Workshop on Human-centered Multimedia (HCM)*, 2007, pp. 13–22.
- [10] J. S. Gómez-Cañón, E. Cano, P. Herrera, and E. Gómez, “Joyful for you and tender for us: the influence of individual characteristics and language on emotion labeling and classification,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020, pp. 853–860.
- [11] A. Flexer and T. Grill, “The Problem of Limited Inter-rater Agreement in Modelling Music Similarity,” *Journal of New Music Research*, vol. 45, no. 3, pp. 239–251, 2016.
- [12] X. Hu and Y.-H. Yang, “Cross-dataset and cross-cultural music mood prediction: A case on Western and Chinese Pop songs,” *IEEE Transactions on Affective Computing*, vol. 8, no. 2, pp. 228–240, 2017.
- [13] D. Su and P. Fung, “Personalized music emotion classification via active learning,” in *Proceedings of the 2nd International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies*, New York, NY, USA, 2012, p. 57–62.
- [14] B. Settles, *Active Learning*. Morgan and Claypool Publishers, 2012.
- [15] Á. Sarasúa, C. Laurier, and P. Herrera, “Support Vector Machine Active Learning for Music Mood Tagging,” in *Proceedings of the 9th International Symposium on Computer Music Modelling and Retrieval (CMMR)*, London, England, 2012, pp. 518–525.
- [16] B. Settles, “Active learning literature survey,” University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.
- [17] C. C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. S. Yu, “Active Learning: A Survey,” in *Data Classification: Algorithms and Applications*. CRC Press, 2014, pp. 571–605.
- [18] Y. Yang, “Towards Practical Active Learning for Classification,” Ph.D. dissertation, TU Delft University, 2018.
- [19] P. Kumar and A. Gupta, “Active learning query strategies for classification, regression, and clustering: A survey,” *Journal of Computer Science and Technology*, vol. 35, no. 4, pp. 913–945, Jul 2020.
- [20] Y. Yan, R. Rosales, G. Fung, and J. G. Dy, “Active Learning from Crowds,” in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, USA, 2011.
- [21] V. S. Sheng, F. Provost, and P. G. Ipeirotis, “Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers,” in *Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, Las Vegas, USA, 2008, pp. 614–622.
- [22] P. Donmez, J. G. Carbonell, and J. Schneider, “Efficiently Learning the Accuracy of Labeling Sources for Selective Sampling General Terms,” in *Proceedings of the 15th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, Paris, France, 2009, pp. 259–267.

- [23] J. Zhang, X. Wu, and V. S. Sheng, "Learning from crowdsourced labeled data: a survey," *Artificial Intelligence Review*, vol. 46, pp. 543–576, 2016.
- [24] Y.-A. Chen, J.-C. Wang, Y.-H. Yang, and H. Chen, "Linear Regression-based Adaptation of Music Emotion Recognition Models for Personalization," in *Proceedings of the IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*, 2014, pp. 2149–2153.
- [25] Y.-A. Chen, J.-C. Wang, Y.-H. Yang, and H. H. Chen, "Component tying for mixture model adaptation in personalization of music emotion recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 7, pp. 1409–1420, 2017.
- [26] D. Cohn, L. Atlas, R. Ladner, and A. Waibel, "Improving Generalization with Active Learning," *Machine Learning*, vol. 15, pp. 201–221, 1994.
- [27] A. Flexer and T. Lallai, "Can we increase inter- and intra-rater agreement in modeling general music similarity?" in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 494–500.
- [28] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Developing a benchmark for emotional analysis of music," *PLoS One*, pp. 1–22, 2017.
- [29] J. A. Russell, "A circumplex model of affect," *Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.
- [30] R. Panda, R. M. Rui, and P. Paiva, "Musical texture and expressivity features for music emotion recognition," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.
- [31] P. N. Juslin, *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford: Oxford University Press, 2010.
- [32] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," *Psychology of Music*, vol. 39, no. 1, pp. 18–49, 2011.
- [33] T. Eerola, "Music and emotion," in *Handbook of Systematic Musicology*, R. Bader and S. Koelsch, Eds. Springer, 2018, ch. Music and Emotion, pp. 539–556.
- [34] P. N. Juslin, *Musical Emotions Explained*. Oxford: Oxford University Press, 2019.
- [35] Y. Chen, Y. Yang, J. Wang, and H. Chen, "The AMG1608 dataset for music emotion recognition," in *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 693–697.
- [36] F. Weninger, F. Eyben, B. W. Schuller, M. Mortillaro, K. R. Scherer, and J. Krajewski, "On the acoustics of emotion in audio: what speech, music, and sound have in common," *Frontiers in Psychology*, vol. 4, pp. 1–12, 2013.
- [37] F. Eyben, F. Weninger, F. Gross, and B. Schuller, "Recent Developments in OpenSMILE, the Munich Open-source Multimedia Feature Extractor," in *Proceedings of the 21st ACM International Conference on Multimedia*, New York, NY, USA, 2013, pp. 835–838.
- [38] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," in *Proceedings of 17th Sound and Music Computing Conference (SMC)*, 2020.
- [39] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference (SCIPY)*, Austin, USA, 2015, pp. 18–25.
- [40] H. Zhang, L. Jiang, and J. Su, "The optimality of naive bayes," in *Proceedings of the 17th Florida Artificial Intelligence Research Society Conference*, 2004, pp. 562–567.
- [41] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Aug 2016.
- [42] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT)*, 2010, pp. 177–186.
- [43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.
- [44] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.
- [45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [46] K. Bullard, Y. Schroecker, and S. Chernova, "Active Learning within Constrained Environments through Imitation of an Expert Questioner," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.
- [47] S. Yang, M. Barthet, and E. Chew, "Multi-scale analysis of agreement levels in perceived emotion ratings during live performance," in *Late-Breaking Demo of the 18th International Society for Music Information Retrieval (ISMIR)*, Suzhou, China, 2017.

- [48] K. H. Krippendorff, *Content Analysis: An Introduction to Its Methodology*, 2nd ed. SAGE Publications, 2004.
- [49] M. Schedl, E. Gómez, E. S. Trent, M. Tkalcic, H. Eghbal-Zadeh, and A. Martorell, “On the interrelation between listener characteristics and the perception of emotions in Classical orchestra music,” *IEEE Transactions on Affective Computing*, vol. 9, no. 4, pp. 507–525, 2018.
- [50] E. Bigand and J.-J. Aucouturier, “Seven problems that keep MIR from attracting the interest of cognition and neuroscience,” *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 483–497, 2013.
- [51] H. V. Koops, W. Bas De Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, “Annotator subjectivity in harmony annotations of popular music,” *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019.
- [52] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, “Selective sampling for beat tracking evaluation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.
- [53] L. F. Barrett, *How Emotions are Made: The Secret Life of the Brain*. Houghton Mifflin Harcourt, 2017.

SEQUENCE-TO-SEQUENCE PIANO TRANSCRIPTION WITH TRANSFORMERS

Curtis Hawthorne Ian Simon Rigel Swavely
Ethan Manilow* Jesse Engel

Google Research

{fjord,iansimon,rigeljs,emanilow,jesseengel}@google.com

ABSTRACT

Automatic Music Transcription has seen significant progress in recent years by training custom deep neural networks on large datasets. However, these models have required extensive domain-specific design of network architectures, input/output representations, and complex decoding schemes. In this work, we show that equivalent performance can be achieved using a generic encoder-decoder Transformer with standard decoding methods. We demonstrate that the model can learn to translate spectrogram inputs directly to MIDI-like output events for several transcription tasks. This sequence-to-sequence approach simplifies transcription by jointly modeling audio features and language-like output dependencies, thus removing the need for task-specific architectures. These results point toward possibilities for creating new Music Information Retrieval models by focusing on dataset creation and labeling rather than custom model design.

1. INTRODUCTION

Automatic Music Transcription (AMT) is one of the core tasks of Music Information Retrieval (MIR). The objective of AMT is to convert raw audio to a appropriate symbolic representation. In this paper we consider the problem of transcribing piano audio to a series of note events indicating precise onset/offset timings and velocities, as opposed to a sheet music score that is aligned to a metrical grid.

Recent progress in piano transcription has been largely driven by two factors: the construction and release of datasets containing aligned piano audio and MIDI (most notably MAPS [1] and MAESTRO [2]) and the use of deep neural networks with architectures specifically designed for piano transcription (e.g., the Onsets and Frames architecture that models note onsets and note presence separately [3]). While domain-specific models have lead to improvements on benchmark datasets, it is not clear if these

approaches can translate to other domains and MIR tasks.

Simultaneously, Transformer models employing self-attention [4] have demonstrated a surprising ability to achieve state-of-the-art results in a variety of domains with the same core architecture, by simply varying the input and output representations [5–14].

In this paper, we demonstrate that a generic Transformer model can achieve state-of-the-art piano transcription without any domain-specific adaptations. Using “off-the-shelf” components (an essentially unmodified encoder-decoder configuration from the T5 paper [6]) and a simple greedy decoding strategy, we train a model to encode raw spectrogram frames and decode directly to a sequence of note events inspired by messages in the original MIDI protocol [15] (e.g., `note on` and `velocity` messages). As such, we refer to the model’s output as “MIDI-like” throughout the rest of this paper. We provide details of our model’s vocabulary in Section 3.2.

Further, we demonstrate that this domain-agnostic approach enables us to train several variations on the transcription task (e.g., transcribing only note onsets) by changing only the training labels and without modifying the inputs or model.

In summary, this work illustrates the value of using generic sequence-to-sequence Transformers for piano transcription, without domain-specific adaptations, and points to the potential to expand a similar approach to a variety of MIR tasks.

2. RELATED WORK

2.1 Piano Transcription

Much progress has been made in piano transcription using deep neural network models trained on datasets of aligned audio and MIDI. In 2012, Boulanger-Lewandowski et al. [16] (building off the acoustic model of Nam et al. [17]) trained a recurrent neural network (RNN) transcription model to output a binary piano roll. Böck and Schedl [18] trained a similar RNN-based model for piano onsets only. Hawthorne et al. [3] improved transcription accuracy by having separate convolution-based model *stacks* for detecting note onsets, note presence, and note velocities. Model outputs were decoded into discrete notes using a hard prior by not initiating a note unless the onset predictor gave probability more than 0.5.

* Work done as a Google Brain Student Researcher.



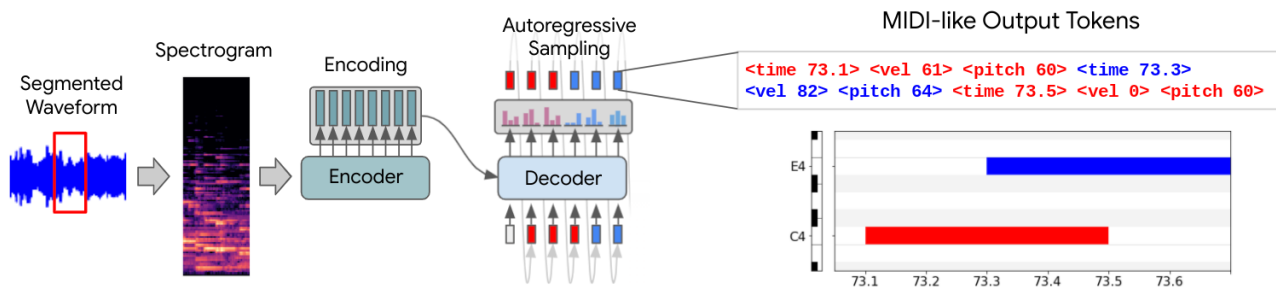


Figure 1. Our model is a generic encoder-decoder Transformer architecture where each input position contains a single spectrogram frame and each output position contains an event from our MIDI-like vocabulary. Outputs tokens are autoregressively sampled from the decoder, at each step taking the token with maximum probability.

More recently, progress on piano transcription has largely involved adding more domain-specific deep neural network components and modifying the decoding process. For the most part, this additional complexity has been geared toward the specific purpose of improving piano transcription accuracy.

Kong et al. [19] achieve higher transcription accuracy by using regression to predict precise continuous onset/offset times, using a similar network architecture to Hawthorne et al. [3]. Kim & Bello [20] use an adversarial loss on the transcription output to encourage a transcription model to output more plausible piano rolls. Our sequence-to-sequence approach explicitly models such inter-output dependencies through the autoregressive decoder, which is trained end-to-end with the encoder that extracts meaningful audio features.

Kwon et al. [21] use a language model of sorts to model per-pitch note state transitions instead of having separate onset, frame, and offset stacks. However, the decoding process is fairly complex, in particular the handling of interactions between different pitches. Similarly, Kelz et al. [22] decode using a hidden Markov model over note states based on attack-decay-sustain-release (ADSR) envelopes.

In a very thorough domain-specific treatment, Elwesson [23] constructs a hierarchical model that extracts fundamental frequency contours from spectrograms and uses these contours to infer note onsets and offsets. While it can be useful for many applications to have such intermediate representations as in Engel et al. [24], in this work we treat polyphonic transcription from audio to discrete notes as an end-to-end problem. This has the advantage of conceptual simplicity and our evaluation (Section 4.2) shows it is also effective.

2.2 Transformers

Recently, a generic Transformer architecture [4] has been used across multiple domains to solve sequence-to-sequence problems, replacing task-specific architectures that had previously been in use. Outside the field of natural language processing in which Transformers initially emerged and are now widely used (e.g., GPT-3 by Brown et al. [25] and T5 by Raffel et al. [6]), Transformers have

been used in computer vision for tasks such as object detection [8], caption-based image generation [9], and pose reconstruction [10], as well as audio-related tasks including speech recognition [11, 12], speech synthesis [13], and audio event classification [14].

Note that many of the above uses of Transformer take advantage of a *pretraining* phase where the model is trained on a large amount of unlabeled data using self-supervision. While it is possible that such a pretraining phase might also help with music transcription, in this work we explore the simpler setting of training the Transformer architecture from scratch on labeled transcriptions in an ordinary supervised fashion.

2.3 Sequence-to-Sequence Transcription

The idea of using Transformers for music transcription has also been considered. Awiszus in 2019 [26] explored several formulations of music transcription as a sequence-to-sequence problem, using a variety of input and output representations (including ones similar to our own) with both LSTM [27] and Transformer models. However, the paper was unable to demonstrate clear success, seemingly due to using a framewise multi-F0 evaluation rather than the note-based evaluation standard in piano transcription, using relative time shifts rather than absolute (see Section 3.2), and training on the MAPS dataset which is much smaller than the MAESTRO dataset we use. Earlier, Ullrich and van der Wel [28] appear to be the first to have posed music transcription as a sequence-to-sequence problem (using LSTMs instead of Transformers), but their system could handle only monophonic music.

3. MODEL

As mentioned above, our model is a generic encoder-decoder Transformer architecture where each input position contains a single spectrogram frame and each output position contains an event from our MIDI-like vocabulary. An overview of our model and our input and output setup is shown in Figure 1.

Inputs are processed through a stack of encoder self-attention layers resulting in a sequence of embeddings the same length as the original input. A stack of decoder lay-

ers then uses both causally masked self-attention over the decoder output and cross-attention over the full output of the encoder stack. Crucially, this allows the symbolic token output to be variable length, dependent only on the number of tokens needed to describe the input audio.

3.1 Model Architecture

The model configuration is based on the “small” model from T5 [6], with modifications as suggested by the T5.1.1 recipe¹. Specifically, our model uses an embedding size of $d_{\text{model}} = 512$, a feed-forward output dimensionality of $d_{\text{ff}} = 1,024$, a key/value dimensionality of $d_{\text{kv}} = 64$, 6-headed attention, and 8 layers each in the encoder and decoder.

Our model has a few minor changes from the standard configuration. Most important is that in order to use continuous spectrogram inputs, we add a dense layer to project each spectrogram input frame to the Transformer’s input embedding space. We also use fixed absolute positional embeddings rather than the logarithmically scaled relative positional bucket embeddings used in T5, to ensure all positions can be attended to with equal resolution. Finally, we use `float32` activations for better training stability because our model is small enough that we do not need the memory efficiency of the less precise `bfloat16` format [29] typically used in large T5 models.

The model is implemented using the T5X framework², which is built on Flax [30] and JAX [31]. We also use SeqIO³ for data preprocessing and evaluation. Code for our implementation will be available at <https://goo.gl/magenta/seq2seq-piano-transcription-code>.

While some recent research using Transformers has favored very large models, such as GPT-3 [25] with 175B parameters, we found that a comparatively small model is sufficient for these tasks. With the configuration described above, our model has only 54M parameters, only roughly twice that of Onsets and Frames [2], which has 28M parameters.

3.2 Inputs and Outputs

The model uses spectrogram frames as input, with one frame per input position. To match the T5 setup, we terminate input sequences with a learnable EOS (End of Sequence) embedding. The model output at each step is a softmax distribution over a discrete vocabulary of events, described below. This vocabulary is heavily inspired by the messages originally defined in the MIDI specification [15]. Using events as the output representation instead of a piano roll matrix has the advantage of being much more sparse because outputs are needed only when an event occurs, instead of needing to annotate every frame. The vocabulary consists of the following token types:

Note [128 values] Indicates a note-on or note-off event for one of the 128 MIDI pitches. We use the full MIDI pitch range for flexibility, but for these experiments, only the 88 pitches corresponding to piano keys are actually used.

Velocity [128 values] Indicates a velocity change to be applied to all subsequent Note events (until the next Velocity event). There are 128 velocity values including zero, a special value which causes subsequent Note events to be interpreted as note-off events.

Time [6,000 values] Indicates the absolute time location within the segment, quantized into 10 ms bins. This time will apply to all subsequent Note events until the next Time event. Time events must occur in chronological order. We define the vocabulary with times up to 60 seconds for flexibility, but because time resets for each segment, in practice we use only the first few hundred events of this type.

EOS [1 value] Indicates the end of the sequence.

Previous work that used such a MIDI-like event vocabulary [32] used relative time shifts between events, indicating the amount of time elapsed since the last time shift. However, in the sequence-to-sequence scenario a single relative time shift error early in the output causes all subsequent output steps to be incorrect, and such errors accumulate as sequence length increases. To adjust for this drift, the Transformer model would have to learn to perform a cumulative sum over all previous time shifts in order to determine the current position in time. We instead use absolute time, where each time event indicates the amount of time from the beginning of the segment, as illustrated in Figure 1. This gives the model the easier task of determining each timestamp independently; we also examine this choice empirically in Section 4.4 and find that using absolute time shifts instead of relative shifts results in much better performance.

We use a temporal resolution of 10 ms for our time events as some experiments have found this displacement to be approximately the limit of human perception [33] (though others have reported smaller values e.g. 5 ms in Handel [34]). We leave open the possibility that our results could be improved further with finer event resolution, for example by predicting continuous times as in Kong et al. [19].

Decoding model output during inference is done with a simple greedy autoregressive algorithm. We choose the maximum probability event at each step and feed that back into the network as the predicted event for that step. We continue this process until the model predicts an EOS token.

Using an event sequence as our training target instead of piano roll matrices or other frame-based formats enables significant flexibility. For example, we demonstrate in Section 4.4 that the exact same model configuration with

¹ https://github.com/google-research/text-to-text-transfer-transformer/blob/master/released_checkpoints.md#t511

² <https://goo.gle/t5x>

³ <http://github.com/google/seqio>

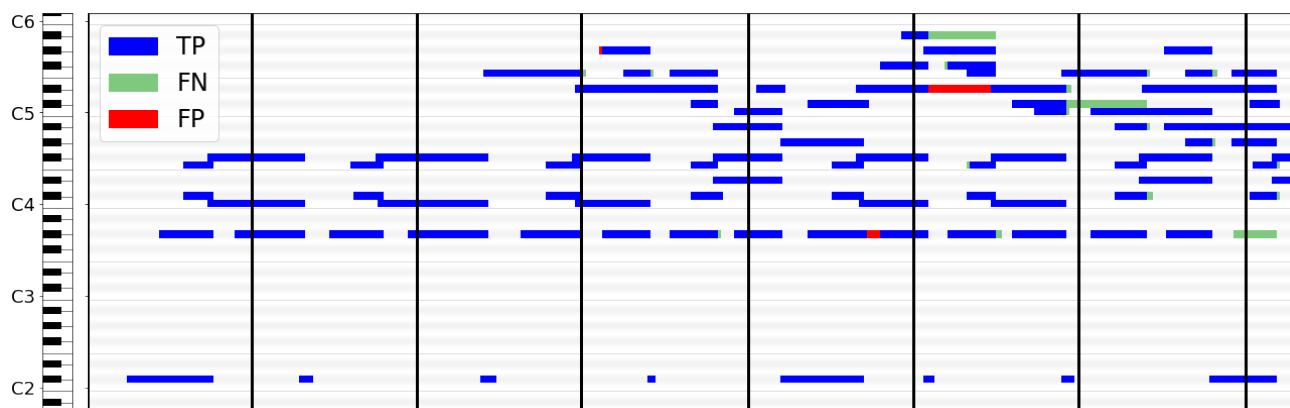


Figure 2. Section of a piano roll rendering of model event output on Chopin’s *Berceuse Op. 57 in D-flat Major* from the MAESTRO validation set versus the ground truth. Black vertical lines represent segment boundaries during inference. True positive (TP) frames are marked in blue, false negatives (FN) in green, and false positives (FP) in red. Note that the model successfully predicts note-off events for notes where the note-on event happened in a different segment.

the same inputs can be trained to predict only onsets (using just the Note, Time, and EOS events) or onsets, offsets, and velocities (using the full vocabulary above). The only change required is using a different set of tokens as the training target. This is in contrast to previous work where predicting a new feature required adding new output heads (or entire stacks), designing losses for those outputs, and modifying the (often non-differentiable) decoding algorithm to combine all model outputs into the final desired representation.

By using a sequence-to-sequence approach, our model can directly output our desired representation by jointly modeling audio features and language-like output dependencies in a fully differentiable, end-to-end training setting. Adding new output features or changing the task definition is simply a matter of changing the tokens used to describe the target output.

3.3 Sequence Length Considerations

Transformers can attend to all tokens in a sequence at every layer, which is particularly suitable to a transcription task that requires fine grained information about pitch and timing for every event. However, this attention mechanism comes at a space complexity of $O(n^2)$ with respect to sequence length n . The practical consequence is that most audio sequences used for transcription cannot fit in memory. To get around this problem, we split the audio sequence and its corresponding symbolic description into smaller segments during training and inference.

During training, we use the following procedure for every sequence in a batch:

1. Select a random audio segment from the full sequence for model input. The length of the selected segment can vary from a single input frame to the maximum input length, and the starting position is selected from a uniform random distribution.
2. Select the symbolic segment for the training target that corresponds to the selected audio segment. Be-

cause notes may start in one segment and end in another, the model is trained to be able to predict note-off events for cases where the note-on event was not observed.

3. Compute a spectrogram for the selected audio and map the symbolic sequence into our vocabulary (see Section 3.2). Absolute time shifts within the symbolic segment are calculated such that time 0 is the beginning of the segment.
4. Provide the continuous spectrogram input and one-hot-encoded MIDI-like events as a training example for the Transformer architecture.

During inference, the following procedure is used:

1. Split the audio sequence into non-overlapping segments using the maximum input length when possible, and then compute spectrograms.
2. For each segment in turn, provide the spectrogram as input to the Transformer model and decode by greedily selecting the most likely token according to the model output at each step until an EOS token is predicted. Any tokens occurring after a time shift beyond the length of the audio segment will be discarded.
3. Concatenate the decoded events from all segments into a single sequence. After concatenation, there may still be note-off events with no corresponding note-on; we remove these. If we encounter a note-on event for a pitch that is already on, we end the note and start a new one. At the end of the sequence, we end any active notes that are missing note-off events.

The model is surprisingly capable at predicting note-on or note-off events where the corresponding event is in a different segment, as illustrated in Figure 2. This ability is also empirically demonstrated by the model’s results on the Onset, Offset, & Velocity F1 scores in Section 4.2.

4. EXPERIMENTS

We trained our model with the Adafactor optimizer [35] using a batch size of 256, a constant learning rate of $1e-3$, and dropout set to .1 for both sub-layer outputs and embedded inputs. Batch size was selected to maximize training throughput because other batch sizes we tried during initial experimentation did not seem to make a difference in final performance. The learning rate and dropout values were set to the same values used by T5 for fine-tuning tasks.

Input spectrograms were calculated using the TensorFlow [36] *tf.signal* library. We used an audio sample rate of 16,000 kHz, an FFT length of 2048 samples, and a hop width of 128 samples. We scaled the output to 512 mel bins (to match the model’s embedding size) and used the log-scaled magnitude.

Input sequences were limited to 512 positions (511 spectrogram frames plus a learnable EOS embedding), and outputs were limited to 1024 positions (1023 symbolic tokens plus a learnable EOS embedding). This corresponds to a maximum segment length of 4.088 seconds. We used 512 input positions to match the sequence length of T5, but future work could explore if other sequence lengths result in better performance. 1024 output positions were used because we found that 512 output positions were not always sufficient to symbolically describe the input audio.

We trained all models on 32 TPUv3 cores, resulting in a per-core batch size of 8. We used this configuration for training speed, but the model is small enough to train on a single TPUv2 instance (8 cores). Based on validation set results, overfitting did not seem to be a problem, so we allowed training to progress for 400K steps, which took about 2.5 days for our baseline models.

4.1 Datasets

To evaluate the performance of our model on the task of piano transcription, we use the MAESTRO dataset [2], which contains about 200 hours of virtuosic piano performances captured with fine alignment between audio and ground truth note annotations. For comparison against previous transcription work, we train on MAESTRO V1.0.0, but for other studies we use MAESTRO V3.0.0 because it contains an additional 92 performances containing 26 hours of data. MAESTRO V3.0.0 also contains *sostenuto* and *una corda* pedal events, though our model (and evaluation) does not make use of these. We also do not model sustain pedal events directly as in Kong et al. [19], instead extending note durations while the sustain pedal is pressed similar to Hawthorne et al. [2].

4.2 Evaluation

In evaluating the performance of a piano transcription system, we use the Note F1 score metric: the harmonic mean of precision and recall in detecting individual notes. This involves matching each predicted note with a unique ground truth note based on onset time, pitch, and optionally offset time. Additionally, onset velocity can be used to discard matches with drastically different velocities. We

primarily use an F1 score that takes into account onsets, offsets, and velocities. We also include results for F1 scores that consider only onsets or onsets and offsets. We defer to the *mir_eval* [37] library for a precise definition of the (standard) transcription metrics we use.

Because piano is a percussive instrument, it is generally easier (and also more perceptually important) to accurately identify note onsets compared to offsets [38]. We use *mir_eval*’s default match tolerance of 50 ms for onsets and the greater of 50 ms or 20% of the note’s duration for offsets.

4.3 Comparison to Previous Work

We compare our sequence-to-sequence approach with the reported scores from previous piano transcription papers on V1.0.0 of the MAESTRO dataset in Table 1. Our method is able to achieve competitive F1 scores compared to the best existing approach while being conceptually quite simple, using a generic architecture and decoding algorithm and standard representations.

4.4 Ablation Study

We perform an ablation study on some of the components of our model using V3.0.0 of the MAESTRO dataset in Table 1. First, we verify the flexibility of this architecture to describe the input audio using a different set of features. We modify the symbolic data to describe only onsets by using just the Note, Time, and EOS events. The model trains successfully and achieves a high F1 score on this modified onset-only task.

Next, we investigate different input representations. For “STFT”, we remove the log mel scaling after FFT calculation. This results in an input frame size of 1025, which is projected to the model embedding size of 512 by the dense layer. For “raw samples”, we simply split the audio samples into segments based on the hop width used for the spectrograms (128 samples) and use those directly as input, again projected to the embedding size by the dense layer. Both of these configurations train successfully, but do not perform as well as log mel input. We suspect this is because the mel scaling produces useful features that the model would otherwise have to use some of its capacity to extract.

We also verify that absolute time shifts are a better fit for this architecture by training a model with relative time shifts. As expected, it does not perform as well. Further, we noticed that the note-based evaluation metrics on the validation set varied dramatically during training, with sometimes as much as a 15 point difference in onset F1 score between adjacent validation steps. We hypothesize this is because small changes in relative time shift prediction are magnified when accumulated across a sequence to determine the absolute times needed for the metrics calculation; i.e., relative time shifts cause the resulting transcriptions to drift out of alignment with the audio.

Finally, we investigate if a larger model size would improve performance. We scaled up our model size based on the “base” configuration from T5. Specifically, we

Model		Onset, Offset, & Velocity F1	Onset & Offset F1	Onset F1
MAESTRO V1.0.0	Transformer (ours)	82.18	83.46	95.95
	Kong et al. 2020 [19]	80.92	82.47	96.72
	Kwon et al. 2020 [21]	–	79.36	94.67
	Kim & Bello 2019 [20]	80.20	81.30	95.60
	Hawthorne et al. 2019 [2]	77.54	80.50	95.32
MAESTRO V3.0.0	Transformer	82.75	83.94	96.01
	Onsets only vocabulary	–	–	96.13
	STFT input	81.81	82.92	95.44
	Raw samples input	74.79	77.26	92.35
	Relative time shifts output	66.25	67.35	80.02
	“Base” model size (100K steps)	81.41	82.78	95.60

Table 1. MAESTRO test set results. Comparisons against previous work are done using V1.0.0 and comparisons against different model configurations are done using V3.0.0 because of its larger size. All Transformer models were trained to 400K steps except for the “Base” configuration which was trained to 100K steps.

modified the following hyperparameters: $d_{\text{model}} = 768$, $d_{\text{ff}} = 2,048$, 12 heads for attention, and 12 layers each in the encoder and decoder. These changes resulted in a model with 213M parameters, as opposed to our “small” configuration which had only 54M. This model quickly overfit the training dataset, with scores on the validation set starting to decline after 100K steps, so we stopped training at that point. Even with early stopping, this model does not perform as well as our “small” configuration, clearly demonstrating that even though piano transcription is a fairly complicated task, it does not require a particularly large Transformer.

5. CONCLUSION AND FUTURE WORK

We have shown that a generic Transformer architecture trained to map spectrograms to MIDI-like output events with no pretraining can achieve state-of-the-art performance on automatic piano transcription. We see this as an appeal to simplicity; we used standard formats and architectures as much as possible and were able to achieve results on par with models customized for piano transcription. Possibly the main source of complexity in our setup is the splitting of examples into segments; future work could include the investigation of sparse attention mechanisms to enable the transcription of an entire piece of music in a single encoding and decoding pass. Also worth exploring would be the use of distillation [39] or related techniques to enable models like this to run in realtime on mobile devices or the web.

Our results suggest that a generic sequence-to-sequence framework with Transformers might also be beneficial for other MIR tasks, such as beat tracking, fundamental frequency estimation, chord estimation, etc. The field of Natural Language Processing has seen that a single large language model, such as GPT-3 or T5, has been capable of solving multiple tasks by leveraging the commonalities between tasks. We are excited by the possibility that similar phenomena could be possible with MIR tasks, and we hope

that these results point toward possibilities for creating new MIR models by focusing on dataset creation and labeling rather than custom model design.

6. REFERENCES

- [1] V. Emiya, R. Badeau, and B. David, “Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2009.
- [2] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=r1IYrjC9F7>
- [3] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and Frames: Dual-objective piano transcription,” in *ISMIR*, 2018.
- [4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NeurIPS*, 2017.
- [5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, “An image is worth 16x16 words: Transformers for image recognition at scale,” in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=YicbFdNTTy>
- [6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text

- Transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.
- [7] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira, “Perceiver: General perception with iterative attention,” *arXiv preprint arXiv:2103.03206*, 2021.
- [8] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with Transformers,” in *ECCV*, 2020.
- [9] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” *arXiv:2102.12092*, 2021.
- [10] K. Lin, L. Wang, and Z. Liu, “End-to-end human pose and mesh reconstruction with Transformers,” in *CVPR*, 2021.
- [11] L. Dong, S. Xu, and B. Xu, “Speech-Transformer: a no-recurrence sequence-to-sequence model for speech recognition,” in *ICASSP*, 2018.
- [12] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, “Very deep self-attention networks for end-to-end speech recognition,” in *Inter-speech*, 2019.
- [13] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, “Neural speech synthesis with Transformer network,” in *AAAI*, 2019.
- [14] Y. Gong, Y.-A. Chung, and J. Glass, “AST: Audio spectrogram Transformer,” *arXiv:2104.01778*, 2021.
- [15] MIDI Manufacturers Association and others, “The complete midi 1.0 detailed specification,” *Los Angeles, CA, The MIDI Manufacturers Association*, 1996.
- [16] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription,” in *ICML*, 2012.
- [17] J. Nam, J. Ngiam, H. Lee, and M. Slaney, “A classification-based polyphonic piano transcription approach using learned feature representations,” in *ISMIR*, 2011.
- [18] S. Böck and M. Schedl, “Polyphonic piano note transcription with recurrent neural networks,” in *ICASSP*, 2012.
- [19] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, “High-resolution piano transcription with pedals by regressing onsets and offsets times,” *arXiv:2010.01815*, 2020.
- [20] J. W. Kim and J. P. Bello, “Adversarial learning for improved onsets and frames music transcription,” in *ISMIR*, 2019.
- [21] T. Kwon, D. Jeong, and J. Nam, “Polyphonic piano transcription using autoregressive multi-state note model,” in *ISMIR*, 2020.
- [22] R. Kelz, S. Böck, and G. Widmer, “Deep polyphonic ADSR piano note transcription,” in *ICASSP*, 2019.
- [23] A. Elowsson, “Polyphonic pitch tracking with deep layered learning,” *Journal of the Acoustical Society of America*, vol. 148, no. 1, pp. 446–468, 2020.
- [24] J. Engel, R. Swavely, L. H. Hantrakul, A. Roberts, and C. Hawthorne, “Self-supervised pitch detection by inverse audio synthesis,” in *ICML Workshop on Self-Supervision in Audio and Speech*, 2020.
- [25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” in *NeurIPS*, 2020.
- [26] M. Awiszus, “Automatic music transcription using sequence to sequence learning,” Master’s thesis, Karlsruhe Institute of Technology, 2019.
- [27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [28] K. Ullrich and E. van der Wel, “Music transcription with convolutional sequence-to-sequence models,” in *ISMIR*, 2017.
- [29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2016.
- [30] J. Heek, A. Levsikaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, “Flax: A neural network library and ecosystem for JAX,” 2020. [Online]. Available: <http://github.com/google/flax>
- [31] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” 2018. [Online]. Available: <http://github.com/google/jax>
- [32] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, “This time with feeling: Learning expressive musical performance,” *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.
- [33] A. Friberg and J. Sundberg, “Perception of just-noticeable time displacement of a tone presented in a metrical sequence at different tempos,” *Journal of The Acoustical Society of America*, vol. 94, no. 3, pp. 1859–1859, 1993.
- [34] S. Handel, *Listening: An introduction to the perception of auditory events*. MIT Press, 1993.
- [35] N. Shazeer and M. Stern, “Adafactor: Adaptive learning rates with sublinear memory cost,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 4596–4604.

- [36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015. [Online]. Available: <https://www.tensorflow.org/>
- [37] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, “mir_eval: A transparent implementation of common MIR metrics,” in *ISMIR*, 2014.
- [38] A. Ycart, L. Liu, E. Benetos, and M. Pearce, “Investigating the perceptual validity of evaluation metrics for automatic piano music transcription,” *Transactions of the International Society for Music Information Retrieval*, 2020.
- [39] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [40] N. Shazeer, “Glu variants improve transformer,” *arXiv preprint arXiv:2002.05202*, 2020.

NEURAL WAVESHAPING SYNTHESIS

Ben Hayes, Charalampos Saitis, George Fazekas

Centre for Digital Music, Queen Mary University of London

{b.j.hayes, c.saitis, g.fazekas}@qmul.ac.uk

ABSTRACT

We present the Neural Waveshaping Unit (NEWT): a novel, lightweight, fully causal approach to neural audio synthesis which operates directly in the waveform domain, with an accompanying optimisation (FastNEWT) for efficient CPU inference. The NEWT uses time-distributed multilayer perceptrons with periodic activations to implicitly learn nonlinear transfer functions that encode the characteristics of a target timbre. Once trained, a NEWT can produce complex timbral evolutions by simple affine transformations of its input and output signals. We paired the NEWT with a differentiable noise synthesiser and reverb and found it capable of generating realistic musical instrument performances with only 260k total model parameters, conditioned on F0 and loudness features. We compared our method to state-of-the-art benchmarks with a multi-stimulus listening test and the Fréchet Audio Distance and found it performed competitively across the tested timbral domains. Our method significantly outperformed the benchmarks in terms of generation speed, and achieved real-time performance on a consumer CPU, both with and without FastNEWT, suggesting it is a viable basis for future creative sound design tools.

1. INTRODUCTION

Synthesisers are indispensable tools in modern music creation. Over the last six decades, their evolving sonic affordances have defined uncountable musical aesthetics and cultures, enabling composers, sound designers, and musicians to interact with human auditory perception in previously impossible ways.

The recent proliferation of deep neural networks as audio synthesisers is further expanding the capabilities of these tools: realistic instrument performances can be synthesised from simple, low dimensional control signals [1–3]; the timbre of one instrument can be convincingly transferred to another [1, 3–5]; instruments can be morphed and interpolated along nonlinear manifolds [6, 7]; and sounds can be manipulated using high level descriptors of perceptual characteristics [7–9]. Yet despite their impressive abilities, these systems have not been widely adopted in music creation workflows.

We argue that this is largely a pragmatic issue. Modern music production centres around the digital audio workstation (DAW), with software instruments and signal processors represented as real-time plugins. These allow users to dynamically manipulate and audition sounds, responsively tweaking parameters as they listen or record. Neural audio synthesisers do not currently integrate elegantly with this environment, as they rely on deep neural networks with millions of parameters, and are often incapable of functioning in real-time on a CPU.

In this work we move towards integrating the benefits of neural audio synthesis into creative workflows with a novel, lightweight architecture built on the principles of digital waveshaping synthesis [10]. Our model implicitly learns a bank of continuous differentiable waveshapers, which are applied to an exciter signal. A control module learns to generate time-varying timbres by dynamically shifting and scaling the learnt waveshaper’s input and output. As the waveshapers encode information about the target timbre, our model can synthesise convincing audio using an order of magnitude fewer parameters than the current state-of-the-art methods.

This paper is laid out as follows. In section 2 we discuss related work on neural audio synthesis and waveshaping. Section 3 introduces our architecture, and we outline our training methodology in section 4. In section 5 we present and discuss evaluations of our model in comparison to the current state of the art methods [1, 3]. Finally, we conclude with suggestions for future work in section 6. We provide full source code¹ and encourage readers to listen to the audio examples in the online supplement².

2. RELATED WORK

2.1 Neural Audio Synthesis

Audio synthesis with deep neural networks has received considerable attention in recent years. Autoregressive models such as WaveNet [11] and SampleRNN [12] defined a class of data-driven, general-purpose vocoder, which was subsequently expanded on with further probabilistic approaches, including flow-based models [13–15] and generative adversarial networks [16–19]. These models allow realistic synthesis of speech, and applications to musical audio [6, 20, 21] have yielded similarly impressive results. A parallel stream of research has focused on controllable musical audio synthesis [1–3, 7, 8, 22], in which



© Ben Hayes, Charalampos Saitis, George Fazekas. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Ben Hayes, Charalampos Saitis, George Fazekas, “Neural Waveshaping Synthesis”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ <https://github.com/ben-hayes/neural-waveshaping-synthesis>

² <https://ben-hayes.github.io/projects/nws/>

models are designed to provide control affordances that may be of practical use. Such controls have included MIDI scores [2, 22], semantic or acoustical descriptors of timbre [7, 8], and F0/loudness signals [1, 3]. The representations of timbre learnt by these models have also been observed to show similarities to human timbre perception [23].

A recent category of model, [1, 3, 24] unified under the conceptual umbrella of differentiable digital signal processing (DDSP) [1], has enabled low-dimensional, interpretable control through strong inductive biases to audio synthesis. Whereas generalised neural vocoders must learn from scratch to produce the features that typify audio signals, such as periodicity and harmonicity, DDSP methods utilise signal processing components designed to produce signals exhibiting such features. These components are expressed as differentiable operations directly in the computation graph, effectively constraining a model’s outputs to a subspace defined by the processor’s capabilities.

DDSP methods fall into two groups: those where the network generates control signals for a processor, and those where the network is trained to be a signal processor itself. The DDSP autoencoder [1] falls into the first category as it generates control signals for a spectral modelling synthesiser [25]. The neural source-filter (NSF) approach [3, 24, 26] is in the second category. It learns a nonlinear filter that transforms a sinusoidal exciter to a target signal, guided by a control embedding generated by a separate encoder. In other words: the control module “plays” the filter network.

The NSF filter network transforms its input through amplitude distortion, as each activation function acts as a nonlinear waveshaper. A given layer’s ability to generate a target spectrum is thus bounded by the distortion characteristics of its activation function. For this reason, neural source-filter models are typically very deep: Wang et al.’s simplified architecture [24] requires 50 dilated convolutional layers, and Michelashvili & Wolf’s musical instrument model [3] consists of 120 dilated convolutional layers – 30 for each of its four serial generators.

Our method avoids the need for such depth by learning continuous representations of detailed waveshaping functions as small multilayer perceptrons. These functions are optimised such that their amplitude distortion characteristics allow them to produce spectral profiles appropriate to the target timbre. This allows our model to accurately transform an exciter signal considerably more efficiently, whilst still exploiting the benefits of the network-as-synthesiser approach.

2.2 Digital Waveshaping Synthesis

In *waveshaping synthesis* [10], timbres are generated using the amplitude distortion properties of a nonlinear shaping function $f: \mathbb{R} \mapsto \mathbb{R}$, which is memoryless and shift invariant. Due to its nonlinearity, f is able to introduce new frequency components to a signal [27]. When a pure sinusoid $\cos \omega n$ is used as the input to f , only pure harmonics are introduced to the signal. An exciter signal with multiple frequency components, conversely, would result in intermodulation distortion, generating components at frequen-

cies $a\omega_1 \pm b\omega_2$, $\forall a, b \in \mathbb{Z}^+$, for input frequencies ω_1 and ω_2 . This would result in inharmonic components if ω_1 and ω_2 are not harmonically related.

The shaping function f is designed to produce a specific spectral profile when excited with $\cos \omega n$. This is achieved as a weighted sum of Chebyshev polynomials of the first kind, which possess the property that the k th polynomial T_k directly transforms a sinusoid to its k th harmonic: $T_k(\cos \omega n) = \cos \omega kn$. With a function specified in this way, we can define a simple discrete time waveshaping synthesiser

$$x[n] = N[n]f(a[n]\cos \omega n), \quad (1)$$

where $a[n]$ is the distortion index and $N[n]$ is a normalising coefficient. As the frequency components generated by a nonlinear function vary with input amplitude, varying the distortion index over time allows us to generate evolving timbres, whilst the normalising coefficient allows us to decouple the frequency content and overall amplitude envelope of the signal.

3. NEURAL WAVESHAPING SYNTHESIS

Our model acts as a harmonic-plus-noise synthesiser [25]. This architecture separately generates periodic and aperiodic components and exploits an inductive bias towards harmonic signals. Fig. 1 illustrates the overall architecture of our model.

3.1 Control Encoder

We condition our model on framewise control signals extracted from the target audio with a hop size of 128. We project these to a 128-dimensional control embedding z using a causal gated recurrent unit (GRU) of hidden size 128 followed by a time distributed dense layer of the same size. We leave the exploration of the performance of alternative sequence models to future work.

3.2 NEWT: Neural Waveshaping Unit

The shaping function f of a waveshaping synthesiser can be fit to only a single instantaneous harmonic spectrum. The spectral evolution afforded by the distortion index $a[n]$ is thus usually unrelated to the target timbre. This is a limitation of the Chebyshev polynomial method of shaping function design. Here, we propose to instead learn a shaping function f_θ parameterised by a multilayer perceptron (MLP). As demonstrated in recent work on implicit neural representations [28, 29], MLPs with sinusoidal activations dramatically outperform ReLU MLPs in learning continuous representations of detailed functions with arbitrary support. We therefore use sinusoidal activations in f_θ , which enables useful shaping functions to be learnt by very compact networks. Here, we use 64 parallel shaper MLPs, each with 4 layers, with a hidden size of 8 neurons.

To enable our model to fully exploit the distortion characteristics of f_θ , we replace the distortion index $a[n]$ and normalising coefficient $N[n]$ with affine transforms before and after the shaping function. The parameters of these transforms, denoted α_a and β_a for the distortion index

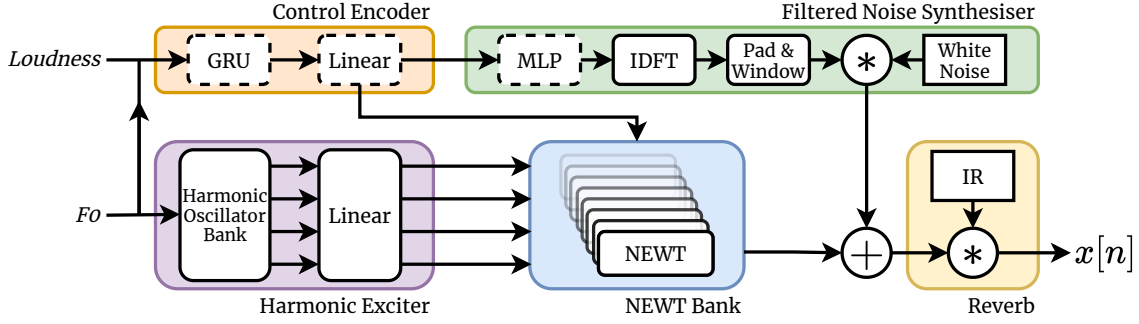


Figure 1. The full architecture of our neural audio synthesiser. All linear layers and MLPs are time distributed. Convolution is denoted $*$ and applied by multiplication in the frequency domain. Blocks with dashed outlines operate at the same coarse time steps as the control signal, whilst those with solid outlines operate at audio rate.

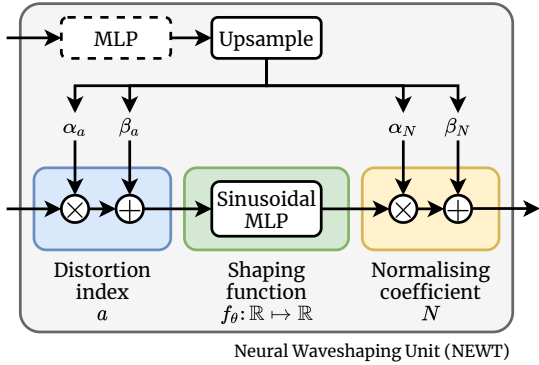


Figure 2. A block diagram depicting the structure of the neural waveshaping unit (NEWT). Blocks with dashed outlines operate at control signal time steps, whilst solid blocks operate at audio rate.

and α_N and β_N for the normalising coefficient, are generated by a separate MLP (depth 4, width 128, ReLU activations with layer normalisation [30]) which takes z as input, and then upsampled to audio rate. The output of a single NEWT in response to exciter signal $y[n]$ is thus given by:

$$x[n] = \alpha_N f_\theta(\alpha_a y[n] + \beta_a) + \beta_N. \quad (2)$$

In this way, the NEWT disentangles two tasks: it learns a synthesiser parameterised by $(\alpha_a, \alpha_N, \beta_a, \beta_N)$, and it learns to “play” that synthesiser in response to a control signal z . Fig. 2 illustrates the structure of the NEWT. In practice, we use multiple such units in parallel. We can implement this efficiently using grouped 1-dimensional convolutions with a kernel size of 1 — essentially a bank of parallel time-distributed dense layers.

3.3 FastNEWT

The NEWT is an efficient approach to generating time-varying timbres, but its reliance on grouped 1-dimensional convolutions best suits it to GPU inference. Many use-cases for our model do not guarantee the availability of a GPU, and so efficient CPU inference is of crucial importance. For this reason, we propose an optimisation called the *FastNEWT*: as each learnable shaping function simply maps $\mathbb{R} \mapsto \mathbb{R}$, it can be replaced by a lookup table of arbitrary resolution. Forward passes through f_θ are then

simply replaced with the $\mathcal{O}(1)$ operation of reading values from an array and calculating an interpolation.

To produce a *FastNEWT*, we sample f_θ across a closed interval. The sampling resolution and interval are tunable parameters of this operation, and represent a trade-off between memory cost and reconstruction quality. Here, we opt for a lookup table of 4096 samples over the interval $[-3, 3]$, using a naïve implementation with linear interpolation. Like the rest of our model, this is implemented using PyTorch operations, and so we treat this as an upper bound on the computational cost of the *FastNEWT*. In practice, an implementation in a language with low level memory access would confer performance improvements.

3.4 Harmonic Exciter

To reduce the resolution required of the shaping functions, we produce our exciter with a harmonic oscillator bank generating up to 101 harmonics, truncated at the Nyquist frequency. The outputs of this oscillator bank are passed through a time distributed linear layer, acting as a mixer which provides each NEWT channel with a weighted mixture of harmonics. Thus, the i th output channel of the exciter module is given by:

$$y_i[n] = \sum_{k=1}^K A(k\omega) w_{ik} \cos k\omega n + b_i, \quad (3)$$

where the antialiasing mask $A(k\omega)$ is 1 if $-\pi < k\omega < \pi$ and 0 otherwise.

3.5 Noise Synthesiser

In spectral modelling synthesis [25], audio signals are decomposed into a harmonic portion and a residual portion. The residual portion is typically modelled by filtered noise, with filter coefficients varying over time according to the spectrum of the residual. Here, we use an MLP (depth 4, hidden size 128, ReLU activations with layer normalisation) to generate 256-tap FIR filter magnitude responses conditioned on z . We apply a differentiable window-design method like that used in the DDSP model [1] to apply the filters to a white noise signal. First, we take the inverse DFT of these magnitude responses, then shift them to causal form, and apply a Hann window to the impulse

response. We then apply the filters to a white noise signal by multiplication in the frequency domain.

3.6 Learnable Reverb

To model room acoustics, we apply a differentiable convolutional reverb to the signal. We use an impulse response $c[n]$ of length 2 seconds, initialised as follows:

$$c[n] \begin{cases} \sim \mathcal{N}(0; 1e-6), & \text{if } n > 1, \\ = 0, & \text{if } n = 0. \end{cases} \quad (4)$$

$c[n]$ is trainable for $n \geq 1$, whilst the 0th value is fixed at 0. The reverberated signal $(c * x)[n]$ is computed by multiplication in the frequency domain, and the output of the reverb is summed with the dry signal.

4. EXPERIMENTS

Our model can be trained directly through maximum likelihood estimation with minibatch gradient descent. Here we detail the training procedure used in our experiments.

4.1 Loss

We trained our model using the multi-resolution STFT loss from [18]. A single scale of the loss is defined as the expectation of the sum of two terms. The first is the spectral convergence L_{sc} (Eqn. 5) and the second is log magnitude distance L_m (Eqn. 6), defined as:

$$L_{sc}(x, \hat{x}) = \frac{\| |STFT_m(x)| - |STFT_m(\hat{x}) \|_F}{\| |STFT_m(x)| \|_F} \quad (5)$$

and

$$L_m(x, \hat{x}) = \frac{1}{m} \|\log |STFT_m(x)| - \log |STFT_m(\hat{x})\|_1 \quad (6)$$

respectively, where $\|\cdot\|_F$ is the Frobenius norm, $\|\cdot\|_1$ is the L1 norm, and $STFT_m$ gives the short-time Fourier transform with analysis window of length m for $m \in \{512, 1024, 2048\}$. We used the implementation of this loss provided in the *auraloss* library [31].

4.2 Data

We collated monophonic audio files from three instruments (violin, trumpet, & flute) from across the University of Rochester Music Performance (URMP) dataset [32], and for each instrument applied the following preprocessing. We normalised amplitude across each instrument subset, made all audio monophonic by retaining the left channel, and resampled to 16kHz. We extracted F0 and confidence signals using the full CREPE model [33] with a hop size of 128 samples. We extracted A-weighted loudness using the procedure laid out in [21] using a window of 1024 samples and a hop size of 128 samples. We divided audio and control signals into 4 second segments, and discarded any segment with a mean pitch confidence < 0.85 . Finally, control signals were standardised to zero mean and unit variance. Each instrument subset was then split into 80% training, 10% validation, and 10% test subsets.

Model	Parameters
HTP	5.6M
DDSP-full	6M
DDSP-tiny	280k*
NWS	266k

* The paper reports 240k [1], but the official implementation contains a model with 280k parameters.

Table 1. Trainable parameter counts of models under comparison.

4.3 Training

We trained our models with the Adam optimiser using an initial learning rate of 1e-3. The learning rate was exponentially decayed every 10k steps by a factor of 0.9. We clipped gradients to a maximum norm of 2.0. All models were trained for 120k iterations with a batch size of 8.

5. EVALUATION & DISCUSSION

To evaluate the performance of our model across different timbres, we trained a neural waveshaping model for each instrument subset. We denote these models *NWS*, specifying the instrument where relevant. After training, we created optimised models with *FastNEWT*, denoted *NWS-FN*, and included these in our experiments also.

5.1 Benchmarks

We evaluated our models in comparison to two state of the art methods: DDSP [1] and Hierarchical Timbre Painting (referred to from here as *HTP*) [3]. We trained these on the same data splits as our model, preprocessed in accordance with each benchmark’s requirements.

Two DDSP architectures were used as benchmarks: the “full” model, originally used to train a violin synthesiser, and the “tiny” model described in the paper’s appendices. Both were trained for 30k iterations as recommended in the supplementary materials. We denote these *DDSP-full* and *DDSP-tiny*, respectively. HTP comprises four distinct Parallel WaveGAN [18] generators operating at increasing timescales. We trained each for 120k iterations, as recommended in the original paper. Table 1 lists the total trainable parameter counts of all models under comparison.

5.2 Fréchet Audio Distance

The Fréchet Audio Distance (FAD) is a metric originally designed for evaluating music enhancement algorithms [34], which correlates well with perceptual ratings of audio quality. It is computed by fitting multivariate Gaussians to embeddings generated by a pretrained VGGish model [35]. This process is performed for both the set under evaluation, yielding $\mathcal{N}_e(\mu_e, \Sigma_e)$, and a set of “background” audio samples which represent desirable audio characteristics, yielding $\mathcal{N}_b(\mu_b, \Sigma_b)$. The FAD is then given by the Fréchet distance between these distributions:

$$F(\mathcal{N}_b, \mathcal{N}_e) = \|\mu_b - \mu_e\|^2 + \text{tr}(\Sigma_b + \Sigma_e - 2\sqrt{\Sigma_b \Sigma_e}). \quad (7)$$

Model	Fréchet Audio Distance		
	Flute	Trumpet	Violin
Test Data	0.463	0.327	0.096
HTP	6.970	14.848	2.529
DDSP-full	3.091	1.391	1.062
DDSP-tiny	3.673	5.301	<i>2.454</i>
NWS	2.704	<i>2.158</i>	5.101
NWS-FN	<i>2.717</i>	2.163	5.091

Table 2. Fréchet Audio Distance scores for all models using background embeddings computed across each instrument’s full dataset. Bold type indicates the best performance in a column and italics the second best.

Thus, a lower FAD score indicates greater similarity to the background samples in terms of the features captured by the VGGish embedding. Here, we used the FAD to evaluate the overall similarity of our model’s output to the target instrument. We computed our background embedding distribution \mathcal{N}_b from each instrument’s full dataset, whilst the evaluation embedding distributions \mathcal{N}_e were computed using audio resynthesised from the corresponding test set. FAD scores for our model, all benchmarks, and the test datasets themselves are presented in Table 2.

In general, the closely matched scores of the NWS and NWS-FN models indicate that, across instruments, the *FastNEWT* optimisation has a minimal effect on this metric of audio quality. On trumpet and flute, our models consistently outperform HTP and DDSP-tiny, and also outperform DDSP-full on flute. On violin, conversely, both DDSP models are the best performers, with HTP achieving a similar score to DDSP-tiny.

5.3 Listening Test

Our model and benchmarks can be considered as highly specified audio codecs. We therefore applied a listening test inspired by the MUSHRA (Multiple Stimuli with Hidden Reference and Anchor) standard [36], which is used to assess the perceptual quality of audio codecs. We used the webMUSHRA framework [37], adapted to incorporate a headphone screening test [38]. For each instrument, we selected two stimuli from the test set representing distinct register and articulation, giving six total trials. In each trial, we used the original recording as the reference and produced the anchor by applying a 1kHz low pass filter. We recruited 19 participants from a pool of audio researchers, musicians, and audio engineers. We excluded the responses of one participant, who rated the anchor above the reference in greater than 15% of trials. Responses for each trial are plotted in Fig. 3. In general, NWS and NWS-FN performed similarly across trials, suggesting that *FastNEWT* has little, if any, impact on the perceptual quality of the synthesised audio. Across flute and trumpet trials our models were rated similarly to the benchmarks. In the first violin trial, our models’ ratings were similar to those of DDSP-tiny, whilst in the second they were lowest overall. These ratings are concordant with FAD scores: our model performs competitively on

Model	Real-time Factor			
	GPU		CPU	
	Mean	90th Pctl.	Mean	90th Pctl.
HTP	0.105	0.106	2.203	2.252
DDSP-full	0.038	0.047	0.363	0.395
DDSP-tiny	0.032	0.039	0.215	0.223
NWS	<i>0.004</i>	<i>0.004</i>	<i>0.194</i>	<i>0.208</i>
NWS-FN	0.003	0.003	0.074	0.076

Table 3. Real-time time factor computed by synthesising four seconds of audio in a single forward pass. Statistics computed over 100 runs. Bold indicates the best performance in a column and italics the second best.

trumpet and flute whilst struggling somewhat with violin.

To examine the influence of melodic stimuli on participants’ ratings, we performed Wilcoxon’s signed-rank test between scores given for each instrument’s two stimuli, for each synthesis model. For example, scores given to DDSP-full for stimulus Flute 1 were compared to scores given to DDSP-full for Flute 2. Out of fifteen tests, significant differences ($p < .001$) were observed in two: between trumpet stimuli for both DDSP-full and HTP. No other significant effects were observed ($\alpha = 0.05$).

To examine the effect of synthesis model, we performed Friedman’s rank sum test on ratings from each trial. For flute stimuli, no significant effects were found. Significant effects were observed for both trumpet stimuli, although Kendall’s W suggested only weak agreement between raters (Trumpet 1: $Q = 27.45, p < 0.001, W = 0.38$; Trumpet 2: $Q = 14.18, p < 0.01, W = 0.20$). Both violin stimuli also resulted in significant effects with moderate agreement between raters (Violin 1: $Q = 42.28, p < 0.001, W = 0.59$; Violin 2: $Q = 37.95, p < 0.001, W = 0.53$). Post-hoc analysis was performed within each trial using Wilcoxon’s signed-rank test with Bonferroni p -value correction. Significant differences (corrected threshold $p < .005$) were observed for Trumpet 1, Violin 1, and Violin 2. These are illustrated as brackets in Fig. 3.

5.4 Real-time Performance

We evaluated the real-time performance of our model in two scenarios. In both cases we took measurements on a GPU (Tesla P100-PCIe 16GB) and a CPU (Intel i5 1038NG7 2.0GHz) and used the real-time factor (RTF) as a metric. The RTF is defined as

$$RTF := \frac{t_p}{t_i}, \tag{8}$$

where t_i is the temporal duration of the input and t_p is the time taken to process that input and return an output. Real-time performance thus requires $RTF < 1$. In all tests we computed RTF statistics over 100 measurements.

The first scenario models applications where an output is expected immediately after streaming an input. To test this, we computed the RTF on four second inputs. We report the mean and 90th percentile in Table 3. On the GPU,

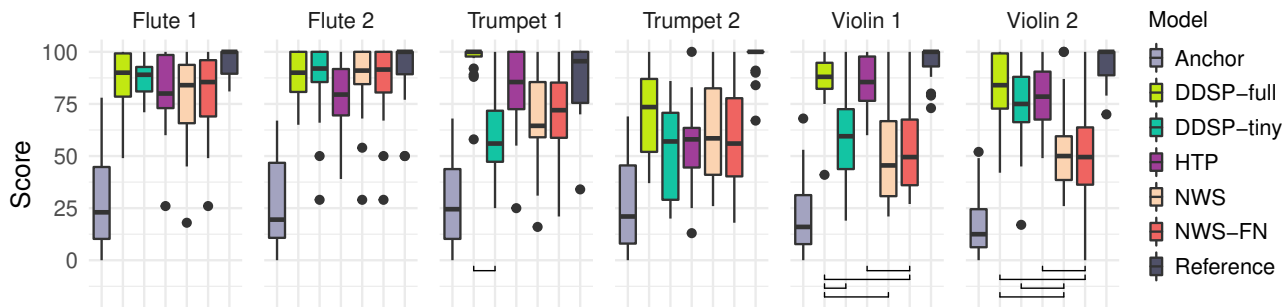


Figure 3. Boxplots of ratings given to each synthesis model during each trial in our listening test. Brackets indicate significant (corrected $p < .005$) differences in pairwise Wilcoxon signed-rank tests with Bonferroni correction.

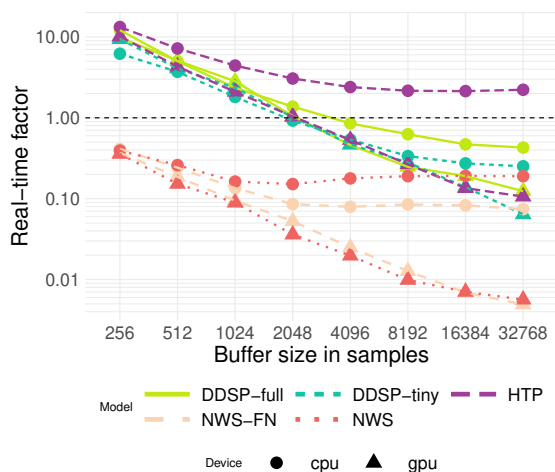


Figure 4. A plot of the mean real-time factor against buffer size across all benchmarks. Mean computed over 100 runs per model per device per buffer size.

NWS and NWS-FN outperformed all benchmarks, including DDSP-tiny. On the CPU, NWS still outperformed all other models, albeit by a narrower margin. The benefit of the *FastNEWT* optimisation was clearer on CPU: NWS-FN had a mean RTF 2.9× lower than the best performing benchmark. On both platforms, HTP was significantly slower, likely due to its much greater depth.

The second scenario assumes applications where immediate response to input is expected, such as in a software instrument. Here, samples are processed in blocks to ensure that sufficient audio is delivered to the DAC in time for playback. We computed the RTF for each buffer size in $B := \{2^n \mid n \in \mathbb{Z}, 8 \leq n < 16\}$. The means of these runs are plotted in Fig. 4. Again, NWS and NWS-FN outperformed all benchmarks on both CPU and GPU, sitting comfortably below the real-time threshold of 1.0 at all tested buffer sizes. HTP did not achieve real-time performance at any buffer size on the CPU, and only did so for buffer sizes over 2048 on the GPU. DDSP-full, similarly, was unable to achieve realtime performance for buffer sizes of 2048 or lower on GPU or CPU, while DDSP-tiny sat on the threshold at this buffer size. It should be noted that a third-party, stripped down implementation of the DDSP model was recently released, which is capable of real-time inference when the convolutional reverb

module is removed³.

6. CONCLUSION

In this paper, we presented the NEWT: a neural network structure for audio synthesis based on the principles of waveshaping [10]. We also present full source code, pre-trained checkpoints, and an online supplement containing audio examples. Our architecture is lightweight, causal, and comfortably achieves real-time performance on both GPU and CPU, with efficiency further improved by the *FastNEWT* optimisation. It produces convincing audio directly in the waveform domain without the need for hierarchical or adversarial training. Our model is also capable of many-to-one timbre transfer by extracting F0 and loudness control signals from the source audio. Examples of this technique are provided in the online supplement.

In evaluation with a multi-stimulus listening test and the Fréchet audio distance our model performed competitively with state-of-the-art methods with over 20× more parameters on trumpet and flute timbres, whilst performing similarly to a comparably sized DDSP benchmark on violin timbres. Due to the use of a harmonic exciter in our architecture and the scope of our experimentation, further work is necessary to ascertain to what degree the NEWT itself contributes to our model’s performance. Therefore, in future work we will perform a full ablation study and a quantitative analysis of the degree to which a trained model makes use of the NEWT’s waveshaping capabilities. In the meantime, the online supplement demonstrates through visualisations of learnt shaping functions, affine parameters $(\alpha_a, \beta_a, \alpha_N, \beta_N)$, and audio taken directly from the output of the NEWT, that the NEWTs in our model do indeed perform waveshaping on the exciter signal.

We suspect the lower scores on violin timbres were due to the greater proportion of signal energy in higher harmonics in these sounds. The NEWT may thus been unable to learn shapers capable of producing these harmonics without introducing aliasing artefacts. Using sinusoidal MLPs with greater capacity inside the NEWT may allow more detailed shaping functions to be learnt, whilst retaining efficient inference with *FastNEWT*. Future work will investigate this and other differentiable antialiasing strategies, including adaptive oversampling [39]. We will also explore extending our model to multi-timbre synthesis.

³ https://github.com/acids-ircam/ddsp_pytorch

7. ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers at *ISMIR* for their thoughtful comments. We would also like to thank our colleague Cyrus Vahidi for many engaging and insightful discussions on neural audio synthesis. This work was supported by UK Research and Innovation [grant number EP/S022694/1].

8. REFERENCES

- [1] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, “DDSP: Differentiable Digital Signal Processing,” in *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.
- [2] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, “Neural Music Synthesis for Flexible Timbre Control,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, United Kingdom, 2019, pp. 176–180.
- [3] M. M. Michelashvili and L. Wolf, “Hierarchical Timbre-painting and Articulation Generation,” in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Oct. 2020.
- [4] S. Huang, Q. Li, C. Anil, S. Oore, and R. B. Grosse, “TimbreTron A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer,” in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.
- [5] D. K. Jain, A. Kumar, L. Cai, S. Singhal, and V. Kumar, “ATT: Attention-based Timbre Transfer,” in *2020 International Joint Conference on Neural Networks (IJCNN)*. Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–6.
- [6] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, Sydney, Australia, Aug. 2017, pp. 1068–1077.
- [7] P. Esling, A. Chemla, and A. Bitton, “Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 175–181.
- [8] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, “Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows,” *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.
- [9] J. Nistal, S. Lattner, and G. Richard, “DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks,” in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Montréal, Aug. 2020.
- [10] M. Le Brun, “Digital Waveshaping Synthesis,” *Journal of the Audio Engineering Society*, vol. 27, no. 4, pp. 250–266, Apr. 1979.
- [11] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A Generative Model for Raw Audio,” *arXiv:1609.03499 [cs]*, Sep. 2016, arXiv: 1609.03499.
- [12] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, “SampleRNN: An Unconditional End-to-End Neural Audio Generation Model,” in *5th International Conference on Learning Representations*, Toulon, France, 2017.
- [13] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A Flow-based Generative Network for Speech Synthesis,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, May 2019, pp. 3617–3621.
- [14] W. Song, G. Xu, Z. Zhang, C. Zhang, X. He, and B. Zhou, “Efficient WaveGlow: An Improved WaveGlow Vocoder with Enhanced Speed,” in *Interspeech 2020*. ISCA, Oct. 2020, pp. 225–229.
- [15] A. v. d. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. v. d. Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, N. Casagrande, D. Grewe, S. Noury, S. Dieleman, E. Elsen, N. Kalchbrenner, H. Zen, A. Graves, H. King, T. Walters, D. Belov, and D. Hassabis, “Parallel WaveNet: Fast High-Fidelity Speech Synthesis,” *arXiv:1711.10433 [cs]*, Nov. 2017, arXiv: 1711.10433.
- [16] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, “MelGAN: Generative adversarial networks for conditional waveform synthesis,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019.
- [17] J. Kong, J. Kim, and J. Bae, “HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 17 022–17 033.
- [18] R. Yamamoto, E. Song, and J.-M. Kim, “Parallel Wavegan: A Fast Waveform Generation Model Based on Generative Adversarial Networks with Multi-Resolution Spectrogram,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech*

- and *Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 6199–6203.
- [19] C. Donahue, J. McAuley, and M. Puckette, “Adversarial Audio Synthesis,” in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 16.
- [20] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, “GANSynth: Adversarial Neural Audio Synthesis,” in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.
- [21] L. Hantrakul, J. Engel, A. Roberts, and C. Gu, “Fast and Flexible Neural Audio Synthesis,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 524–530.
- [22] N. Jonason, B. L. T. Sturm, and C. Thome, “The control-synthesis approach for making expressive and controllable neural music synthesizers,” in *Proceedings of the 2020 AI Music Creativity Conference*, 2020, p. 9.
- [23] B. Hayes, L. Brosnahan, C. Saitis, and G. Fazekas, “Perceptual Similarities in Neural Timbre Embeddings,” in *DMRN+15: Digital Music Research Network One-day Workshop 2020*, London, UK, 2020.
- [24] X. Wang, S. Takaki, and J. Yamagishi, “Neural Source-Filter Waveform Models for Statistical Parametric Speech Synthesis,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 402–415, 2020.
- [25] X. Serra and J. Smith, “Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition,” *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.
- [26] Y. Zhao, X. Wang, L. Juvela, and J. Yamagishi, “Transferring Neural Speech Waveform Synthesizers to Musical Instrument Sounds Generation,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, May 2020, pp. 6269–6273.
- [27] J. D. Reiss and A. P. McPherson, “Overdrive, Distortion, and Fuzz,” in *Audio effects: theory, implementation and application*. Boca Raton London New York: CRC Press, Taylor & Francis Group, 2015, pp. 167–188, oCLC: 931666647.
- [28] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, “Implicit Neural Representations with Periodic Activation Functions,” *arXiv:2006.09661 [cs, eess]*, Jun. 2020, arXiv: 2006.09661.
- [29] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn, “CKConv: Continuous Kernel Convolution For Sequential Data,” *arXiv:2102.02611 [cs]*, Feb. 2021, arXiv: 2102.02611.
- [30] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” *arXiv:1607.06450 [cs, stat]*, Jul. 2016, arXiv: 1607.06450.
- [31] C. J. Steinmetz and J. D. Reiss, “auraloss: Audio focused loss functions in PyTorch,” in *Digital music research network one-day workshop (DMRN+15)*, 2020.
- [32] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, “Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications,” *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, Feb. 2019.
- [33] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, “Crepe: A Convolutional Representation for Pitch Estimation,” in *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 161–165.
- [34] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms,” in *Inter-speech 2019*. ISCA, Sep. 2019, pp. 2350–2354.
- [35] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 131–135.
- [36] I.-R. BS.1534-3, “Method for the subjective assessment of intermediate quality level of audio systems,” ITU-R, Tech. Rep., 2015.
- [37] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, “webMUSHRA — A Comprehensive Framework for Web-based Listening Tests,” *Journal of Open Research Software*, vol. 6, p. 8, Feb. 2018.
- [38] A. E. Milne, R. Bianco, K. C. Poole, S. Zhao, A. J. Oxenham, A. J. Billig, and M. Chait, “An online headphone screening test based on dichotic pitch,” *Behavior Research Methods*, Dec. 2020.
- [39] B. De Man and J. D. Reiss, “Adaptive control of amplitude distortion effects,” in *Audio engineering society conference: 53rd international conference: Semantic audio*, Jan. 2014.

A SEMI-AUTOMATED WORKFLOW PARADIGM FOR THE DISTRIBUTED CREATION AND CURATION OF EXPERT ANNOTATIONS

Johannes Hentschel¹, Fabian C. Moss¹, Markus Neuwirth², Martin Rohrmeier¹

¹ École Polytechnique Fédérale de Lausanne, Switzerland

² Anton Bruckner University Linz, Austria

johannes.hentschel@epfl.ch

ABSTRACT

The creation and curation of labeled datasets can be an arduous, expensive, and time-consuming task. We introduce a workflow paradigm for remote consensus-building between expert annotators, while considerably reducing the associated administrative overhead through automation. Most music annotation tasks rely heavily on human interpretation and therefore defy the concept of an objective and indisputable ground truth. Thus, our paradigm invites and documents inter-annotator controversy based on a transparent set of analytical criteria, and aims at putting forth the consensual solutions emerging from such deliberations. The workflow that we suggest traces the entire genesis of annotation data, including the relevant discussions between annotators, reviewers, and curators. It adopts a well-proven pattern from collaborative software development, namely distributed version control, and allows for the automation of repetitive maintenance tasks, such as validity checks, message dispatch, or updates of meta- and paradata. To demonstrate the workflow’s effectiveness, we introduce one possible implementation through GitHub Actions and showcase its success in creating cadence, phrase, and harmony annotations for a corpus of 36 trio sonatas by Arcangelo Corelli. Both code and annotated scores are freely available, and the implementation can be readily used in and adapted for other MIR projects.

1. INTRODUCTION

Labeled datasets are an essential prerequisite for many tasks in Music Information Retrieval (MIR) and Digital Musicology, and those tasks are directly dependent on the quality of the labels. Thus, great care needs to be taken during data creation and curation processes in order to provide reliable data for algorithmic evaluation and other purposes. However, the procedures underlying data creation processes as well as how data quality is assessed and assured are not always made explicit and well-documented,

a fact which consequently impedes *post hoc* quality control and reproducibility.

The literature on annotation workflows is sparse and widely dispersed, and the description of data, their meta-data, and creation processes is commonly tailored to specific datasets and research questions. Generic procedures that emphasize common aspects and steps in labeling pipelines are rare. Fortunately, recent years have seen an increasing number of publications directly addressing these issues for MIR contexts, and researchers are more and more actively describing and documenting the procedures that lead to the creation of datasets, along with discussions of the challenges faced and proposed solutions [1–6]. In particular, publications presenting dedicated datasets [7–13] or workflows [14, 15] discuss these aspects in more detail and present the solutions adopted for the specific research purposes. Standardized solutions for the wider community, however, do not yet exist, both due to the diverse and specific requirements for different data and to the relatively high workload and generally low recognition associated with documentation. The ability to rely on established workflows thus would liberate researchers from this arduous resource- and time-consuming responsibilities.

We propose a viable solution that addresses these issues by introducing a novel workflow paradigm for the distributed production of expert or crowd-sourced annotations that is easy to adopt and adapt. It has the overarching aim to streamline the annotation procedure by reducing the associated administrative overhead. Our workflow is designed to optimize the trade-off between working hours spent and data quality achieved, with the goal to maximize trustworthiness and usability of the resulting annotation labels. We demonstrate its effectiveness through an example implementation¹ by means of GitHub Actions which we use to create cadence, phrase, and harmony annotations for a corpus of scores of 36 trio sonatas by Arcangelo Corelli².

Our proposal may serve as a starting point for a wider discussion in the MIR community about how to optimize annotation tasks on a larger scale according to a set of agreed-upon criteria.



© J. Hentschel, F. C. Moss, M. Neuwirth, and M. Rohrmeier. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Hentschel, F. C. Moss, M. Neuwirth, and M. Rohrmeier, “A semi-automated workflow paradigm for the distributed creation and curation of expert annotations”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ github.com/DCMLab/dcm1_annotation_workflow

² github.com/DCMLab/corelli

2. SETTING

This paper addresses distributed settings in which music experts generate symbolic research data by annotating given music representations such as scores, recordings, or timelines. In this section we describe (1) the different roles of individuals involved, (2) some requirements for the annotations that are to be produced, and (3) common steps in an annotation workflow. Finally, we (4) highlight some concrete problems pertinent to these aspects, motivating our novel workflow paradigm.

2.1 Roles

Individuals involved in the annotation of datasets for MIR tasks can assume one or several of the following roles:

Annotators: experts who provide labels after having familiarized themselves with a set of annotation guidelines.

Reviewers: experts who proof-read the provided annotations and provide feedback, criticism, or corrections.

Curators: individuals who are responsible for the initiation, planning, coordination, and/or financing of the project and whose tasks might also involve the creation and maintenance of annotation guidelines and standards (e.g. through the use of ontologies, vocabularies, or syntax specifications).

Annotators, reviewers, and curators all benefit from the workflow we propose, since it aims to streamline their interaction and to reduce their workload.

2.2 Labels

We use the words annotations and labels interchangeably and in their widest sense, meaning that they could, for instance, follow a pre-defined syntax, pertain to a closed vocabulary, or represent score reductions, graphs, trees etc. Our workflow paradigm is applicable in projects for which the following general assumptions hold: (a) annotation labels are encoded and stored as plain text in order to allow for transparent version control; (b) each label thus provided uniquely refers to a specific segment in the music, such as a range of bars, a set of events, or a time-span; (c) each label corresponds to a precisely formalized encoding scheme, structured vocabulary, or other annotation standard that can be algorithmically validated; (d) annotators and reviewers share the goal to bring forth a final version of labels that reconciles their particular musical expertise with the analytical guidelines underlying the project; (e) the creation and curation of annotated datasets is an inherently open-ended process and one must be able to potentially subject the data to future changes, in particular when the curators introduce changes in the annotation guidelines or syntactic specifications; (f) access to the full history of each label makes a dataset's provenance transparent and increases its trustworthiness.

2.3 Annotation process

Usually, it is the curators' responsibility to clearly define the annotation task(s), to assemble and organize the music representations to be annotated ('original data'), to set

up the project infrastructure, and to engage a pool of experts in the endeavor. Depending on the setting, annotators and reviewers need to be familiarized with the tasks, processes, tools, and specific guidelines, often supported by tutorials, training videos, trial phases, or individual coaching. The original data needs to be made available and assigned to (or self-assigned by) annotators, and annotation data to reviewers. The latter assess the quality of the provided labels, e.g. on a case-by-case basis or by applying specific sampling criteria. Finally, curators may check further samples to ensure highest possible quality, and admit the proof-read and validated labels to the final stage, which might coincide with publication.

2.4 Problem statement

The problem we address with this publication is the optimization of the portrayed annotation process in terms of human resources and data quality. We consider this process complete as soon as all envisaged annotations have been created, validated, and verified at least once (for details, see Section 3). At any given moment, the latest validated version of the dataset should be retrievable and the full history of the data genesis, including the provenance of every label, must be stored for maximum transparency.

Most music annotation tasks rely heavily on human judgement and are thus to a large extent subject to interpretation [5, 16–21]. MIR as well as other domains in need of great amounts of subjective annotation data have long since turned to crowd-sourcing as a means of leveraging a massive inexpensive labor force, a paradigm that entails a whole range of well-known problems with respect to quality control, amongst others [22–25]. By valuing quality over quantity, however, our approach favors soliciting fewer and appropriately remunerated experts. This requires diligence and careful organization to ensure outcomes that are effective regarding the tasks to be accomplished as well as economically feasible.

Annotation tasks moreover require appropriate technological infrastructure that allows annotators and reviewers to debate their interpretations within the scope set forth by the annotation guidelines, and to subsequently incorporate the consensual labels in the dataset. This process of consensus-building between experts, proposed in [17], needs to be recorded for future reference.

Finally, curating such an endeavor 'manually', i.e., by exchanging commissions, files, and arguments (e.g. via e-mail), creates a strong desire for a workflow paradigm that easily automates repetitive maintenance tasks. These laborious tasks include the dispatch of notification messages, data validation, and updates of metadata, paradata, and data facets. Not only are these tasks time-consuming for annotators, reviewers, and curators alike, they also bear the danger of being oblivious of issues that require crucial attention. Thus, a system must be put in place that prevents important production steps from being forgotten.

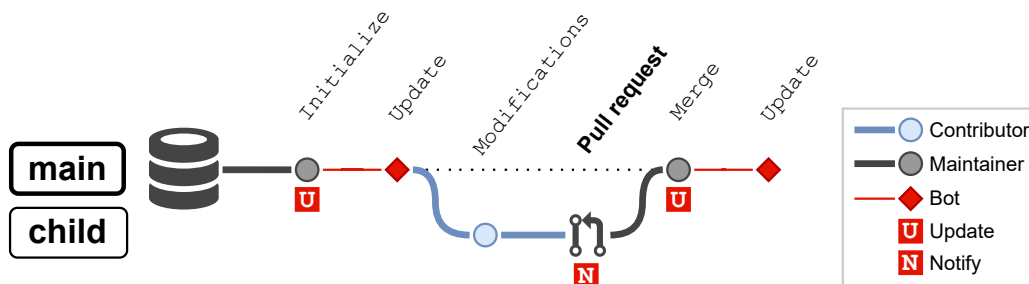


Figure 1. Basic branching scheme. The main branch of the repository exposes the latest version of the dataset that has been approved by a maintainer, whereas contributors can commit only to child branches. Circles represent commits by humans, red diamonds commits by bots, and red squares with a white letter represent triggered scripts.

3. A NOVEL ANNOTATION WORKFLOW PARADIGM

The problems and characterizations of the annotation process stated in the preceding section exhibit a large overlap with the domain of collaborative software engineering [26]. Both describe distributed settings in which a potentially great number of contributors collaborate on creating a possibly large collection of texts: a codebase in the case of software, a dataset of annotations in the present case. Both scenarios can be regarded as open-ended tasks since there is commonly no clearly defined final state; continuous development, possibly based on user feedback, is the norm rather than the exception [27]. Quality control plays a crucial role for the creation of both software and annotations, and is often aided by guidelines, mutual reviews, and automated tests. Moreover, keeping a version history is required for compatibility in software and for reproducibility in data evaluation. Consequently, adopting best practices from distributed version control presents itself as a naturally viable solution. Throughout this section we use terminology that has partly been coined and popularized through the version control system Git [28,29], but our proposed workflow paradigm can equally well be implemented with similar systems such as Mercurial or Subversion.

3.1 Distributed version control

Our workflow paradigm builds on a well-proven pattern from distributed version control, namely parallel branching [26, 30]. The patterns that we describe in this section are generic in the sense that the involved concepts can be understood as abstract classes which can manifest and be implemented in many different ways.

The basic principle is shown in Figure 1 that displays the version history of a data ‘repository’ along a timeline. Each new version is created by a ‘commit’, that is, a set of changes applied to the previous version by an ‘author’ and summarized in a ‘commit message’. In the sketch, commits by human authors are shown as circles and those made by bots as diamonds. Each of the two horizontal levels represents a ‘branch’, of which, in principle, there can be infinitely many in parallel. Every child branch branches off directly or indirectly of the repository’s main branch,

or ‘trunk’, and the commits it contains can be merged back into it anytime. We refer to individuals with the permission to merge, or ‘pull’, changes into the main branch as ‘maintainers’.

All other human authors, or ‘contributors’, do not have permission to merge into main and therefore need to issue a ‘pull request’ that may or may not be accepted by a maintainer based on their review of the commits on the child branch. Maintainers therefore keep full control over the trunk which exposes at all times the latest version of the repository in which all commits have been made, or approved by them. This is a design choice for a scenario where one entity (person or institution) guarantees the integrity of the repository’s main branch (e.g., with respect to a stipulated set of guidelines) while allowing for external contributions in a controlled manner.

3.2 Data maintenance

The red elements in Figure 1 express automation. Specifically, squares represent scripts that are triggered by certain events, and which may result in a bot pushing a commit. For instance, the script `Notify` simply dispatches automated messages to the relevant persons to inform them about a newly issued pull request, so that it does not go unnoticed. The `Update` script is triggered upon push to the main branch and uses a bot to commit its outputs. In the case of an annotation workflow such updates most typically comprise (a) **metadata**, e.g., amount, type, and format of annotation labels, information on the annotated pieces, free text descriptions; (b) **paradata**, e.g., annotator identities, name and version of the employed annotation software, review status, time stamp; or (c) **data facets**, e.g., extraction/separation of particular features for increased accessibility, or summary statistics for individual pieces or the entire dataset. For a concrete example, see Section 4.4. Including an `Update` routine immensely facilitates the project coordination—e.g. through an always up-to-date dashboard or README file—and allows one to use the dataset in its current state at any point in time.

3.3 Data validation

Figure 2 exemplifies the workflow paradigm for the completion of one (partial or comprehensive) set of annota-

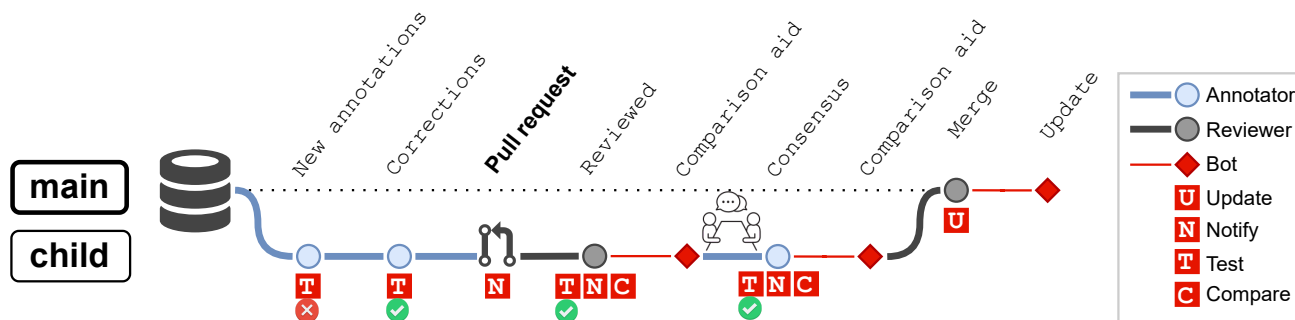


Figure 2. Example for a complete annotation cycle. In a parallel branch, an annotator pushes new annotations that don’t pass the automatic validation (**Test**), corrects the errors, and issues a pull request. As soon as a reviewed version is pushed, **Compare** adds files to aid the annotator with the process of going through the changes and deliberating with the reviewer.

tions, including data verification (see the follow subsection) and validation. Since we established in Section 2.2 that the annotation labels may be automatically validated, we included the script **Test** which is triggered upon every commit to any child branch (except if performed by a bot). This way we ensure that syntactic and orthographic errors in the annotations can be corrected right away and we disallow merging into the main branch in cases where the validity test did not pass. This way we ensure that invalid data cannot make its way into the main branch.

3.4 Data verification through expert consensus

As a consequence of the subjective nature of annotations, data verification can not completely be delegated to algorithmic evaluation. Conceptually, we substitute for the concept of an objective and indisputable ground truth the idea of consensual solutions based on transparent deliberation. The data verification procedure begins with the annotator issuing a pull request (see Figure 2) for merging a set of new annotations into the main branch, thus setting off the **Notify** script (see Section 3.2). From here on, every additional commit is included in the open pull request and triggers (except if performed by a bot) an additional third script, here called **Compare**. In the example, the reviewer updates the proposed annotations, correcting obvious errors and substituting diverging musical interpretations. Pushing these changes causes (a) **Test** to validate the reviewer’s changes, (b) **Notify** to inform the annotator about the completed review, and (c) **Compare** to create one or several files that may aid the annotator to retrace the reviewer’s changes easily, for instance through a diff file, a revision report, or a compilation of reviewer comments. In case the annotator agrees with all changes, the new set of annotations is considered to represent a consensus between the two experts. Otherwise, this consensus is to be reached through transparent deliberation, i.e., a recorded exchange of arguments (symbolized in Fig. 2 by the discussion table), at the end of which the annotator pushes the consensual solution, whereupon the reviewer or anyone with the relevant permissions may merge the thus verified data, completing this subset. Consequently, every verified label represents a consensus between at least two experts.

4. GITHUB ACTIONS IN ACTION: THE GENESIS OF A NOVEL DATASET

To demonstrate our paradigm’s effectiveness we implemented one possible instance¹ of the proposed workflow paradigm and showcase its success in creating cadence, phrase, and harmony annotations for a corpus of 36 trio sonatas (opp. 1, 3, and 4) by Arcangelo Corelli.² The implementation was created using the code hosting service `github.com` because (a) it is frequently used in music research projects for storing research data; (b) it is well-known and chances are that users and new annotators are already familiar with it; (c) it offers a free plan that includes server run time for automating tasks; and because (d) of its easy-to-use automation capacities which include predefined code patterns (called **Actions**) as well as custom scripts. Since the automation is defined in configuration files contained in the repository, reusing our proposed implementation can be easily achieved by simply starting from the corresponding template repository. It makes use of the Python library `ms3`³ to perform tasks on the annotated scores, such as extracting and processing the contained labels, and storing them as tabular TSV files.

4.1 Score annotation: harmony, phrases, and cadences

To create the annotated dataset we commissioned trained music theorists to enter the corresponding labels directly into the provided digital scores, using the latest version of the DCML harmonic annotation standard.⁴ It has a predefined syntax that can be automatically validated, and a set of annotation guidelines⁵ on the basis of which our contractors were able to put forth consensual solutions. The annotation task was performed using the free and open-source notation software **MuseScore** because it is well-known to many musicians and theorists and offers one of the most convenient interfaces for digitally annotating scores. Thus, the annotation labels are stored within the original data and can be viewed by opening the **MuseScore** files or the corresponding annotation tables in TSV format.

³ pypi.org/project/ms3/

⁴ github.com/DCMLab/standards/

⁵ dclmlab.github.io/standards/tutorial

4.2 Validation: Automated syntax checks

Putting into practice what was introduced in Section 3, annotators create new annotations by committing changes made in the MuseScore files to side branches of the central repository on GitHub, called ‘origin’. Every time a set of such commits is pushed to the origin, a virtual machine is initialized on GitHub’s server infrastructure in order to run the above-mentioned `Test` script. It is defined in the configuration file `check.yml` and uses `ms3` to parse the newly annotated files and validate the syntactic correctness of the contained labels. If the validation fails the annotators will be informed instantly and see a list of all syntactically wrong labels together with their exact positions in the score.

4.3 Verification: Automated comparison files

As soon as the new labels have been validated, the annotator issues a pull request, causing GitHub to dispatch notifications to reviewers and curators, and the data verification by a reviewer ensues, as described in Section 3.4. The main problem that our workflow solves at this stage is tracking changes to the proposed set of new annotations individually and in conjunction with the respective justifications. Reviewers are requested to combine modifications into individual commits such that the commit messages include a measure number and the reasoning behind the changes. That way, the original annotators are able to go through these commit messages one by one in order to decide whether they consent to the change or not. In the latter case, they object by engaging in a written exchange of arguments with the respective reviewer. The pull request functionality on code hosting sites such as GitHub ideally supports the subsequent consensus-building process by providing and storing practical and interactive summaries of the comments and commits added in the process as well as by notifying the involved parties about such events.

Although GitHub lets users conveniently visualize the changes made with every commit, a difficulty arises from the fact that, in most cases, it is not sufficient to inspect the source code excerpts from the modified MuseScore files to appraise a changed label. Therefore, our automation script `Compare` adds or updates an additional MuseScore file upon every commit into an open pull request, in which the modifications pertaining to the current verification phase are highlighted with different colors. These files immensely facilitate the discussions about the proposed solutions and may also serve at a later point for documentation purposes or evaluations (see Section 4.5).

As soon as consensus is reached and the new annotations have thus been verified, the corresponding branch is merged into `main` and the pull request is archived, storing and documenting the verification process for the future.

4.4 Maintaining metadata, paradata, and data facets

In our implementation, the `Update` script (see Section 3) is called `extract` because it automatically commits in-

formation extracted from the source code of the modified MuseScore files upon every push to the origin’s main branch. Further, meta- and paradata are obtained for and copied from the changed MuseScore files and then included in a tabular metadata file summarizing the dataset as well as in the repository’s README file. Moreover, annotation labels, notes, and other score elements are extracted and stored as individual tabular files. This mechanism ensures that those facets of the dataset that users will generally be most interested in stay updated and may be loaded, transformed, and evaluated with greater ease, thus maximizing the dataset’s accessibility and reusability.

4.5 Workflow evaluation

# changes	count	%	# syntax errors	count
0	5333	62.6	0	5333
1	2511	29.5	0	2466
			1	45
2	574	6.7	0	534
			1	40
3	89	1.0	0	72
			1	15
			2	2
4	14	0.2	0	12
			1	1
			3	1
5	3	0.0	0	1
			1	1
			3	1

Table 1. For 8542 verified labels in the Corelli dataset, we report the label count for the number of changes that they underwent, and how many of these changes addressed syntactic errors rather than inter-expert disagreement. For example, 3 labels were changed 5 times and throughout 6 versions, they saw 0, 1, and 3 syntactic errors respectively.

Our workflow implementation allows for multiple ways of evaluating the annotation procedure, of which we will showcase two examples. First, for every annotated MuseScore file in the Corelli dataset, we extracted all versions from the Git history and tallied the labels and their positions from each. Tracking all occurring positions over the file’s entire history allowed us to count the number of changes that the label at each position was subjected to and whether they were required due to syntactic errors or otherwise. The results in Table 1 show that roughly 63 % of labels were considered correct from the start and did not change over the course of a file’s history, whereas only 1.2 % needed to be modified more than twice until a consensus was reached. Note that this approach does not reveal whether changes were effected by annotators or reviewers. However, since after a full workflow cycle all labels eventually represent consensual solutions between at least two experts, we can deduce that our workflow is highly efficient in putting forth validated and trustworthy annotation data. The overall rate of labels that has been syntactically wrong at one point is extremely low (1.2 %),

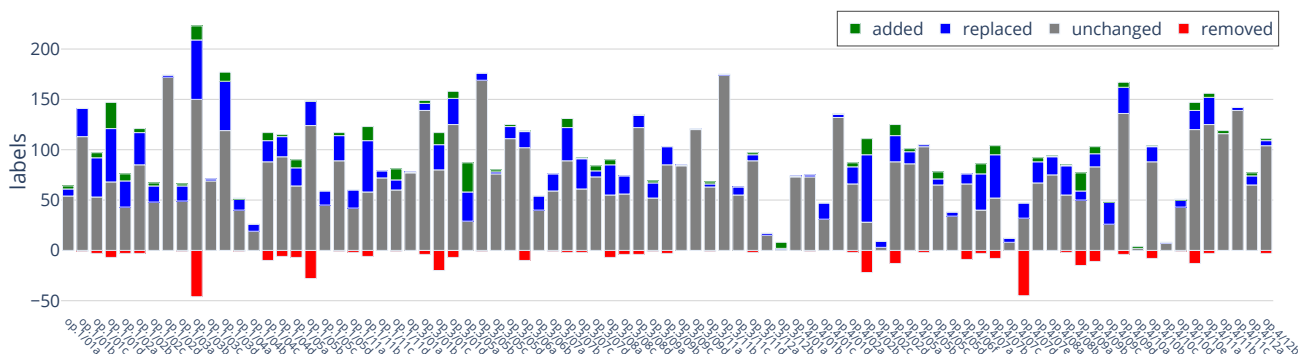


Figure 3. This plot shows for 85 movements how many of the labels have been added, replaced, or left unchanged during the most recent consensus-building phase, and how many have been removed that were previously included. Files showing very few modifications might originate from later minor revisions of verified labels.

suggesting that the guidelines the annotators received were comprehensible and easy to implement. Nonetheless, detecting these kinds of errors automatically constitutes an invaluable advantage, since syntactic validity is the minimal requirement for usable annotations.

For our second evaluation we exploited the annotated MuseScore files that were automatically generated during every verification procedure. They show the difference between exactly two versions, namely between the one that the reviewer started off with and the verified version that represents a consensus between reviewer and annotator. These files maintain the labels from the previous version and show the differences through a color coding, namely red for deleted labels and green for substituted or added labels. This color coding is independent of who committed the modifications, since the most recent version always represents the consensus.

Figure 3 shows an evaluation of the changes made to the 8524 labels of 85 movements from the trio sonatas. We interpreted co-occurrences of a green and a red label at the same position as replacements, which we show in blue and subtract from the green and red bars. Labels that were removed during the verification are shown in the negative range so that the positive range reflects the *status quo*.

A closer analysis of the changed content of the labels by substitution or replacement shows that these often entail music-theoretically fine distinctions, such as whether a chord should be interpreted as $\vee 7 / IV$ or $I 7$. Both have the identical absolute surface realizations (e.g. a C7 triad in the key of C major) but their relative, hierarchical interpretations differ. Another example would be $\vee(6)$ versus $iii6$, i.e. a dominant triad with a suspension of its fifth or a minor triad on the third scale degree in first inversion. Again, both chords are identical in terms of their pitch-class content but differ with respect to their harmonic function. A full and detailed analysis of these changes is beyond the current scope and left for future research.

While one could have assumed that annotations of harmony, phrases, and cadences is a relatively straightforward task for music theory experts, our evaluation reveals that in many cases considerable modifications are necessary to arrive at an agreed-upon solution. This result

emphasizes the need for broader studies on inter-annotator (dis-)agreement [3, 13, 20] and moreover corroborates the weak status of ‘ground truth data’ for annotations with a high degree of interpretability [5, 10–12, 14, 16–19].

5. CONCLUSIONS

In this contribution we proposed a semi-automated workflow paradigm for streamlining the creation of annotated datasets by experts, and introduced, demonstrated, and evaluated one possible implementation. It bridges a gap between two by and large distinct skill sets: on the one hand researchers with expertise in computational methods and paradigms, and on the other hand expert music theorists and musicologists who contribute their considerable domain knowledge but may lack technical prowess.

Our proposal overcomes this ‘communicative barrier’ by providing a clearly defined workflow for the creation of annotated data that requires on behalf of the domain experts only the comprehension of the branching model outlined above and the usage of graphical user interfaces for label entry and revisions. Whereas our proposal greatly reduces the workload of annotators and reviewers, too, the automated notifications and validations, as well as the ease of communication and discussion, renders the curators its main beneficiaries, who usually bear the responsibility for a project’s coordination and success. As a proof of concept we have provided with this publication a GitHub repository with a new annotated dataset for which our workflow implementation was used. Future discussions within the MIR community may illuminate the repercussions of and alternatives to using proprietary hosting services in terms of cost, functional range, and data longevity/security.

Although our case study (building, providing, and evaluating corpora of annotated scores) is somewhat specific, we believe that a wide range of research projects will benefit from adopting or adapting it. We welcome alterations or alternative proposals, trusting that an active and constructive discussion around the topics laid out in this paper is valuable for the consolidation of data creation and annotation practices in the MIR community.

6. ACKNOWLEDGEMENTS

This research was supported by the Swiss National Science Foundation within the project “Distant Listening – The Development of Harmony over Three Centuries (1700–2000)” (Grant no. 182811). This project is being conducted at the *Latour Chair in Digital and Cognitive Musicology*, generously funded by Mr. Claude Latour.

7. REFERENCES

- [1] C. McKay, I. Fujinaga, M. Müller, and F. Wiering, “Building an Infrastructure for a 21st-Century Global Music Library,” in *Proceedings of the 16th International Music Information Retrieval Conference*, Malaga, Spain, 2015, pp. 1–2.
- [2] J. Yaolong, S. Howes, C. McKay, N. Condit-Schultz, J. Calvo-Zaragoza, and I. Fujinaga, “An Interactive Workflow for Generating Chord Labels for Homorhythmic Music in Symbolic Formats,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, Nov. 2019, pp. 862–869.
- [3] J. Degroot-Maggetti, T. de Reuse, L. Feisthauer, S. Howles, Y. Ju, S. Kokubu, S. Margot, N. N. López, and F. Upham, “Data Quality Matters: Iterative Corrections on a Corpus of Mendelssohn String Quartets and Implications for MIR Analysis,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, Online, 2020, pp. 432–438.
- [4] M. Gotham, P. Jonas, B. Bower, W. Bosworth, D. Rootham, and L. VanHandel, “Scores of scores: An openscore project to encode and share sheet music,” in *Proceedings of the 5th International Conference on Digital Libraries for Musicology - DLfM '18*, Paris, France, 2018, pp. 87–95.
- [5] J. Devaney, “Using Note-Level Music Encodings to Facilitate Interdisciplinary Research on Human Engagement with Music,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 205–217, Oct. 2020.
- [6] D. Lewis, D. Weigl, and K. Page, “Musicological Observations During Rehearsal and Performance: A Linked Data Digital Library for Annotations,” in *6th International Conference on Digital Libraries for Musicology*, ser. DLfM '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 1–8.
- [7] H. Schaffrath, *The Essen Folksong Collection*, D. Huron, Ed., Stanford, CA: Center for Computer Assisted Research in the Humanities.
- [8] M. Kemal Karaosmanoğlu, “A Turkish makam music symbolic database for music information retrieval: SymbTr,” in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, Porto, Portugal, 2012, pp. 223–228.
- [9] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, “Mirdata: Software for Reproducible Usage of Datasets,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019.
- [10] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, “The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets,” *Frontiers in Digital Humanities*, vol. 5, no. July, pp. 1–5, 2018.
- [11] C. Weiß, F. Zalkow, V. Arifi-Müller, M. Müller, H. V. Koops, A. Volk, and H. G. Grohgan, “Schubert Winterreise Dataset: A Multimodal Scenario for Music Analysis,” *Journal on Computing and Cultural Heritage*, vol. 14, no. 2, pp. 25:1–25:18, May 2021.
- [12] J. Albrecht and D. Shanahan, “Can I Have the Keys?: Key Validation Using a MIDI Database,” in *Proceedings of the 14th International Conference for Music Perception and Cognition*, San Fransisco, California, 2016, pp. 752–755.
- [13] J. B. L. Smith, J. Ashley Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, “Design and Creation of a Large-Scale Database of Structural Annotations,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 555–650.
- [14] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, and D. Tidhar, “OMRAS2 metadata project 2009,” in *Late-Breaking Session at the 10th International Conference on Music Information Retrieval*, Kobe, Japan, 2009.
- [15] F. Simonetta, S. Ntalampiras, and F. Avanzini, “ASMD: An automatic framework for compiling multimodal datasets with audio and scores,” in *Proceedings of the 17th Sound and Music Computing Conference*, Turin, Italy, Apr. 2020, pp. 40–46.
- [16] D. Harasim, C. Finkensiep, P. Ericson, T. J. O’Donnell, and M. Rohrmeier, “The Jazz Harmony Treebank,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, Montreal, Canada, 2020, pp. 207–215.
- [17] J. Hentschel, M. Neuwirth, and M. Rohrmeier, “The Annotated Mozart Sonatas: Score, Harmony, and Cadence,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 67–80, 2021.
- [18] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, “Annotator subjectivity in harmony annotations of popular music,” *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019.

- [19] C. Raffel and D. P. W. Ellis, “Extracting Ground Truth Information from MIDI Files: A Midifesto,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York City, NY, 2016, pp. 796–802.
- [20] A. Selway, H. V. Koops, A. Volk, D. Bretherton, N. Gibbins, and R. Polfreman, “Explaining harmonic inter-annotator disagreement using Hugo Riemann’s theory of ‘harmonic function’,” *Journal of New Music Research*, vol. 49, no. 2, pp. 136–150, 2020.
- [21] J. A. Burgoyne, J. Wild, and I. Fujinaga, “An Expert Ground-Truth Set for Audio Chord Recognition and Music Analysis,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, Miami, Florida, 2011, pp. 633–638.
- [22] M. Buhrmester, T. Kwang, and S. D. Gosling, “Amazon’s Mechanical Turk: A New Source of Inexpensive, Yet High-Quality, Data?” *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, Jan. 2011.
- [23] A. T. Nguyen, M. Halpern, B. C. Wallace, and M. Lease, “Probabilistic Modeling for Crowdsourcing Partially-Subjective Ratings,” *4th AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, pp. 149–158, 2016.
- [24] A. Dumitrache, “Truth in Disagreement: Crowdsourcing Labeled Data for Natural Language Processing,” Ph.D. dissertation, Amsterdam University, 2019.
- [25] M. Kutlu, T. McDonnell, M. Lease, and T. Elsayed, “Annotator Rationales for Labeling Tasks in Crowdsourcing,” *Journal of Artificial Intelligence Research*, vol. 69, pp. 143–189, Sep. 2020.
- [26] A. Leon, *Software Configuration Management Handbook*, 3rd ed. Boston & London: Artech House, 2015.
- [27] J. Humble, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, ser. The Addison-Wesley Signature Series. Upper Saddle River, NJ: Addison-Wesley, 2011.
- [28] T. Swicegood, *Pragmatic Version Control Using Git*, ser. The Pragmatic Starter Kit, S. Davidson Pfalzer, Ed. Raleigh, North Carolina: The Pragmatic Bookshelf, 2008, no. 1.
- [29] E. J. Hogbin Westby, *Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git*. Beijing: O’Reilly, 2015.
- [30] B. Appleton, S. P. Berczuk, R. Cabrera, and R. Orenstein, “Streamed Lines: Branching Patterns for Parallel Software Development,” in *Proceedings of the 1998 Pattern Languages of Programs Conference*, Monticello, Illinois, USA, 1998, pp. 1–67.

BEATNET: CRNN AND PARTICLE FILTERING FOR ONLINE JOINT BEAT DOWNBEAT AND METER TRACKING

Mojtaba Heydari

Frank Cwitkowitz

Zhiyao Duan

Department of Electrical and Computer Engineering

University of Rochester, 500 Wilson Blvd, Rochester, NY 14627, USA

{mheydari, fcwitkow}@ur.rochester.edu zhiyao.duan@rochester.edu

ABSTRACT

The online estimation of rhythmic information, such as beat positions, downbeat positions, and meter, is critical for many real-time music applications. Musical rhythm comprises complex hierarchical relationships across time, rendering its analysis intrinsically challenging and at times subjective. Furthermore, systems which attempt to estimate rhythmic information in real-time must be causal and must produce estimates quickly and efficiently. In this work, we introduce an online system for joint beat, downbeat, and meter tracking, which utilizes causal convolutional and recurrent layers, followed by a pair of sequential Monte Carlo particle filters applied during inference. The proposed system does not need to be primed with a time signature in order to perform downbeat tracking, and is instead able to estimate meter and adjust the predictions over time. Additionally, we propose an information gate strategy to significantly decrease the computational cost of particle filtering during the inference step, making the system much faster than previous sampling-based methods. Experiments on the GTZAN dataset, which is unseen during training, show that the system outperforms various online beat and downbeat tracking systems and achieves comparable performance to a baseline offline joint method.

1. INTRODUCTION

Rhythm plays an essential role in nearly all musical endeavors, including listening to, playing, learning, or composing music. This is why the estimation of rhythmic information, such as beat positions, downbeat positions and meter has always been an important subject of study in the field of Music Information Retrieval (MIR). Depending on the requirements and constraints imposed by the application at hand, these estimation tasks can either be performed in an offline or online fashion. Offline approaches are typically non-causal, meaning that they make predictions for a given time using data or features associated with a future time. These approaches are suitable for applications such

as music transcription, music search and indexing, and musicological analysis. Online approaches are causal, meaning that they operate using only past and present features. These are typically desirable for human-computer interaction (HCI) systems, which must make immediate predictions, like real-time music accompaniment systems.

Many offline methods have been proposed for beat tracking [1–3]. Most of them are unsupervised and attempt to utilize low-level features like onset strengths with some inference model to estimate beat positions within a music piece. However, with the growing success of deep learning, supervised beat tracking methods have become more prominent. Böck et al. [4] employed Recurrent Neural Networks (RNNs) to estimate beat positions; Various other neural network structures have also been proposed for onset detection and beat tracking [5, 6].

Some methods have also been proposed for online beat tracking. However, many of them, e.g., [4, 7–10], feed a sliding window of data into an offline model to estimate beat positions within upcoming frames. The sliding window strategy has several major drawbacks, including the discontinuity of beat predictions and the need for priming for predictions in the first window, which causes a delay [11]. Some other approaches involve inferring beat positions in real time using multi agent models [11–14], which initialize a set of agents with various hypotheses that try to validate their respective hypotheses based on observations across time.

The task of downbeat tracking is often considered to be more difficult than beat tracking. This is because a deeper understanding of rhythmic structure in music is required to be able to differentiate between beats and downbeats. Making matters worse, at the signal level, these two events have very similar characteristics. For instance, downbeats are not necessarily associated with stronger signal energy, nor do they necessarily feature a distinct percussive profile. Moreover, both beats and downbeats are likely to be the intersection of melodic and harmonic changes. These factors can make it challenging, and in some cases subjective, to distinguish between the two rhythmic events. For instance, for a 4/4 music piece with kick drum events on the first and third beats, it is hard to distinguish downbeats and determine whether the time signature is 4/4 or 2/4.

There has been some previous work on offline downbeat tracking, both as an isolated task and within a joint beat and downbeat tracking framework. Durand et al. [15–



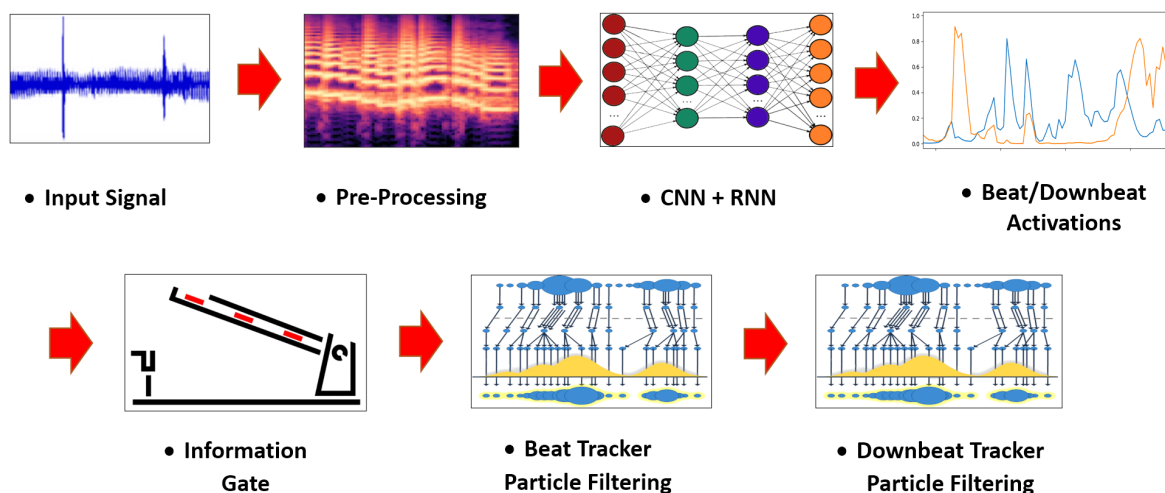


Figure 1. Overview of the joint beat, downbeat, and meter tracking procedure using the proposed BeatNet model.

[17] used some combinations of features and CNN structures to obtain downbeats. Giorgi et al. [18] proposed tempo-invariant convolutional filters for downbeat tracking. Peeters and Papadopoulos [19] performed joint beat and downbeat tracking by decoding hidden states using the Viterbi algorithm. Böck et al. [20] and Krebs et al. [21] employed an RNN structure for joint beat and downbeat tracking and only downbeat tracking using beat synchronous features, respectively. Furthermore, some recent works investigate Convolutional Recurrent Neural Network (CRNN) structures for beat and downbeat tracking. Fuentes et al. [22] showed that CRNN structures outperform RNNs in downbeat tracking when taking the input observations over a tatum grid. Cheng et al. [23] found that CRNN structures with larger receptive fields outperform other downbeat tracking models. Böck and Davies. [24] used a CNN and Temporal Convolutional Network (TCN) structure to improve the performance of their offline beat and downbeat tracking model, and also performed data augmentation to expose the neural network to more tempi.

The task of online downbeat tracking has received considerably less attention. Goto and Muranoka [13] introduced an unsupervised model which leverages a measure inference stage for detecting chord changes. In [25], the same beat tracking neural network with forward algorithm from [4, 20] is paired with [21] to estimate downbeats and other rhythmic patterns by extracting percussive and harmonic beat-synchronous features. It is important to note that this method must be primed with a known time signature and all possible rhythmic pattern choices. Liang [26] proposed an online downbeat tracking method which feeds a sliding window of data to an offline model [17]. This method is vulnerable to the sliding window strategy drawbacks described above.

Particle filtering is advantageous for two main reasons when it comes to online processing. The first reason is that it does not require future data. Popular maximum a posteriori (MAP) algorithms like the Viterbi algorithm and maximizer of the posterior marginals (MPM) smoothing algorithms, e.g. forward-backward, are not applicable to

online processing. The second reason is that, among the filtering methods which are causal, particle filtering is a general (non-parametric) approach which can be utilized to decode any unknown distribution. However, most music rhythmic analysis approaches that utilize particle filtering, e.g., [27–30], are classical and do not incorporate neural networks. Alternatively, in our previous work [31], we utilized a particle filtering inference model to infer beat positions using the activations produced by an RNN in an online fashion, but that approach does not attempt to estimate downbeats nor meter.

In this paper, we propose BeatNet, a novel online system for joint beat, downbeat, and meter tracking. The system produces beat and downbeat activations using a CNN and RNN combination, and performs inference using two particle filtering stages. The beat tracking stage outperforms state-of-the-art online beat tracking methods. The other stage simultaneously infers downbeats and time signature and achieves comparable results to state-of-the-art offline downbeat tracking models that require the time signature as input. In contrast, BeatNet actively monitors tempo and time signature changes over time. Finally, we introduce an information gate mechanism in the inference module to speed up the inference significantly, making our method suitable for many real-time applications.

2. METHOD

In this section, we describe BeatNet, our online system for joint beat, downbeat, and meter tracking, illustrated in Figure 1. BeatNet consists of a causal neural network stage for producing activations and a particle filtering stage for inference. The neural network comprises convolutional, recurrent and fully connected layers as described in section 2.2 which compute beat and downbeat activations for each frame of audio. The activations are fed to a two-stage particle filtering module to infer beat and downbeat positions and to estimate meter. The code for the BeatNet model is open-source¹, along with video demos and further docu-

¹ <https://github.com/mjhydri/BeatNet>

mentation.

2.1 Feature Representation

The input of the network module is a sequence of filterbank magnitude responses, each of which corresponds to one audio frame. Specifically, short-time Fourier transform (STFT) with a Hann window of the length of 93 ms and hop size of 46 ms is applied to the audio signal to compute the log-amplitude magnitude spectrogram. Then a logarithmically spaced filterbank ranging from 30 Hz to 17 kHz with 24 bands per octave is applied to yield a 136-d filterbank response. The first-order temporal difference of this response is also calculated and concatenated, resulting in a 272-d filterbank response vector for each frame.

We also experimented with alternative feature representations, including the 329-d hand-crafted feature set from [15], which comprises chroma features, onset strengths, low-frequency spectral features, and melodic constant-Q spectral features. The motivation for this feature set is to aggregate the harmonic, percussive, bass, and melodic content of the music. However the 272-d filterbank response feature set described above achieved notably better performance than these hand-crafted features, and was thus chosen for subsequent experiments.

2.2 Network Architecture

Following the common design of other similar works, we employ a convolutional-recurrent neural network (CRNN) architecture, illustrated in Figure 2, to process the input features in order to obtain beat and downbeat activations. Ideally, the convolution models relationships along the frequency axis, and the unidirectional recurrence models long-term relationships across time in a causal fashion.

The input features are fed into a 1D convolutional layer with 2 filters of kernel size 10, followed by ReLU activation. The two filter responses are max pooled with kernel size 2 along frequency and then concatenated into a single feature embedding for each frame. Then, a fully-connected layer with 150 neurons reduces the dimensionality of the embedding, and feeds it through two subsequent unidirectional Long Short-Term Memory (LSTM) layers, each with a hidden size of 150. The embedding is then fed through a final fully-connected layer and a softmax operation to obtain three activations which represent beat, downbeat, and non-beat, respectively. Note that due to the softmax function, the final activations for each class always sum to one.

2.3 Particle Filtering Inference

In this section, we discuss the two-stage online Monte Carlo particle filtering inference module, which generates the beat and downbeat predictions. Sequential Monte Carlo particle filtering is a sampling-based model which iteratively estimates any unknown distribution $p(x)$ by gathering a large number of independent samples from an arbitrary proposal distribution. The unknown distribution of

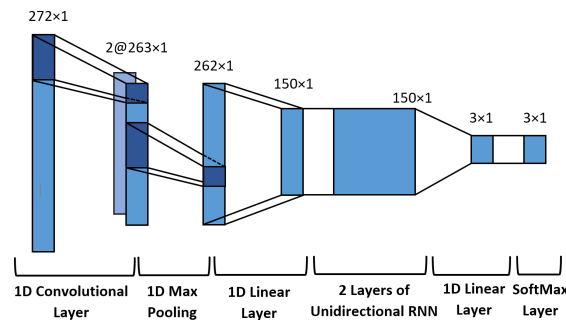


Figure 2. Proposed CRNN architecture for processing input features and computing beat and downbeat activations.

interest in our case, up to the K -th frame, is the following posterior $p(x_{1:K} | y_{1:K})$ of underlying beat or downbeat positions $x_{1:K}$ conditioned on beat observations $y_{1:K}$. It can be inferred according to the key equations below. For more detailed information, please refer to our previous work [31].

$$p(x) = \lim_{N \rightarrow \infty} \sum_{i=1}^N \frac{\omega^{(i)}}{\sum_{i=1}^N \omega^{(i)}} \delta(x - x^{(i)}), \quad (1)$$

$$p(x_{1:K}^{(i)} | y_{1:K}) \propto \prod_{k=1}^K p(y_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)}), \quad (2)$$

$$\omega_k^{(i)} = p(y_k | x_k^{(i)}) \omega_{k-1}^{(i)}, \quad (3)$$

where $\omega^{(i)}$ is the importance weight of particle i , and $\delta(\cdot)$ is the Dirac function. Eq. (1) describes the estimation of $p(x)$ using a large number of particles ($N \rightarrow \infty$) and their importance weights. Eq. (2) is a dynamic model which updates the posterior of each frame k using the transition (motion) and observation (correction) probabilities. Eq. (3) describes a recursive process to update the importance weights using the current observation and the importance weights of the previous step.

2.3.1 State spaces, transition and observation models

We use a cascade of two sequential Monte Carlo particle filters, one for beat tracking, and the other for downbeat and meter tracking. The state space and transition model of the beat estimator are similar to [32]. The beat state space is a type of 2D bar pointer model and its transition for the phase (horizontal) and the tempo (vertical) of the frame are described in Eqs. (4) and (5), respectively. The phase of frame k within a beat interval and the tempo at frame k are respectively denoted by $\phi_{b,k}$ and $\dot{\phi}_{b,k}$. A constant λ_b influences the intensity of potential jumps across the tempo axis.

We propose a new beat observation model in Eq. (6), where $x_{b,k}$ and $y_{b,k}$ are the beat state and beat observations at frame k . For non-beat states we allocate a small likelihood as $\gamma = 0.03$ instead of using the non-beat activation output from the neural network. For beat frames, since downbeats can also be considered beats, we assess

the maximum of the beat and downbeat activations. If the maximum exceeds a certain threshold, i.e., $T = 0.4$, then it is set as the likelihood; Otherwise, γ is used. When γ is used, we also bypass the costly re-sampling step in the beat particle filtering module. Therefore, the threshold serves as an *information gate*, through which the computational cost is significantly reduced.

$$\phi_{b,k} = (\phi_{b,k-1} + \dot{\phi}_{b,k-1}) \bmod (\phi_b^{max} + 1), \quad (4)$$

$$p(\dot{\phi}_{b,k} | \dot{\phi}_{b,k-1}) = \begin{cases} \exp\left(-\lambda_b \left| \frac{\dot{\phi}_{b,k}}{\dot{\phi}_{b,k-1}} \right| \right) & \text{if } \phi_{b,k} = 0 \\ \mathbb{1}(\dot{\phi}_{b,k} = \dot{\phi}_{b,k-1}) & \text{if } \phi_{b,k} > 0 \end{cases}, \quad (5)$$

$$p(y_{b,k} | x_{b,k}) = \begin{cases} \max(b_k, d_k) & \text{if } \phi_{b,k} = 0 \text{ and} \\ & \max(b_k, d_k) \geq T \\ \gamma & \text{otherwise} \end{cases}, \quad (6)$$

The second particle filter detects downbeats and the time signature jointly. The state space is similar to that of beat tracking. However, here we introduce $\dot{\phi}_{d,k}$ corresponding to the meter, i.e., $\dot{\phi}_{d,k} \in 2, 3, \dots, \dot{\phi}_d^{max}$, and $\phi_{d,k}$ to describe the phase of the beat within the bar interval, i.e. $\phi_{d,k} \in 0, 1, 2, \dots, \phi_d^{max}$. Eqs. (7) and (8) describe the phase and meter transition models. We only let meter change at the states belonging to the downbeat area i.e. $\phi_{d,k} = 0$, and λ_d is a constant parameter that decides what percent of the particles jump to other meters at the downbeat states. Also, in Eq. (9) we define the observation model used in the downbeat particle filter. The first states within the bar (downbeat area) take the downbeat activation and the rest of them (beat states) take the beat activation. Note that as the second particle filter operates less often, i.e., only when a beat is detected, no information gate is needed here.

$$\phi_{d,k} = (\phi_{d,k-1} + \dot{\phi}_{d,k-1}) \bmod (\phi_d^{max} + 1), \quad (7)$$

$$p(\dot{\phi}_{d,k} | \dot{\phi}_{d,k-1}) = \begin{cases} \lambda_d & \text{if } \phi_{d,k} = 0 \text{ and} \\ & \dot{\phi}_{d,k} \neq \dot{\phi}_{d,k-1} \\ 1 - \lambda_d & \text{if } \phi_{d,k} = 0 \text{ and} \\ & \dot{\phi}_{d,k} = \dot{\phi}_{d,k-1} \\ \mathbb{1}(\dot{\phi}_{d,k} = \dot{\phi}_{d,k-1}) & \text{if } \phi_{d,k} > 0 \end{cases}, \quad (8)$$

$$p(y_{d,k} | x_{d,k}) = \begin{cases} d_k & \text{if } \phi_{d,k} = 0 \\ b_k & \text{if } \phi_{d,k} > 0 \end{cases}, \quad (9)$$

2.3.2 Inference process

Algorithm 1 describes the inference process in detail. Particles are initialized randomly for both inference modules by sampling from a uniform distribution within their state space. By proceeding to a new frame, particles within the beat state space are transferred to the new positions by sampling from the transition model, and new importance weights are then calculated and normalized. If the activations of the frame satisfy the information gate condition, the re-sampling process is invoked for all particles; Otherwise, the re-sampling step is skipped as it is likely a non-beat frame. Afterwards, if the median of the particles is within the tolerance window T_w of a beat area and the

time of the current frame is longer enough than the last detected beat considering the estimated tempo, the frame is classified as a beat frame. A similar process follows for the downbeat and meter inference module.

Algorithm 1 Joint Inference Procedure

beats, downbeats, meters = [], [], []

Sample $(x_{b,0}^{(i)}) \sim \mathcal{U}(S_b)$, $(x_{d,0}^{(j)}) \sim \mathcal{U}(S_d)$

Set $w_{b,0}^{(i)} = \frac{1}{N_b}$, $w_{d,0}^{(j)} = \frac{1}{N_d}$

for $k = 1$ to K **do**

Sample $(x_{b,k}^{(i)}) \sim p(\phi_{b,k}^{(i)} | \phi_{b,k-1}^{(i)})$, $p(\dot{\phi}_{b,k}^{(i)} | \dot{\phi}_{b,k-1}^{(i)})$

$\tilde{\omega}_{b,k}^{(i)} = \omega_{b,k-1}^{(i)} \times p(y_{b,k} | x_{b,k}^{(i)}) \quad \forall i \in N_b$

$\omega_{b,k}^{(i)} = \frac{\tilde{\omega}_{b,k}^{(i)}}{\sum \tilde{\omega}_{b,k}^{(i)}} \quad \forall i \in N_b$

if $\max(b_k, d_k) \geq T$ **then**

Resample $x_{b,k}^{(i)}$ according to $\omega_{b,k}^{(i)}$

end if

if $\text{median}(\phi_{b,k}^{(i)}) < T_w$ **and** $(k\Delta - \text{beats}[-1]) >$

$0.4 \text{median}(\dot{\phi}_{b,k}^{(i)})$ **then**

Append (beats, $k\Delta$)

Sample $(x_{d,k}^{(j)}) \sim p(\phi_{d,k}^{(j)} | \phi_{d,k-1}^{(j)})$, $p(\dot{\phi}_{d,k}^{(j)} | \dot{\phi}_{d,k-1}^{(j)})$

$\tilde{\omega}_{d,k}^{(j)} = \omega_{d,k-1}^{(j)} \times p(y_{d,k} | x_{d,k}^{(j)}) \quad \forall j \in N_d$

$\omega_{d,k}^{(j)} = \frac{\tilde{\omega}_{d,k}^{(j)}}{\sum \tilde{\omega}_{d,k}^{(j)}} \quad \forall j \in N_d$

Resample $x_{d,k}^{(j)}$ according to $\omega_{d,k}^{(j)}$

if $\text{mode}(\phi_{d,k}^{(j)}) == 0$ **then**

append (downbeats, $k\Delta$)

append (meters, $\text{mode}(\dot{\phi}_{d,k}^{(j)})$)

end if

end if

end for

A visualization of the inference process is presented in Figure 3. Each pair of plots demonstrates one step of the inference procedure, where the top and the bottom plots show the beat and downbeat tracking process, respectively. In the first pair of plots, the beat particles are initialized randomly. In the second pair, the first beat is detected and the downbeat state particles are simultaneously initialized randomly. In the third pair, beat tracking particles have converged, but the downbeat particles have not yet converged. Here the downbeat clutter is located in the lowest row of the downbeat state space, which represents a six-beat time signature. The next few plot pairs illustrate convergence of both the beat and downbeat particles, producing an estimate of the tempo and beat phase (top plots), and the meter and bar phase (bottom plots).

3. EXPERIMENTS

3.1 Methodology

In order to analyze the performance of BeatNet, we compare it to several publicly available online beat tracking methods, We additionally provide the online downbeat tracking performance of BeatNet for each of the experiments. Following standard evaluation practices, in

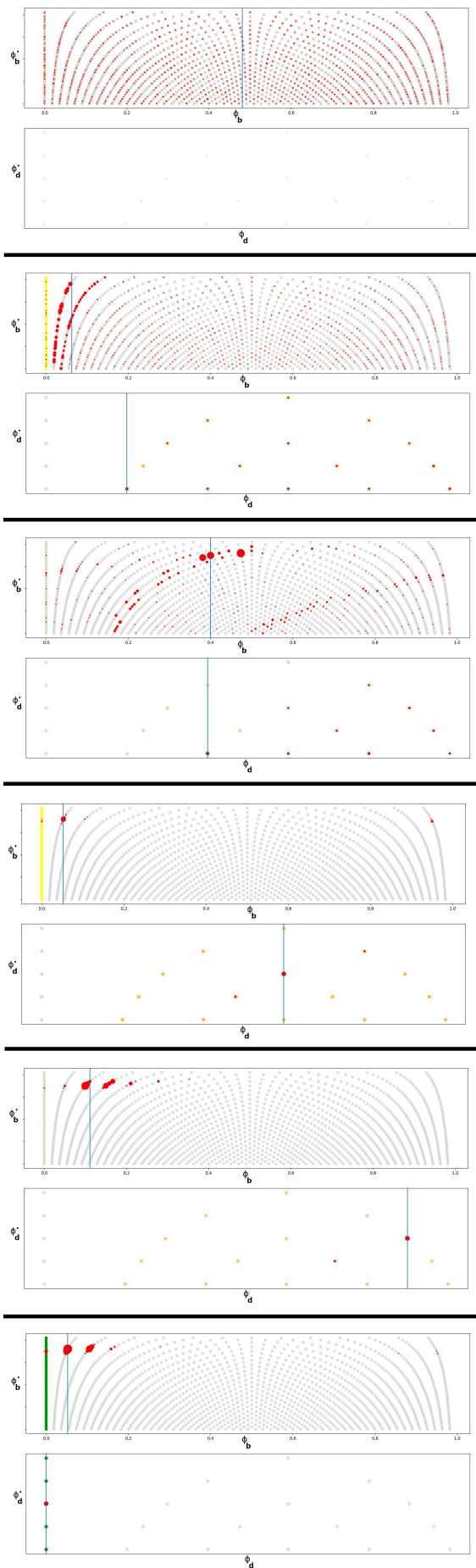


Figure 3. Inference example, detailed in Section 2.3.2.

Dataset	# Files	Total Length
Ballroom [33,34]	685	5 h 57 m
Beatles [2]	180	8 h 9 m
Carnatic [35]	176	16 h 38 m
GTZAN [36,37]	999	8 h 20 m
Rock Corpus [38]	200	12 h 53 m

Table 1. Datasets used for training and testing.

this work, F-measure with a tolerance window of $T_w = \pm 70 ms$ is used as the evaluation metric for all experiments.

We utilize all five datasets [2, 33–38] described in Table 1 for training, validation, and testing, with different splits and arrangements for various experiments. In the first comparison, we evaluate BeatNet on the GTZAN dataset, which covers 10 different music genres and was unseen from training of all comparison methods. In order to demonstrate the generalization ability of our approach, we also experiment with two other comparison schema where we respectively set aside the Ballroom and Rock datasets during training and use them entirely for evaluation. Note that all of the supervised comparison methods included the Ballroom and Rock datasets in their training set, so we only compare BeatNet with unsupervised methods in these cases.

3.2 Training Details

For training the beat and downbeat activation neural network described in Section 2.2, all weights and biases are initialized randomly, and the network is trained using Adam optimizer with a learning rate of 5×10^{-4} and a batch size of 200. Since the number of non-beat frames within a music piece is typically much larger than the number of beat and downbeat frames, our objective function is chosen to be weighted cross entropy loss of the beat, downbeat, and non-beat, where the weights are inverse proportional to the frequency of occurrence of each type of frame. Batches comprise 8-second long excerpts randomly sampled from each audio file available in the training set. Given that some datasets (e.g., Beatles) contain full songs and others (e.g., Ballroom) contain short excerpts of songs, we sample from longer audio files more often during the training Batch creation. Training proceeds until the performance on the validation set has not increased over a span of 20 epochs for a given experiment.

3.3 Results and Discussion

The evaluation results of the proposed BeatNet model and comparison methods are presented in Table 2. All online comparison methods only perform beat tracking, and all except IBT [11] and Aubio [9] are supervised methods using deep neural networks. We can see that the online beat tracking portion of BeatNet outperforms all comparison methods. The Böck FF [6, 20] and Don’t Look Back (DLB) models [31] achieve the next best performance.

<i>Method</i>	<i>F-Measure Beats</i>	<i>F-Measure Downbeats</i>
Comparison of Online Methods		
<i>GTZAN Dataset</i>		
Aubio [9]	57.09	—
BeatNet	<u>75.44</u>	<u>46.49</u>
Böck ACF [4]	64.63	—
Böck FF [6, 20]	74.18	—
DLB [31]	73.77	—
IBT [11]	68.99	—
<i>Ballroom Dataset</i>		
Aubio [9]	56.73	—
BeatNet	<u>77.41</u>	<u>47.45</u>
IBT [11]	70.79	—
<i>Rock Corpus Dataset</i>		
Aubio [9]	59.83	—
BeatNet	<u>73.13</u>	<u>44.98</u>
IBT [11]	68.55	—
Comparison of Offline Methods		
<i>GTZAN Dataset</i>		
BeatNet + DBN	<u>80.64</u>	<u>54.07</u>
Böck [20]	79.09	51.36

Table 2. Comparison of BeatNet with other beat and downbeat tracking methods on various datasets.

Böck FF uses the forward algorithm to estimate beats in a similar manner to the other online joint model described earlier [25]. Aside from the different neural network structures, the beat tracking inference processes of the DLB model [31] and BeatNet are largely the same. The main difference is that the latter benefits from the information gate, which decreases the computational time drastically.

Additionally, we report the performance comparison with an offline joint beat and downbeat tracking model [20] on the GTZAN dataset. In this case, we replaced the particle filtering modules of BeatNet with the DBN used in [20] to directly compare neural network architectures in BeatNet and [20]. Same to [20], we also provided the time signatures to the DBN. For [20], we utilized the Madmom [39] library, which is the official implementation of the paper. Note that due to the existence of different GTZAN beat annotations, the reported offline results obtained by us differ from those of the original paper [20]. However, since we used the same annotations for all of the experiments, the offline comparison is valid. As the table suggests, with the same DBN estimator, both neural networks yield similar results for beat tracking. However, for downbeat tracking, the BeatNet architecture yields marginally better performance. These results are interesting, since we are comparing a causal network to a non-causal network which leverages bidirectional recurrence. However, our network is larger and contains more parameters.

The comparison between BeatNet (second row) and [20] (last row) is also interesting. BeatNet underperforms [20] by 3.65% on beat tracking and by 4.9% on

downbeat tracking. However, it is noted that BeatNet is an online method and it does not require the time signature input, while [20] is offline method and it requires the time signature input.

One limitation of our model is that the performance of the downbeat tracker depends on the beat tracker. This means that if the beat tracker makes incorrect predictions, errors will carry through to the downbeat tracker. This is a common characteristic of cascade systems such as [25]. Another limitation is the high computation cost of sequential Monte Carlo particle filtering methods. This limitation has been partially addressed in our previous work [31] by using efficient models, e.g., [32] in the inference stage. The information gate proposed in this paper further reduces the computational cost.

On a typical windows machine with AMD Ryzen 9 3900X CPU and 3.80 GHz clock, the processing time for the pre-processing stage and passing a frame through the neural network is 0.12 ms and 0.01 ms, respectively. These times are relatively insignificant, as the inference process takes more time. The inference process takes 5.23 and 8.87 seconds using 1000 and 1750 particles, respectively, to process a 30-sec long music excerpt. This is much faster than the previous sampling-based model [31] which took 21.30 seconds using a 1000 particle setup. Larger numbers of particles lead to longer processing times with a roughly linear relationship. Hence, we reported these results using 1500 particles for the beat inference block and 250 for the downbeat inference block (1750 particles in total) to keep the process minimal.

4. CONCLUSION

We proposed BeatNet, a new online system for joint beat, downbeat, and meter tracking. The system incorporates a convolutional-recurrent neural network for generating beat and downbeat activations in each audio frame, and a two-stage particle filtering algorithm to estimate tempo, beats, downbeats, and musical meter. An information gate is added to the beat tracking particle filter to skip many re-sampling steps hence reduces the computational cost significantly. The system is compared to multiple online and offline methods under various experimental conditions, and it achieves superior performance for both online beat and downbeat tracking.

5. ACKNOWLEDGEMENT

This work has been partially supported by the National Science Foundation grants 1846184 and DGE-1922591.

6. REFERENCES

- [1] D. Ellis, “Beat tracking with dynamic programming,” *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.
- [2] M. E. P. Davies, N. Degara, and M. D. Plumbley, “Evaluation methods for musical audio beat tracking algorithms,” in *Technical Report C4DM-TR-09-06*,

- Centre for Digital Music, Queen Mary University of London, 2009.
- [3] F. Gouyon, *A computational approach to rhythm description Audio features for the computation of rhythm periodicity functions and their use in tempo induction and music content processing*. PhD thesis, Universitat Pompeu Fabra, 2005.
- [4] S. Böck and S. Schedl, “Enhanced beat tracking with context-aware neural networks,” in *In Proc. of the 14th International Conference on Digital Audio Effects (DAFx-11)*, 2011, pp. 135–140.
- [5] M. E. P. Davies and S. Böck, “Temporal convolutional networks for musical audio beat tracking,” in *In Proc. of the 27th European Signal Processing Conference (EUSIPCO)*, 2019.
- [6] S. Böck, F. Krebs, and G. Widmer, “A multi-model approach to beat tracking considering heterogeneous music styles,” in *In Proc. of the 15th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2014, pp. 603–608.
- [7] M. E. P. Davies, P. M. Brossier, and M. D. Plumbley, “Beat tracking towards automatic musical accompaniment,” *Audio Eng. Soc. Conv. Spring Prepr.*, vol. 2, pp. 751–757, 2005.
- [8] A. Gkiokas and V. Katsouros, “Convolutional neural networks for real-time beat tracking: A dancing robot application,” in *In Proc. of the 18th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2017, pp. 286–293.
- [9] P. M. Brossier, “Automatic annotation of musical audio for interactive applications,” P. dissertation, Ed., Queen Marry University, London, UK, August 2006, pp. 58–102.
- [10] A. Mottaghi, K. Behdin, A. Esmaeili, M. Heydari, , and F. Marvasti, “OBTAIN: Real-time beat tracking in audio signals index terms—onset strength signal, tempo estimation, beat onset, cumulative beat strength signal, peak detection,” *International Journal of Signal Processing Systems*, pp. 123–129, 2017.
- [11] J. L. Oliveira, F. Gouyon, L. G. Martins, , and L. P. Reis, “IBT: A real-time tempo and beat tracking system,” in *In Proc. of the 11th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2014, pp. 291–296.
- [12] M. Goto., “AIST annotation for the RWC music database.” in *In Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2006, pp. 359–360.
- [13] M. Goto and Y. Muraoka, “Real-time rhythm tracking for drumless audio signals: Chord change detection for musical decisions,” *Speech Communication*, vol. 27, no. 3, pp. 311–335, 1999.
- [14] M. Goto and Y. Muraok, “Music understanding at the beat level real-time beat tracking for audio signals,” in *In Proceedings of IJCAI- 95 Workshop on Computational Auditory Scene Analysis*, 1995, pp. 67–75.
- [15] S. Durand, J. P. Bello, B. David, , and G. Richard, “Robust downbeat tracking using an ensemble of convolutional networks,” *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, vol. 25, no. 1, pp. 255–261, 2017.
- [16] S. Durand, J. Bello, B. D., and G. Richard, “Downbeat tracking with multiple features and deep neural networks,” in *In Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2015.
- [17] S. Durand, J. P. Bello, B. D., and G. Richard, “Feature adapted convolutional neural networks for downbeat tracking,” in *In Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2016.
- [18] B. D. Giorgi, M. Mauch, , and M. Levy, “Downbeat tracking with tempo-invariant convolutional neural networks,” in *In Proc. of the 17th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2020, pp. 216–222.
- [19] G. Peeters and H. Papadopoulos, “Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation,” *IEEE Transactions on audio, speech, and language processing*, vol. 19, no. 6, August 2011.
- [20] S. Böck, F. Krebs, and G. Widmer, “Joint beat and downbeat tracking with recurrent neural networks,” in *In Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2016.
- [21] F. Krebs, S. Böck, M. Dorfer, and G. Widmer, “Downbeat tracking using beat-synchronous features and recurrent neural networks,” in *In Proc. of the 17th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2016.
- [22] M. Fuentes, B. Mcfee, H. C. Crayencour, S. Essid, and J. P. Bello, “Analysis of common design choices in deep learning systems for downbeat tracking,” in *Proc. of the 19th Int. Society for Music Information Retrieval Conf.*, 2018.
- [23] T. Cheng, S. Fukayama, and M. Goto, “Joint beat and downbeat tracking based on CRNN models and a comparison of using different context ranges in convolutional layers,” in *In Proc. of the International Computer Music Conference (ICMC)*, 2016.
- [24] S. Böck and M. E. P. Davies, “Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation,” in *Proc. of the 21th Int. Society for Music Information Retrieval Conf.*, 2020, pp. 574–582.
- [25] S. Böck, F. Krebs, A. Durand, S. Pöll, and R. Balsyte, “ROBOD: a real-time online beat and offbeat drummer sebastian,” in *2017 IEEE signal processing cup*, 2017.
- [26] C.-Y. Liang, “Implementing and adapting a downbeat tracking system for real-time applications,” in *Master Thesis*, Carnegie Mellon University, 2017.

- [27] S. Hainsworth and M. Macleod, "Particle filtering applied to musical tempo tracking." *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2385–2395, 2004.
- [28] S. Hainsworth and M. D. Macleod, "Beat tracking with particle filtering algorithms," in *Proc. in the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.
- [29] A. T. Cemgil and B. Kappen, "Monte carlo methods for tempo tracking and rhythm quantization," *Journal of Artificial Intelligence Research*, vol. 18, pp. 111–222, 2003.
- [30] F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer, "Inferring metrical structure in music using particle filters," *IEEE/ACM Transactions on audio, speech, and language processing*, vol. 23, no. 5, pp. 111–222, May 2015.
- [31] M. Heydari and Z. Duan, "Don't look back: An online beat tracking method using RNN and enhanced particle filtering," in *In Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021.
- [32] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," in *In Proc. of the 16th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2015.
- [33] F. Gouyon, A. P. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano., "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, 2006.
- [34] F. Krebs, S. Böck, and G. Widmer., "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, 2013.
- [35] A. Srinivasamurthy and X. Serra, "A supervised approach to hierarchical metrical cycle tracking from audio music recordings," in *In Proc. of the IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.
- [36] U. Marchand and G. Peeters, "Swing ratio estimation," in *In Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx)*, 2015.
- [37] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, 2002.
- [38] T. de Clercq and D. Temperley., "A corpus analysis of rock harmony," *Popular Music*, vol. 30, no. 1, pp. 47–70, 2011.
- [39] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, , and G. Widmer, "Madmom: A new python audio and music signal processing library," in *in Proc. ACM Multimed. Conf, MM 2016*,, 2016, pp. 1174–1178.

JOINT ESTIMATION OF NOTE VALUES AND VOICES FOR AUDIO-TO-SCORE PIANO TRANSCRIPTION

Yuki Hiramatsu¹ Eita Nakamura^{1,2} Kazuyoshi Yoshii^{1,3}

¹Graduate School of Informatics, Kyoto University, Japan

²The Hakubi Center for Advanced Research, Kyoto University, Japan

³PRESTO, Japan Science and Technology Agency (JST), Japan

{hiramatsu, enakamura, yoshii}@sap.ist.i.kyoto-u.ac.jp

ABSTRACT

This paper describes an essential improvement of a state-of-the-art automatic piano transcription (APT) system that can transcribe a human-readable symbolic musical score from a piano recording. Whereas estimation of the pitches and onset times of musical notes has been improved drastically thanks to the recent advances of deep learning, estimation of note values and voice labels, which is a crucial component of the APT system, still remains a challenging task. A previous study has revealed that (i) the pitches and onset times of notes are useful but the performed note durations are less informative for estimating the note values and that (ii) the note values and voices have mutual dependency. We thus propose a bidirectional long short-term memory network that jointly estimates note values and voice labels from note pitches and onset times estimated in advance. To improve the robustness against tempo errors, extra notes, and missing notes included in the input data, we investigate data augmentation. The experimental results show the efficacy of multi-task learning and data augmentation, and the proposed method achieved better accuracies than existing methods.

1. INTRODUCTION

The ultimate goal of automatic piano transcription (APT) is to convert a piano recording into a human-readable musical score that can be used for music analysis and performance [1]. This is a challenging task because of the polyphonic nature of piano music; musical notes form weakly-synchronous multiple streams called *voices* running in parallel. Much work on APT aims to estimate not a musical score but a *piano roll* from a music signal, *i.e.*, estimate the quantized pitches and non-quantized onset times of musical notes [2–7]. Although noticeable research progress has independently been made for multipitch detection [8–10] and rhythm transcription [11, 12], estimation of note values and voice labels, which is crucial for score typesetting

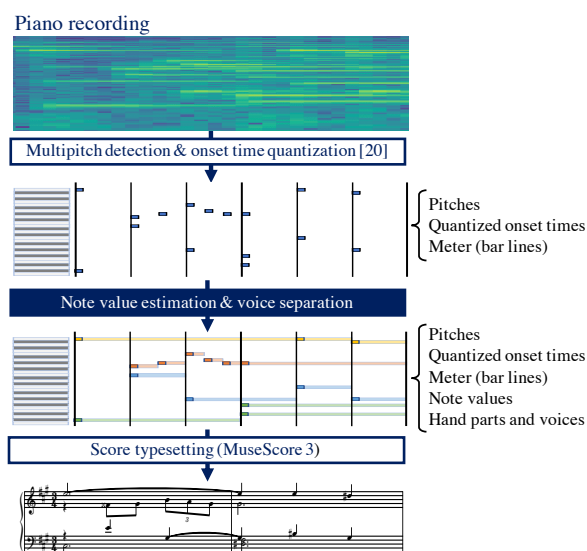


Figure 1. Overview of proposed method that jointly estimates note values and voice labels from transcribed pitches and onset times.

with high readability [13], still remains a challenging task. Note that the note value represents the duration of a note on a symbolic musical score, and the voice label of a note specifies one of the voices of the upper or lower staff (right- or left-hand part) the note belongs to (Fig. 1).

Several attempts have been made for estimating voice labels for piano scores having no voice labels [14–17]. Under an assumption that each voice has a strictly monophonic structure and note values are already transcribed with a certain degree of accuracy, voice labels can be estimated accurately [18, 19]. In practice, however, each voice has a homophonic structure consisting of concurrent notes (chords) and accurate estimation of note values is still an open problem. To deal with such a realistic situation, a state-of-the-art APT method uses a rule-based cost function with limited performance [20]. This calls for a principled statistical approach based on modern deep learning.

Note value estimation has relatively scarcely been studied [4, 21] and is still considered a challenging task [20]. Nakamura *et al.* [21] conducted a detailed statistical analysis and found that (i) the pitches and onset times of notes are useful but the performed note durations are less informative for estimating the note values and that (ii) the note

values and voices have mutual dependency. The second conclusion derives from the fact that in a voice stream, the offset time of a note usually matches the onset time of the next note, or equivalently, short rests are rarely inserted between notes [4]. This indicates that joint estimation of note values and voice labels is an effective way of bringing improvements on both tasks.

In this paper, we propose a deep neural network (DNN) that estimates note values and voice labels jointly from a transcribed sequence of pitches and onset times for audio-to-score APT. Specifically, we train a bidirectional long short-term memory (BiLSTM) network in multi-task learning. We also investigate data representation, network architecture, post-processing, and data augmentation, which are considered to have an impact on the estimation performance. We report experimental evaluation conducted on datasets of classical and popular music, to investigate the efficacy of the joint estimation framework.

Our main contribution is to propose joint estimation of note values and voice labels based on deep learning. Combined with the state-of-the-art methods for multipitch detection and rhythm transcription used in the latest piano transcription system [20], we can achieve the state-of-the-art performance of audio-to-score APT. Another contribution is to propose new evaluation metrics for the polyphonic music transcription task, extending the one proposed in [22] to deal with voice labels. The example transcription results and the source code for the evaluation tool are available on the accompanying webpage¹.

2. RELATED WORK

This section reviews methods for audio-to-score piano transcription, note value estimation, and voice separation.

2.1 Audio-to-Score Piano Transcription

Some piano transcription methods that can yield symbolic piano scores have been proposed, and the methods consisting of multi-stage processing [13, 20] achieved high accuracies. Cogliati *et al.* [13] proposed a transcription method that performs rhythm quantization and voice estimation for a piano performance MIDI file and generates a piano score. This method uses metrical, stream, and harmonic structures from the MIDI sequence estimated by a probabilistic model by Temperley [4]. Shibata *et al.* [20] proposed a state-of-the-art transcription method that can generate a piano score from audio signals with multi-stage processing. The method first estimates from a piano recording a performance MIDI sequence consisting of pitches, onset and offset times, and velocities using a convolutional neural network (CNN). The onset times are then quantized using a hidden Markov model (HMM). After note values and voice labels are separately estimated, piano scores are generated using MuseScore 3. In this study, we jointly estimate note values and voice labels, and aim to improve the accuracies that were lower than those of pitches and onset times in the method.

There are also end-to-end approaches that directly estimate musical scores from audio signals. Carvalho *et al.* [23] proposed a seq2seq model that estimates from an audio signal a piano score represented in the Lilypond music notation language. Román *et al.* [24] used a convolutional recurrent neural network (CRNN) that estimates a musical score represented in the **kern format. The network is trained with a connectionist temporal classification (CTC) loss function. These end-to-end methods have been tested only on very short or synthetic recordings, and there has been no account in the literature describing how well they perform in practice.

2.2 Note Value Estimation

Note value estimation is a difficult problem because note values do not always correspond to the performed durations [21]. Temperley [4] proposed a rhythm quantization method based on a probabilistic model. The method quantizes onset times by estimating beat positions. After voice labels are estimated, an offset time is set to the onset time of the next note in the same voice. One of the problems of the method is outputting no rests that are essential to make scores easy to read. Nakamura *et al.* [21] proposed a method based on Markov random fields. The method consists of a context model that represents a distribution of note values given pitches and onset times, and a performance model that generates actual performance durations from note values. It was shown that the performance model had a small impact on the estimation performance [21]. Therefore, we estimate note values only from pitches and onset times and do not use performed durations.

2.3 Voice Separation

Voice separation aims to divide musical notes into groups of notes representing musical streams. Karydis *et al.* [15] proposed a rule-based voice separation method for symbolic piano scores. The method is based on vertical integration, which integrates notes with the same onset time and the same duration, and horizontal integration, which integrates notes close in time and pitch. While this method can deal with homophonic voices, most other methods can only estimate monophonic voices. McLeod *et al.* [18] proposed a voice separation method for MIDI data using an HMM, and achieved high accuracy. Valk *et al.* [19] proposed a DNN-based voice separation method. The method uses a deep feedforward neural network that classifies each note represented by 33 handcrafted features into five classes. In piano transcription, these existing voice separation methods are not appropriate because voices often contain chords and durations are not estimated precisely. Explicit hand-part and voice labeling (rather than clustering) are also necessary for typesetting piano scores; for example, voice labels are used for determining the directions of note stems. Shibata *et al.* [20] proposed a cost-function-based voice separation method. Although this method is applicable to the situation of piano transcription, there is room for improvement in accuracy. We attempt to develop an improved DNN-based method.

¹ <https://nvvest.github.io>

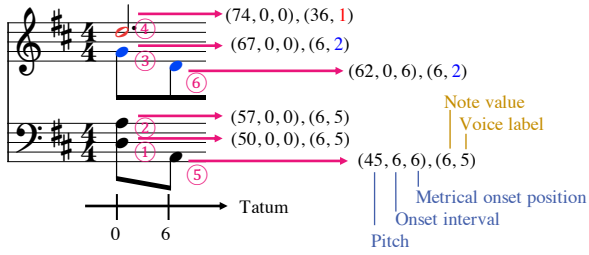


Figure 2. Data representation of the input and the output of the BiLSTM network.

3. PROPOSED METHOD

In this section, we propose BiLSTM networks that jointly estimate note values and voice labels, post-processing methods that correct the estimated note values, and data augmentation methods.

3.1 Problem Specification

Each note $\mathbf{z}_n = (p_n, o_n, d_n, v_n)$ in a musical score is represented by a pitch p_n , an onset time o_n , a note value d_n , and a voice label v_n . The pitch $p_n \in \{0, \dots, 127\}$ is represented by a MIDI note number ($60 = C4$). The onset time $o_n \geq 0$ and the note value $d_n \in \{0, \dots, 479\}$ are represented by integers (one measure is divided into 48 units); a zero note value is used for a grace note. Note that different meters have different tatum units: for example, a quarter note is represented as 12 in 4/4 time and 16 in 3/4 time. The maximum number of voices in each hand part is set to 4, following the convention for score notation used in score editing software such as MuseScore3 and Finale; labels $v_n = 1, 2, 3, 4$ are used for the right-hand part and $v_n = 5, 6, 7, 8$ for the left-hand part. We represent a piano score as a sequence of notes $\mathbf{Z} = \{\mathbf{z}_n\}_{n=1}^N$, where notes are arranged in the increasing order of onset times, and notes with the same onset time are ordered according to the pitches. Our goal is to estimate the note values and voice labels $\{(d_n, v_n)\}_{n=1}^N$ from a given set of pitches and onset times $\{(p_n, o_n)\}_{n=1}^N$.

3.2 BiLSTM Network

We propose a BiLSTM network that estimates note values and voice labels from pitches and onset times. We first represent the onset time o_n as the interval from the previous onset $i_n \in \{0, \dots, 767\}$ and the metrical position $b_n \in \{0, \dots, 47\}$ calculated as follows:

$$i_n = o_n - o_{n-1}, \quad b_n = o_n \bmod 48. \quad (1)$$

The input is then represented as $\mathbf{X} = \{(p_n, i_n, b_n)\}_{n=1}^N$ and the output is $\mathbf{Y} = \{(d_n, v_n)\}_{n=1}^N$ (Fig. 2).

The proposed network architecture is shown in Fig. 3(a). Each musical note of input \mathbf{X} is represented as a $(128 \times 768 \times 48)$ -dimensional one-hot vector. These one-hot vectors are first transformed to 25-dimensional feature vectors by a fully connected layer. The resulting vectors are then transformed to 50-dimensional vectors (latent representations) through a BiLSTM layer. Note value probabilities $\pi_n(\mathbf{X}) = \{\pi_n(d; \mathbf{X})\}_{d=0}^{479}$ and voice label probabilities

$\phi_n(\mathbf{X}) = \{\phi_n(v; \mathbf{X})\}_{v=1}^8$ are separately calculated at each time step n after passing through fully connected layers and softmax layers, where $\pi_n(d; \mathbf{X})$ denotes the probability that the n -th note has duration d and $\phi_n(v; \mathbf{X})$ denotes the probability that the n -th note is in voice v .

We train the network by minimizing a cross-entropy loss function given by

$$\mathcal{L} = \mathcal{L}_d + \mathcal{L}_v, \quad (2)$$

where

$$\mathcal{L}_d = - \sum_{n=1}^N \log \pi_n(d_n^*; \mathbf{X}), \quad (3)$$

$$\mathcal{L}_v = - \sum_{n=1}^N \log \phi_n(v_n^*; \mathbf{X}), \quad (4)$$

where d_n^* and v_n^* are the correct note value and voice label, respectively. In the inference step, note values and voice labels are estimated from given pitches and onset times \mathbf{X} as follows:

$$\hat{d}_n = \arg \max_d \pi_n(d; \mathbf{X}), \quad (5)$$

$$\hat{v}_n = \arg \max_v \phi_n(v; \mathbf{X}), \quad (6)$$

where \hat{d}_n and \hat{v}_n indicate the estimated note value and voice label, respectively.

3.3 Alternative Network Architectures

The network architecture in Fig. 3(a) is a simple joint network that equally treats the note value and voice label probabilities. We call this network SIM (simultaneous). We examine other network architectures shown in Fig. 3. As discussed in the Introduction, the voice structure has a strong impact on determining note values. To reflect this dependency structure, we propose the second network architecture (VLF; voice label first). In this network (Fig. 3(b)), voice labels are estimated first and note values are estimated with the latent representations used to estimate voice labels. For comparison, we also consider the third network architecture (NVF; note value first) that has a reverse structure (Fig. 3(c)). The networks SIM, VLF, and NVF are trained in a multi-task learning framework by minimizing the loss function \mathcal{L} in Eq. (2). To confirm the efficacy of multi-task learning, we examine the fourth network architecture (IND; independent) that estimates note values and voice labels independently (Fig. 3(d)). IND consists of two BiLSTM networks and they are trained separately: by minimizing the loss functions \mathcal{L}_d and \mathcal{L}_v , respectively. In all the network architectures, the first fully connected layer outputs 25-dimensional vectors, and each BiLSTM layer outputs a 50-dimensional hidden vector at each time step.

3.4 Post-Processing Methods

The note values and voice labels $\{(\hat{d}_n, \hat{v}_n)\}_{n=1}^N$ estimated by the network are sometimes inconsistent with the musical convention. As general rules, notes with the same onset time and the same voice should have the same note values. Also, the offset times of those notes should not be larger

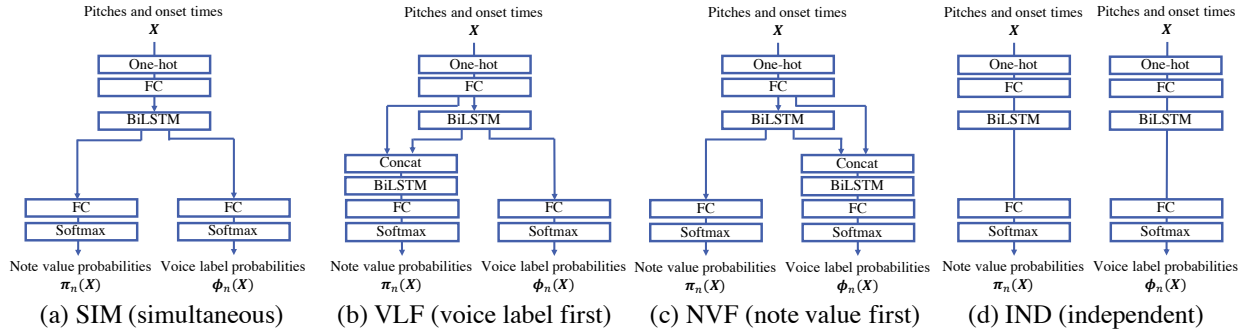


Figure 3. Proposed BiLSTM networks for estimating note values and voice labels. The four architectures are explained in Section 3.3.

than the next onset time in that voice. For two notes n and m in the same voice, these constraints are represented as follows:

$$o_n = o_m \implies d_n = d_m, \quad (7)$$

$$o_n < o_m \implies d_n \leq o_m - o_n. \quad (8)$$

To impose the constraints, we consider three possible post-processing methods to adjust the estimated note values $\{\hat{d}_n\}_{n=1}^N$. Let $\{n_k\}_{k=1}^K$ index a set of notes with the same onset time in the same voice. In the first method (PP1), the note values $\{\hat{d}_{n_k}\}_{k=1}^K$ are all set to the interval to the next onset time as in [13]. Note that in this method note values are determined by the estimated voice labels and the note value probabilities are not used. In the second method (PP2), the note values are modified to their maximum value as follows:

$$\hat{d}'_{n_k} = \max_{l=1, \dots, K} \hat{d}_{n_l}. \quad (9)$$

If the adjusted note values \hat{d}'_{n_k} are longer than the interval to the next onset time, they are set to this interval. In the third method (PP3), we calculate the note value with the maximum probability from the candidate note values that satisfy the constraints as follows:

$$\hat{d}'_{n_k} = \arg \max_{d: d \leq d'} \prod_{l=1}^K \pi_{n_l}(d; \mathbf{X}), \quad (10)$$

where d' indicates the interval to the next onset time.

3.5 Data Augmentation

In the situation under consideration, the pitches and onset times used as the input \mathbf{X} are estimated in advance by some pitch and rhythm transcription methods. As the result, the input contains tempo errors, extra notes, and missing notes. To make the networks robust to these errors, we can apply data augmentation methods that add tempo errors, extra notes, and missing notes to the original training data \mathcal{D} . Since rhythm transcription methods often produce half-tempo and double-tempo errors [20], we create a tempo-transformed dataset \mathcal{D}_t by halving or doubling the correct onset times and note values. Extra notes produced by multipitch detection methods often have a pitch shifted by an octave from a correct note. We thus create a dataset containing extra notes and missing notes \mathcal{D}_{em} by randomly deleting correct notes and adding notes whose

pitches differ from correct pitches by one octave. In addition, to increase the amount of the training data, we implement another data augmentation method. Assuming that transposed piano scores are also musically valid, we train the network using data obtained by transposing the original data by an interval of δ semitones ($\delta = -12, -11, \dots, 12$).

4. EVALUATION

We report experiments to evaluate the transcription accuracy of the proposed method.

4.1 Experimental Conditions

To evaluate the method in a practical condition, we incorporated it in an audio-to-score transcription system and generated transcriptions for test piano recordings. We first estimated pitches and quantized onset times by the state-of-the-art methods for multipitch detection and rhythm transcription used in the transcription system in [20]. For the results (called quantized MIDI data) we estimated note values and voice labels with the proposed method. We finally used public score editing software MuseScore 3 for score typesetting and generated transcriptions in the MusicXML format (Fig. 1). For comparison, we also generated transcriptions by existing methods [13, 20] using the same quantized MIDI data and with the same procedure for score typesetting. The CTD16 method [13] uses the Melisma Analyzer [4] for estimating note values and voice labels. The SNY21 method [20] is currently the best-performing system and uses a statistical model for note value estimation and a dynamic-programming method for voice separation.

As test data, we used 30 recordings of classical piano music in the MAPS-ENSTDkC1 dataset [25] and 81 piano cover recordings of popular music used in [20]. The ground-truth musical scores for these recordings were prepared in the MusicXML format and used for assessing the generated transcriptions. We used the musical scores of 80 classical music pieces and 763 popular music pieces for training the BiLSTM networks; the same training data were used in [20]. We applied the data augmentation in Section 3.5 to these training samples.

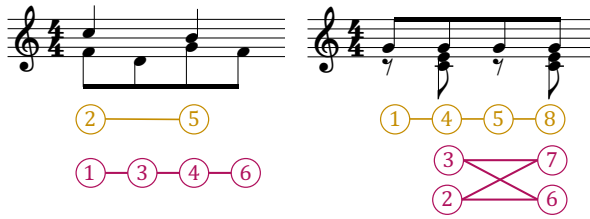


Figure 4. Voice structures represented by graphs.

4.2 Evaluation Metrics

We used the edit-distance-based error rates [22] for evaluating the quality of generated transcriptions. The error rates are the pitch error rate \mathcal{E}_p , the missing note rate \mathcal{E}_m , the extra note rate \mathcal{E}_e , the onset-time error rate \mathcal{E}_{on} , and the offset-time error rate \mathcal{E}_{off} . They are calculated after aligning an estimated score with a correct score. In particular, we can use the offset-time error rate \mathcal{E}_{off} as a metric for evaluating the accuracy of estimated note values. Since these metrics do not evaluate the accuracy of estimated voice labels, we consider the voice error rate \mathcal{E}_v defined as the proportion of notes with incorrect voice labels. The overall error rate \mathcal{E}_{all} is defined as the mean of these six error rates.

The edit-distance-based metrics have a clear interpretation: they count how many notes or score elements should be edited to obtain the correct score. This is an advantage over other metrics for music transcription [26,27]. We call the above defined metrics MUSTER (MUSIC Score Transcription Error Rates) and the evaluation tool is made available online².

We also used an F-measure [17] for assessing the quality of estimated voice labels; this metric is conventionally used in studies on voice separation. The original metric [17] is formulated for monophonic voices and we here extend it for homophonic voices. A voice structure can be represented by a graph, where notes of consecutive chords in a voice are connected by an edge (Fig. 4). The graph can be represented by an adjacency matrix (a_{ij}) , where $a_{ij} = 1$ when the i -th note is in a chord and the j -th note is in the next chord of the same voice, and otherwise $a_{ij} = 0$. We use the notation (a_{ij}) for a correct score and (\hat{a}_{ij}) for an estimated score. The precision \mathcal{P}_v , the recall \mathcal{R}_v , and the F-measure \mathcal{F}_v are defined as follows:

$$\mathcal{P}_v = \frac{\sum_{i < j} a_{ij} \hat{a}_{ij} / \hat{w}_i}{\sum_{i < j} \hat{a}_{ij} / \hat{w}_i}, \quad \mathcal{R}_v = \frac{\sum_{i < j} a_{ij} \hat{a}_{ij} / w_i}{\sum_{i < j} a_{ij} / w_i}, \quad (11)$$

$$\mathcal{F}_v = \frac{2\mathcal{P}_v\mathcal{R}_v}{\mathcal{P}_v + \mathcal{R}_v}. \quad (12)$$

Here, $\sum_{i < j}$ signifies a summation over all notes i and all notes j that appear after i , and we have defined the weight for each note i as

$$w_i = \sum_{j > i} a_{ij}, \quad \hat{w}_i = \sum_{j > i} \hat{a}_{ij} \quad (13)$$

in order to normalize the contribution of each chord no matter how many notes it contains.

Method	\mathcal{E}_{off}	\mathcal{E}_v	\mathcal{P}_v	\mathcal{R}_v	\mathcal{F}_v
SIM+DA	33.3	39.1	63.9	64.9	64.0
VLF+DA	32.2	39.0	65.2	65.7	65.1
NVF+DA	32.9	40.7	63.1	62.6	62.5
IND+DA	32.9	40.5	64.1	63.8	63.6
VLF	33.1	39.1	64.3	64.3	64.0

Table 1. Error rates (%) and accuracies (%) of estimated note values and voice labels for the MAPS dataset. DA indicates that each network is trained with augmented data.

Method	\mathcal{E}_{off}	\mathcal{E}_v	\mathcal{P}_v	\mathcal{R}_v	\mathcal{F}_v
SIM+DA	17.9	12.2	87.1	87.3	87.1
VLF+DA	17.2	11.4	87.7	87.7	87.6
NVF+DA	18.1	12.4	87.4	87.2	87.2
IND+DA	18.7	12.5	86.8	86.4	86.5
VLF	17.5	11.4	87.5	87.8	87.6

Table 2. Error rates (%) and accuracies (%) of estimated note values and voice labels for the J-pop dataset.

Method	MAPS	J-pop
VLF+DA	32.2	17.2
VLF+DA+PP1	28.0	15.3
VLF+DA+PP2	31.4	16.3
VLF+DA+PP3	32.2	16.8

Table 3. Error rates \mathcal{E}_{off} (%) of estimated note values with different post-processing methods.

4.3 Experimental Results

We first compare the four network architectures (SIM, VLF, NVF, and IND) with or without the application of data augmentation (DA). The evaluation results are listed in Tables 1 and 2 for the MAPS dataset and the J-pop dataset, respectively. Among the four architectures trained with data augmentation, VLF achieved the best accuracy in both note values and voice labels. The higher accuracy of VLF compared to NVF indicates that it is better to estimate voice labels first. A comparison between VLF and IND confirms the efficacy of multi-task learning. By comparing the results for VLF with and without data augmentation, we found a positive effect of data augmentation. Similar results were obtained for the other network architectures.

We next compare the three post-processing methods (Table 3). The first method (PP1) achieved the lowest error rates. The second one (PP2) slightly reduced the offset error rates for both datasets. Before and after the third method (PP3), the error rates were almost the same. In the first post-processing method, note values are calculated from estimated voice labels and note value probabilities estimated by the network are not used. Importantly, this does not mean that note value estimation was useless in the present method: estimating note values by the network was effective for improving the voice estimation through the multi-task learning, which in turn led to more accurate note value estimations.

The first method also has a limitation that it cannot estimate rests. Rests are used to express articulations and to make scores easier to read. An example of the tran-

² <https://amtevaluation.github.io/>

Method	Test	\mathcal{E}_p	\mathcal{E}_m	\mathcal{E}_e	\mathcal{E}_{on}	\mathcal{E}_{off}	\mathcal{E}_v	\mathcal{E}_{all}	\mathcal{P}_v	\mathcal{R}_v	\mathcal{F}_v
Proposed (VLF+DA+PP1)	MAPS	0.67	8.11	6.23	11.6	28.0	39.1	15.6	65.2	65.7	65.1
SNY21 [20]	MAPS	0.67	8.11	6.23	11.5	28.3	44.6	16.6	62.4	59.4	60.6
CTD16 [13]	MAPS	0.88	13.5	6.33	16.8	44.0	74.3	26.0	56.0	42.5	47.9
Proposed (VLF+DA+PP1)	J-pop	0.61	4.03	7.29	2.67	15.3	11.4	6.89	87.6	87.7	87.6
SNY21 [20]	J-pop	0.61	4.03	7.29	2.69	20.9	18.0	8.92	78.6	77.0	77.7
CTD16 [13]	J-pop	0.82	12.8	7.21	8.48	55.7	65.8	25.1	51.3	38.8	44.0

Table 4. Error rates (%) and accuracies (%) of transcription. The CTD16 method could output results for 27 (72) pieces in the MAPS (J-pop) dataset; the metrics are calculated from these pieces.

Figure 5. Example transcription results. The proposed method improved the accuracy of note values.

scription results is shown in Fig. 5, where the proposed method with the second post-processing method correctly estimated rests. In the future, it is important to estimate rests in order to improve the average accuracy of note values.

We finally compare the proposed method with existing transcription methods [13, 20]. The full set of MUSTER metrics and the voice F measure for the MAPS and J-pop datasets are listed in Table 4. The present method achieved the best accuracy for both datasets. The transcription accuracies for the MAPS dataset were lower than those for the J-pop dataset because the former has more complicated voice structures and there were a small number of classical music pieces in the training data.

To compare the performance of the voice estimation by our method with a recent method focusing on voice separation, we also evaluated the HMM-based voice separation method (MS16) [18]. Since this method requires as input pitches, onset times, and offset times, we used the note values estimated by the network IND. The F-measures \mathcal{F}_v of the voice separation results by MS16 were 55.7% and 66.5% for the MAPS dataset and the J-pop dataset, respectively. It is confirmed that the proposed method significantly outperformed the MS16 method.

An example of the transcription results is shown in Fig. 6, for the proposed method and the SNY21 method. The proposed method estimated voice labels close to the ground truth, and made the piano score easier to read than the one estimated by the SNY21 method. Other examples are shown on the supplemental web page³.

³ <https://nvvest.github.io>

Figure 6. Example transcription results. The proposed method improved the accuracy of voice labels and improved the readability.

5. CONCLUSION

This paper presented a neural method that jointly estimates note values and voice labels from transcribed pitches and onset times. Since note values and voices are interrelated, we constructed a BiLSTM network in a multi-task learning framework. We demonstrated through experiments that the proposed method achieved the state-of-the-art performance of audio-to-score APT when combined with the latest methods for multipitch detection and onset time quantization.

The error rates of note values and voice labels are still high compared to the other metrics. In future work, we plan to further investigate the data representation and network architecture to increase the consistency between estimated note values and voice labels. To correctly estimate rests and improve the readability of transcribed scores, we will develop a more sophisticated post-processing method and study the effective use of performed durations.

Although we focused on the estimation of note values and voices in this study, we found that the result is affected by errors made by the onset time quantization method. It is thus important to develop a method that integrates onset time quantization. As the fully end-to-end approaches still have difficulties in practical applications [23, 24], it is also considered effective to unify the multiple stages in Fig. 1 one step after another.

6. ACKNOWLEDGEMENT

This work is supported in part by JST PRESTO No. JP-MJPR20CB, JSPS KAKENHI Nos. 19H04137, 19K20340, 20K21813, and 21K12187, and ISHIZUE 2021 of Kyoto University Research Development Program.

7. REFERENCES

- [1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.
- [2] P. Desain and H. Honing, "The quantization of musical time: A connectionist approach," *Computer Music Journal*, vol. 13, no. 3, pp. 56–66, 1989.
- [3] C. Raphael, "A hybrid graphical model for rhythmic parsing," *Artificial Intelligence*, vol. 137, pp. 217–238, 2002.
- [4] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, 2009.
- [5] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE TASLP*, vol. 18, no. 3, pp. 528–537, 2010.
- [6] E. Benetos and T. Weyde, "An efficient temporally-constrained probabilistic model for multiple-instrument music transcription," in *ISMIR*, 2015, pp. 701–707.
- [7] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM TASLP*, vol. 24, no. 5, pp. 927–939, 2016.
- [8] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *ISMIR*, 2018, pp. 50–57.
- [9] Y.-T. Wu, B. Chen, and L. Su, "Polyphonic music transcription with semantic segmentation," in *ICASSP*, 2019, pp. 166–170.
- [10] Q. Kong, B. Li, J. Chen, and Y. Wang, "GiantMIDI-Piano: A large-scale MIDI dataset for classical piano music," *arXiv preprint arXiv:2010.07061*, 2020.
- [11] E. Nakamura, K. Yoshii, and S. Sagayama, "Rhythm transcription of polyphonic piano music based on merged-output HMM for multiple voices," *IEEE/ACM TASLP*, vol. 25, no. 4, pp. 794–806, 2017.
- [12] E. Nakamura and K. Yoshii, "Music transcription based on Bayesian piece-specific score models capturing repetitions," *arXiv preprint arXiv:1908.06969*, 2019.
- [13] A. Cogliati, D. Temperley, and Z. Duan, "Transcribing human piano performances into music notation." in *ISMIR*, 2016, pp. 758–764.
- [14] J. Kilian and H. H. Hoos, "Voice separation-A local optimization approach," in *ISMIR*, 2002, pp. 39–46.
- [15] I. Karydis, A. Nanopoulos, A. Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos, "Horizontal and vertical integration/segregation in auditory streaming: A voice separation algorithm for symbolic musical data," in *Proc. of Sound and Music Computing Conference*, 2007, pp. 299–306.
- [16] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, pp. 75–94, 2008.
- [17] B. Duane and B. Pardo, "Streaming from MIDI using constraint satisfaction optimization and sequence alignment." in *Proc. of International Computer Music Conference*, 2009.
- [18] A. McLeod and M. Steedman, "HMM-based voice separation of MIDI performance," *Journal of New Music Research*, vol. 45, no. 1, pp. 17–26, 2016.
- [19] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *ISMIR*, 2018.
- [20] K. Shibata, E. Nakamura, and K. Yoshii, "Non-local musical statistics as guides for audio-to-score piano transcription," *Information Sciences*, vol. 566, pp. 262–280, 2021.
- [21] E. Nakamura, K. Yoshii, and S. Dixon, "Note value recognition for piano transcription using Markov random fields," *IEEE/ACM TASLP*, vol. 25, no. 9, pp. 1846–1858, 2017.
- [22] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon, "Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization," in *ICASSP*, 2018, pp. 101–105.
- [23] R. G. C. Carvalho and P. Smaragdis, "Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score," in *Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2017, pp. 151–155.
- [24] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "A holistic approach to polyphonic music transcription with neural networks," in *ISMIR*, 2019, pp. 731–737.
- [25] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2009.
- [26] A. Cogliati and Z. Duan, "A metric for music notation transcription accuracy." in *ISMIR*, 2017, pp. 407–413.
- [27] A. McLeod and M. Steedman, "Evaluating automatic polyphonic music transcription." in *ISMIR*, 2018, pp. 42–49.

LEARNING NOTE-TO-NOTE AFFINITY FOR VOICE SEGREGATION AND MELODY LINE IDENTIFICATION OF SYMBOLIC MUSIC DATA

Yo-Wei Hsiao Li Su

Institute of Information Science, Academia Sinica, Taiwan

{willyhsiao, lisu}@iis.sinica.edu.tw

ABSTRACT

Voice segregation, melody line identification and other tasks of identifying the horizontal elements of music have been developed independently, although their purposes are similar. In this paper, we propose a unified framework to solve the voice segregation and melody line identification tasks of symbolic music data. To achieve this, a neural network model is trained to learn note-to-note affinity values directly from their contextual notes, in order to represent a music piece as a weighted undirected graph, with the affinity values being the edge weights. Individual voices or streams are then obtained with spectral clustering over the learned graph. Conditioned on minimal prior knowledge, the framework can achieve state-of-the-art performance on both tasks, and further demonstrates strong advantages on simulated real-world symbolic music data with missing notes and asynchronous chord notes.

1. INTRODUCTION

Identifying the horizontal elements of music (e.g., melody, accompaniment, voice, stream, and counterpoint) is crucial for understanding musical data. As a mandatory step in music transcription [1] and generation [2, 3], this problem has been widely discussed; related tasks include melody extraction [4] and multi-pitch streaming [5, 6] for audio music data and voice segregation [7] and melody line identification [8] for symbolic music data. In this paper, we will focus on the case of symbolic music data.

It should be noted that the afore-mentioned tasks are rarely considered under a unified framework, but are solved individually according to the music texture of the input music. For example, voice segregation is only for *polyphony*, the texture with multiple independent melody lines; while melody line identification is only for *homophony*, the texture with one predominant melody line plus an accompaniment. The input music piece with hybrid or unknown texture cannot be discussed under these frameworks. Besides, most of these frameworks still heavily rely on further information of the input (e.g., number of

voices, whether each voice is monophonic, whether chord notes are perfectly synchronized) and strictly follow some predefined rules (e.g., avoiding voice crossing), and turn out to be inflexible to real-world performance data with missing notes or asynchronous chord notes.

In this paper, we propose a unified horizontal element extraction framework which works with minimal constraints on musical textures and pre-defined rules. The major idea is to let the model learn the configuration of voice directly from the training data, and learn the *note-to-note affinity* for arbitrary pairs of notes within a musical segment from their shared contextual notes: the model outputs 1 if a pair of notes are in the same horizontal elements, while 0 if they are not. Then, based on the learned affinity values, a clustering algorithm is used to estimate the number of horizontal elements, and to partition the notes which are linked with high affinity values into one element. Finally, these elements extracted from different musical segment can be merged without any perceptual or musical assumptions by applying the *minimal overlapping* principle proposed in this paper.

To verify our ideas, the same framework is applied on multiple tasks with various conditions, including 1) the polyphony voice segregation task with unknown number of voices, missing notes and asynchronous chord notes for simulating real-world performance, and 2) the melody line identification task of homophonic music. The framework achieves state-of-the-art performance on both tasks and shows strong advantages on simulated real-world cases. Furthermore, we demonstrate the potential of using the graph constructed with the learned note-to-note affinity as a tool in computational analysis of general music data.

2. RELATED WORK

2.1 Voices, streams, and their perceptual rules

A *voice* or a *stream* is a horizontal music structure which is *perceived* as single sonority by humans. A voice is a sequence of monophonic and non-overlapped musical tones in polyphony texture, such as the S, A, T, and B in a 4-part chorale. On the other hand, a stream can be either a monophonic voice or a multi-tone sonority fused by several musical lines, such as the predominant *melody line* and *accompaniment* in homophonic texture. A monophonic note sequence may also contain multiple voices. The perception of voice or stream in music is highly subjective. The voice analysis for the very same music piece might end up



with diverse results [9]. With abuse of terminology, the terms of voice, stream, and horizontal element are used interchangeably in the paper.

A number of perceptual rules have been proposed for voice segregation and melody identification tasks [10]. The *pitch proximity* and *temporal continuity* rules suggest that the temporal and pitch distance between two neighboring notes in a voice should be minimized, and large leaps or rests should be avoided [11]. The *new stream* rule suggests that the number of streams in a music piece should be minimized. The *voice collision* rule states that common tones shared between different voices should be avoided. The *voice crossing* suggests that two voices do not cross each other even when their pitch ranges overlap significantly.

2.2 Prior art

Voice segregation and melody line identification methods can be categorized into three classes. First, the *rule-based* methods, such as local optimization [12], contig mapping [13–16], graph-based method [17], complexity-based method [18], and the voice integration and segregation algorithm (VISA) [19] utilize heuristics or perceptual principles as constraints for tracing voices. These methods do not strictly designate the role of each stream (e.g., one stream should be melody and the other should be accompaniment), and therefore can be applied to the data having arbitrary configurations of streams without labels [20]. The major limitation of these methods is that they are less flexible dealing with real-world performance data.

Second, the *data-driven* methods introduce either classifiers to predict the voice or stream labels of each note from annotated data [7, 21–23], or regression models to predict the ratings over all the mappings from notes to voices for each chord [24, 25]. Representative examples include the convolutional neural network (CNN) model which predict the position of melody notes on a piano roll [26], or a feedforward neural network to classify voice indices from note-level features [27]. Different from the rule-based methods, data-driven methods are based on supervised learning. Therefore, the output dimension of the model is usually restricted by the label classes in the training data. This issue can be solved with neural greedy search such as [28], which implicitly indicated the importance of learning note-to-note affinity.

Besides the rule-based and data-driven approaches, most of the methods are *hybrid* ones which incorporate both perceptual rules and supervised learning in voice segregation and melody line identification. For example, in the hidden Markov model (HMM)-based voice segregation method, the probability of note transition is defined according to the perceptual principles, while the pitch score and gap score in probability function can be tuned to fit the training data [29]. In the classification-based methods, hand-crafted input features which consider the perceptual principles have been proposed in various ways [21, 24, 27]. Some of these features can be used only when the number of voice, the metric positions and other note attributes of the input music are known.

3. PROPOSED METHOD

We represent a symbolic music piece as a undirected weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each vertex $v_i \in \mathcal{V}$ represents a note in MIDI, and each weighted edge $w_{ij} \in \mathcal{E}$ corresponds to the affinity between two notes v_i and v_j . We assume $w_{ij} = 1$ if v_i and v_j are in the same voice or melodic line, while $w_{ij} = 0$ if they are situated in different parts. The task of voice segregation is then equivalent to the task of learning w_{ij} from the training data having the binary-valued w_{ij} as the ground truth label.

Denote the affinity matrix of \mathcal{G} as W , and $|\mathcal{V}|$ the vertex count of \mathcal{G} . Then, we have $W \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, and w_{ij} is the (i, j) th element of W . To learn W , we adopt a CNN model which takes the information carried by a pair of notes (v_i, v_j) as inputs, and outputs an affinity value $\hat{w}_{ij} \in [0, 1]$ which minimizes the binary cross-entropy (BCE) between \hat{w}_{ij} and w_{ij} . Once the predicted affinity matrix \hat{W} is obtained, the task of voice segregation is simplified into a clustering problem. With the help of spectral clustering algorithm, \mathcal{G} is partitioned into multiple subgraphs, each of which represents a voice.

3.1 Data representation

For simplicity, the vertex v_i directly represents the content of the i th note: each note event $v_i := [p_i, o_i, d_i]^T$ is a 3-dimensional vector composed of its pitch (in MIDI number), onset time, and duration (both in second); see Figures 1a and 1b. The order index i of each note is obtained by sorting all the note events with the following rules: 1) notes are sorted by its onset time in ascending order; 2) if multiple notes have the same onset time, they are then sorted by their pitch in ascending order; and 3) if multiple notes happen to have identical onset time and pitch value, they are sorted by their duration in descending order. The adopted data representation of each v_i , denoted as x_i , is simply constructed by v_i and its neighbouring notes. More specifically, x_i is defined as

$$x_i := [\bar{v}_{i-M,i}, \bar{v}_{i-M+1,i}, \dots, \bar{v}_{i+M-1,i}, \bar{v}_{i+M,i}]^T, \quad (1)$$

and we have $x_i \in \mathbb{R}^{(2M+1) \times 3}$. It should be noted that $\bar{v}_{j,i}$ is the content of the j th note event with its onset time expressed relative to the i th note, i.e. $\bar{v}_{j,i} := [p_j, o_j - o_i, d_j]$. This operation makes the onset time information in each x_i be centered at the same temporal position. For $\bar{v}_{j,i}$ with $j \leq 0$ or $j > |\mathcal{V}|$, the note sequence is zero-padded (i.e., add virtual notes with its MIDI pitch, onset time and duration being all zeros) such that the $(M + 1)$ th row of x_i is $\bar{v}_{i,i}$, as shown in Figure 1c.

The idea behind the above data representation is that it simulates human behavior. When given tasks like voice segregation, humans do not plainly judge the affinity of a pair of notes merely by their own pitch and position, but by the local musical context lying in the structure. The hyperparameter M determines the context window, and we set $M = 60$ notes in this paper.

In the setup of affinity learning, the training data is then the pairs of the data representation given a binary label



(a) The score

$$\begin{array}{l}
 v_1 = (64, 0, 0.5) \\
 v_2 = (74, 0.25, 0.25) \\
 v_3 = (65, 0.5, 0.5) \\
 v_4 = (76, 0.5, 0.25) \\
 \dots
 \end{array}
 \quad
 x_2 = \begin{bmatrix}
 0 & 0 & 0 \\
 64 & -0.25 & 0.5 \\
 74 & 0 & 0.25 \\
 65 & 0.25 & 0.5 \\
 76 & 0.25 & 0.25
 \end{bmatrix}$$

(b) Note events

(c) The matrix

Figure 1: An example of data representation for $M = 2$.

$w_{ij} \in \{0, 1\}$. A training sample (x_i, x_j) is labeled as $w_{ij} = 1$ if v_i and v_j are in the same voice, whereas (x_i, x_j) is labeled as $w_{ij} = 0$ if they are not.

3.2 Model training

We employ a multi-layer deep 1-D CNN $f(X)$ to classify whether x_i and x_j are in the same musical stream. More specifically, given $X_{ij} \in \mathbb{R}^{(2M+1) \times 6}$ the concatenation of two matrices x_i and x_j , we have $\hat{w}_{ij} = f(X_{ij})$ so as to minimize $\text{BCE}(w_{ij}, \hat{w}_{ij})$. The architecture of the CNN contains six 1D convolution layers, with the kernel size of each layer being [32, 16, 16, 8, 8, 4], and the number of each kernel being [32, 32, 64, 64, 128, 128]. The output of the final convolution layer is flattened and mapped to an output logit using a fully connected layer.

For a music piece with $|\mathcal{V}|$ notes, there are totally $|\mathcal{V}|(|\mathcal{V}| - 1)/2$ pairs of notes that can be used for training. However, taking all the pairs for training is computationally intensive, and unnecessary from the perspective of music perception. When people listen to music, it is impractical to have them distinguish whether a pair of notes several bars apart belong to the same voice. Instead, listeners tend to focus on a restricted time interval and identify a voice from others according to the relationship among the notes in the interval. Therefore, we choose all of the pairs (x_i, x_j) with $|i - j| \leq N$ for training, where the hyperparameter N represents the maximum distance of two notes. In this paper, we set $N = 30$ notes according to our study.

3.3 Voice extraction with graph clustering

We employ spectral clustering [30, 31], one of the most widely used graph-based clustering techniques, to separate the voices or streams according to the learned affinity matrix $\hat{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ for the music data graph \mathcal{G} . However, it should be noted that the model $f(X_{ij})$ outputs \hat{w}_{ij} (i.e. the estimated value between v_i and v_j) only for $|i - j| \leq N$; other \hat{w}_{ij} are still unknown. Therefore, the following assignment processes are adopted to adjust the affinity ma-

trix:

$$\begin{aligned}
 \hat{w}_{ij} &:= 1 && \text{if } \hat{w}_{ij} > 0.5 \\
 \hat{w}_{ij} &:= \epsilon_1 && \text{if } \hat{w}_{ij} \leq 0.5 \\
 \hat{w}_{ij} &:= \epsilon_2 && \text{if } \hat{w}_{ij} \text{ is unknown}
 \end{aligned}
 \quad (2)$$

where $:=$ is the assignment operator. We assume $\epsilon_1 < \epsilon_2 < 1$ in order to represent the uncertainty of the affinity for distant note pairs (i.e. with $|i - j| > N$). Therefore, in this paper we set $\epsilon_1 = 10^{-6}$ and $\epsilon_2 = 10^{-3}$. Our study showed that these discretized values give stable eigendecomposition in the spectral clustering process and perform better than directly using predicted values.

It should be noted that performing spectral clustering over the whole music piece is unfeasible, as the eigendecomposition of a large affinity matrix tends to be highly unstable, an unfavorable effect in spectral clustering. To address this issue, we divide a musical piece into multiple overlapping segments and perform spectral clustering for each segment. Each of the segments contains S consecutive note events and each pair of consecutive segments are overlapped by O note events, where $0 \leq O < S$.

For \hat{W}_i , which denotes the affinity matrix of the i th segment, the normalized graph Laplacian L_i for spectral clustering is represented as

$$L_i = I - D^{-\frac{1}{2}} \hat{W}_i D^{\frac{1}{2}}, \quad (3)$$

where I is the identity matrix, and the degree matrix D is a diagonal matrix, whose i th diagonal element $d_{ii} := \sum_j \hat{w}_{ij}$ is the sum of the affinity measure between v_j and v_i for all v_j which are connected with v_i . Given the Laplacian L , a matrix $Z = [z_1, z_2, \dots, z_k] \in \mathbb{R}^{n \times k}$ is formed by stacking the top- k largest eigenvectors z_i of L , and the k subsets is obtained by applying the k -means algorithm to the rows of Z [31].

3.4 Estimating the number of clusters

The number of clusters, or the number of estimated voices or streams, k , is a pre-determined parameter in the spectral clustering process. For the melody line identification task, k is simply 2. For the voice segregation task, we assume that k is unknown and needs to be estimated. An intuitive estimation is to set k as the maximal number of synchronous note events occurring in a piece [13]. However, this method may not be suitable for voices or streams having overlapped notes, such as human-performed MIDI or homophonic music data.

To address the issue, the *eigengap* of Laplacian L is employed to predict k of each segment. Let $\{\lambda_i\}_{i=1}^N$ be the eigenvalues of L sorted in descending order. According to graph theories, the number of clusters k of the graph can be estimated by the k th eigenvalue having the most significant eigengap of L [32]. To implement this, we calculate the top-10 largest eigenvalues of each L_i , and find the index k_i satisfying that the difference between λ_{k_i} and λ_{k_i+1} is the largest one. To simplify our discussion, we assume that the number of voices of the whole music piece is a constant, meaning that each voice in a piece contributes at least one note in any segment of the piece. The number of voices, k , is therefore obtained by the mode of k_i over all segments.

3.5 Segment merging

After we apply spectral clustering to \hat{W}_i , note events in the i th segment are partitioned into k segregated voices (or streams). Let $\mathcal{C}_i = \{\mathcal{S}_{ij}\}_{j=1}^k$ denote the i th segment with voice labeling, where \mathcal{S}_{ij} denotes the j th voice obtained from the spectral clustering result of the i th segment. We consider two methods to merge the segmented voices into complete ones. The first method, called the *pitch proximity* method, is performed straightforwardly by sorting the segmented voices \mathcal{S}_{ij} according to their average pitch for each i , and connecting the segmented voices with the same sorting index. Being stable and perceptually plausible, the pitch proximity method however assumes the prior knowledge of part-crossing rule (i.e. voices tend not to cross with respect to pitch) [11], which is no longer valid in the situations such as voice crossing nearby the end of a segment.

The second method, coined as the *minimal overlapping* method, is a newly proposed method which does not rely on any perceptual rules. In a nutshell, this method attempts to find a one-to-one mapping for voices from two adjacent segments ($\mathcal{C}_i, \mathcal{C}_{i+1}$) to attain a newly merged segment $\mathcal{C}_* = \{\mathcal{S}_{*j}\}_{j=1}^k$ such that

- a voice in the new segment \mathcal{S}_{*j} is merged from two voices, \mathcal{S}_{im} and $\mathcal{S}_{(i+1)n}$, $1 \leq m, n \leq k$, if and only if they are overlapped (connectivity condition);
- each note in either \mathcal{C}_i or \mathcal{C}_{i+1} should be in \mathcal{S}_{*j} for some j and therefore in \mathcal{C}_* (consistency condition);
- the total number of notes which belong to multiple merged voices of $\{\mathcal{S}_{*j}\}_{j=1}^k$ should be minimized (minimal redundancy condition).

These conditions are implemented with the following procedures. First, we list all the possible merged voices satisfying the connectivity condition, denoted as $\Sigma_1 := \{\mathcal{S}_{ij} \cup \mathcal{S}_{(i+1)l} \mid \mathcal{S}_{ij} \cap \mathcal{S}_{(i+1)l} \neq \emptyset\}$. A merged segment \mathcal{C}_* with these merged voices is constructed by choosing k elements from Σ_1 . Denote all candidates of such k chosen elements as $\binom{\Sigma_1}{k}$. Given a fixed-valued k and the consistency condition, we narrow down the set of candidates to $\Sigma_2 := \{\binom{\Sigma_1}{k} \mid \{\bigcup_{j=1}^k \mathcal{S}_{*j}\} = \mathcal{C}_i \cup \mathcal{C}_{i+1}\}$. To apply the minimal redundancy condition, we begin with calculating the number of notes existing in multiple voices (#NMV) for an arbitrary merged segment \mathcal{C}'_* , which is defined as

$$\text{\#NMV} := \sum_{j=i+1}^k \sum_{i=1}^{k-1} |\mathcal{S}'_{*i} \cap \mathcal{S}'_{*j}|, \quad (4)$$

where $\mathcal{S}'_{*i}, \mathcal{S}'_{*j} \in \mathcal{C}'_*$. Then, we pick the segment \mathcal{C}_* with the smallest #NMV among all possible merged segments in Σ_2 to be our final choice. Finally, because we require that a note should be classified to one voice strictly in our study, we remove notes existing in multiple voices from a random voice to ensure \mathcal{C}_* behaves like an ideal segment.

Once a larger segment is obtained, it works like a crystal nucleus. It will continue growing its size by merging with another adjacent segment when we feed them into the algorithm. In the end, there would be only one merged section left, which is then the final result. We set the size of

the segment to $S = 40$ notes. The overlap size is set to $O = 0$ for the pitch proximity method, and $O = 30$ notes for the minimal overlapping method.

4. DATASETS

Three major datasets are used to evaluate our method. For voice segregation, we use the Bach Chorales Dataset (BCD) collected from IMSLP.org, which originally contains 364 four-voice chorales composed by Johann Sebastian Bach. To test the robustness of the models, the dataset is augmented with voice dropping, note dropping, and onset/offset shifting. As a result, the dataset includes the original four-voice pieces, 2,184 two-voice pieces, 1,456 three-voice pieces, and 364 four-voice pieces with onset/offset shifting, all of which are augmented with three different note dropping rates (see the data augmentation process described as below). For melody extraction, we use two datasets, Mozart Piano Sonatas (MPS) [26] and Americans Folks (AF) [8]. MPS consists of 38 movements from Mozart’s piano sonatas, and their melody lines were annotated by a professional pianist. AF is the subset of “Big Dataset,”¹ and contains 1,262 folk songs in MIDI format. Every folk song contains a Soprano track, which is picked as the melody line. For AF, we crop the piece so that the melody line consistently exists among note events, and merge all the other note events that do not belong to the melody into a single accompaniment voice.

To further enhance the generalization ability of the model, the following data augmentation techniques are applied for the training data:

1. Tempo scaling: the tempo of each piece is by 2^i times faster (or slower), where i is uniformly sampled from the interval $[-1, 1]$.
2. Key shifting: each piece is transposed by n semitones, for n being uniformly sampled from $\{-6, -5, \dots, 5, 6\}$.
3. Note dropping: a piece has an equal chance for dropping 0%, 5%, or 10% of its notes.
4. Voice dropping: the voices in a Bach 4-part chorales are randomly dropped with rates $p(c)$, $c \in \{0, 1, 2\}$, where $p(c)$ donates the probability of dropping c voices. In our setting, we set $p(0) = 0.6$, $p(1) = 0.3$, and $p(2) = 0.1$. Note that for the melody line identification task, voice dropping is not used because there are only two voices (i.e. melody and accompaniment) in the training data and no more voices can be further dropped.
5. Onset and offset shifting: the onset and offset time of a note event are stretched or shrunk by $(1+r)$ times of its duration, respectively; r is a sample from the truncated normal distribution [33], whose PDF is set to $f(r; \mu = 0, \sigma = 0.15, a = -0.15, b = 0.15)$; see [33] for detailed implementation.

¹ https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet/

Voices	Original (4-voice)			2-voice			3-voice			Onset-offset		
	0%	5%	10%	0%	5%	10%	0%	5%	10%	0%	5%	10%
Skyline	95.76	88.63	82.23	98.87	96.32	93.71	97.45	92.25	87.62	26.26	26.82	27.49
VoSA	97.03	95.42	93.77	99.06	98.67	98.29	98.08	97.22	96.27	-	-	-
HMM	97.79	96.10	93.50	99.28	98.99	98.44	98.59	97.75	96.29	67.43	68.95	67.75
Ours (Min)	97.40	96.33	95.04	99.03	98.74	98.44	98.28	97.64	97.02	95.56	94.44	92.70
Ours (Pitch)	97.43	96.26	94.99	99.03	98.75	98.46	98.30	97.66	97.03	96.62	94.43	92.84

Table 1: Frame-level accuracy for voice segregation (in %) on BCD.

Data	Model	P	R	F1
MPS	Skyline	88.49	93.91	91.09
	CNN	93.00	89.69	91.22
	Ours (Min)	91.30	92.04	91.60
	Ours (Pitch)	95.80	95.53	95.64
AF	Skyline	73.33	74.40	73.74
	CNN	69.00	89.61	77.27
	Ours (Min)	78.19	82.10	79.09
	Ours (Pitch)	85.10	85.04	84.77

Table 2: Results (in %) for melody identification

Finally, to facilitate the experiment process, we assume that one note belongs to only one voice. In the voice segregation task, if there exist identical note events in different voices, we assign it to one voice randomly and remove others. As for the the same situation in the melody line identification task, the note is assigned to the melody part.

5. EVALUATION AND DISCUSSION

The proposed neural networks are implemented with Tensorflow 2.0. We adopt Adam optimization [34] and the learning rate is set to 10^{-3} . All experiments are run with one NVIDIA GTX1060 GPU. The training time for every 1M pairs of notes is approximately one minute. Source codes are available at our website ².

Four baseline methods are considered. For voice segregation, we consider the skyline algorithm [26, 28], the VoSA algorithm [13] (which is re-implemented by ourselves), and the HMM-based voice segregation methods [29]. For melody line identification, the skyline algorithm and the state-of-the-art CNN-based melody extractor [26] are considered. These methods are compared with our proposed methods with two segment merging modes, which are denoted by Pitch (the pitch proximity method) and Min (the minimal overlapping method).

Several metrics are reported in this section. For the performance of neural networks, the first evaluation metric we consider is simply the *pairwise accuracy*, the accuracy of the binary prediction (i.e. voice connection of note pairs) of our 1D-CNN model. For voice segregation, we use frame-level accuracy, the ratio of correctly predicted frames to the total frames, to present the results. For melody line identification, the frame-level precision (P), recall (R), and F1-score (F1) are used [26].

² <https://github.com/Willy07/musical-stream-segregation>

5.1 Results

To evaluate the performance of our system, we performed 9-fold cross-validation on BCD under all the conditions mentioned in Section 4. We split each fold on MIDI files to ensure that all the note pairs from the same music piece are in the same fold. All the augmentation methods were also applied to the training data. The training and validation pairwise-accuracy of the 1D-CNN are 95.41% and 96.81%, respectively. Among all the validation data (including augmented data), our system can correctly predict the cluster number k over 99.8% (12865/12888) by eigen-gap. The result of voice segregation is shown in Table 1. First, for zero note drop and zero onset/offset shifting, HMM remains as the most superior method. However, the proposed methods prevails as the note dropping rate increases; for example, Ours (Min) outperforms HMM by 1.54 percentage points in 4-voice and 10% note dropping rate. In addition, in the case of onset-offset shifting, the proposed systems outperform all the others by at least 25 percentage points. These findings highlight the advantage of the proposed model on high tolerance to noisy data. Furthermore, the differences of performance between Ours (Min) and Ours (Pitch) are very small (within 0.1% for most of the cases), suggesting that the proposed method can work without imposing perceptual rules.

Similarly, in the task of melody line identification, we also performed 9-fold cross-validation on MPS and AF dataset, respectively. Only the first three data augmentation methods in Section 4 were applied in training. The training and validation pairwise accuracy values are 98.28% and 93.96% for MPS, and 91.87% and 88.43% for AF, respectively. From Table 2, we observe that our method with minimal overlapping is on par with the CNN baseline [26] on both MPS and AF. ³ When pitch proximity is applied, the proposed method outperforms others by at least 4 percentage points in F1-score.

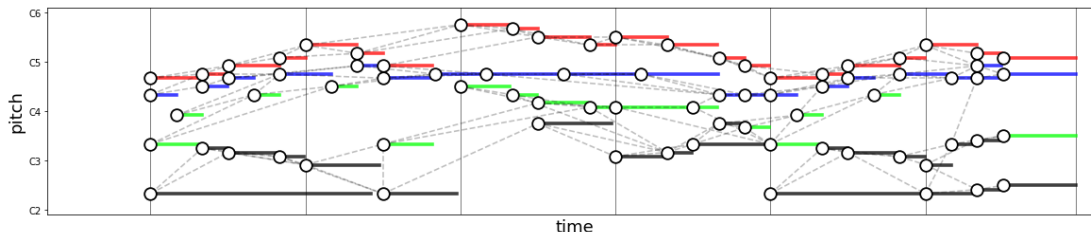
5.2 Discussion

To investigate the behaviors of our proposed framework, we conduct voice segregation and melody line identification for *unseen* types of data using the models we trained. Figure 2 takes the first six bars of the first movement of Beethoven’s Sonata No. 28 as an example. Figure 2b and 2c show the graphs constructed with the note-to-note affinity inferred by the BCD and AF models, respectively. Both

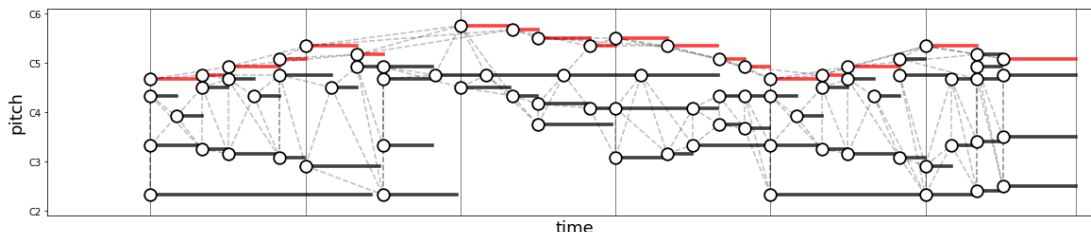
³ The CNN-based model for evaluating MPS is directly provided from [26]. We retrained a model using the source code to evaluate AF.



(a) The sheet music



(b) The result of voice segregation by the model trained in BCD



(c) The result of melody extraction predicted by the model trained in AF

Figure 2: Results on the first 6 bars of Beethoven Sonata No. 28 Mov. 1. Dashed lines in the piano rolls represent the edges with affinities $\hat{w}_{ij} > 0.5$ of the learned graph. The links between distant notes are omitted for better visualization.

graphs exhibit the “interpretation” from the two models. For the BCD model, notes tend to be linked in the horizontal direction and long-distance links are enforced to construct the voices. By contrast, the AF model prefers to establish vertical links in lower pitches (e.g., accompaniment), while enforcing horizontal links in higher pitches (e.g., melody). It is intriguing to see in Figure 2c that although the note A5 in the third bar is linked with several accompaniment notes, A5 is still classified as melody because of the even more abundant links to melody notes.

According to the directions of note stems, four voices can be identified from Figure 2a. The eigengap of the graph generated by the BCD model does give an estimation of $k = 4$, which is consistent with such a common interpretation. By comparing Figures 2a and 2b, most interpretation of the BCD model are consistent with the score sheet except some interesting exceptions. For example, in the first bar, the BCD model assigns B3 and E4’ of the second voice (denoted as E4–B3–F#4–G#4–E4’–A4) to the third voice, while D#3, D#3, and C#3 in the third voice (E3–D#3–D#3–C#3) are assigned to the fourth voice. That means, the model treats the second voice as *pseudopolyphony* and manages to derive two valid voices from it. An explanation of this phenomenon is that the BCD model learns the principles of counterpoint and tends to have more large leaps for lower voices. It can be found that the notes in the fourth voice from the BCD model are highly overlapped, although perceiving them as a voice is

indeed possible for human, if the duration information is ignored. The characteristics of the training data may also provide another explanation of this phenomenon; the density of notes in the fourth voice of this excerpt is sparser than the density of bass notes in BCD. Therefore, the model tends to build more links between the lowest note E2 and its contextual notes, and merges all of them into the same stream.

6. CONCLUSION

In this paper, we have presented a new, generalizable, and straightforward framework to learn note-to-note affinity for symbolic music data. The framework, taking only note attributes as training features, can outperform several state-of-the-art horizontal element extraction methods in various musical textures and in real-world scenarios. The graph representation induced from the framework can also serve as a tool for in-depth music analysis on the relationship among the notes if the label configuration describing the training data of interest is given. The major limitation of this work is that it has not considered the case of varying number of voices (e.g., voices shrinking or expansion) in the music data, which can however be well solved with a modified segment merging process and more accurate prediction on the eigengaps within segments. Adopting more advanced neural networks and statistical methods to achieve this goal is left as our future work.

7. REFERENCES

- [1] Y.-T. Wu, B. Chen, and L. Su, “Multi-instrument automatic music transcription with self-attention-based instance segmentation,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, pp. 2796–2809, 2020.
- [2] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Proc. of the AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018, pp. 34–41.
- [3] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, “Pypianoroll: Open source python package for handling multitrack pianoroll,” *Late-breaking/Demo paper in International Society Music Information Retrieval Conference (ISMIR)*, 2018.
- [4] D. Basaran, S. Essid, and G. Peeters, “Main melody extraction with source-filter NMF and CRNN,” in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 82–89.
- [5] Z. Duan, J. Han, and B. Pardo, “Multi-pitch Streaming of Harmonic Sound Mixtures,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 22, no. 1, pp. 138–150, 2014.
- [6] C.-Y. Kuan, L. Su, Y.-H. Chin, and J.-C. Wang, “Multi-pitch streaming of interwoven streams,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, USA, 2017, pp. 311–315.
- [7] A. Jordanous, “Voice separation in polyphonic music: A data-driven approach,” in *Proc. of the 2008 International Computer Music Conference (ICMC)*, Belfast, Ireland, 2008.
- [8] W.-T. Lu and L. Su, “Deep learning models for melody perception: An investigation on symbolic music data,” in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Honolulu, USA: IEEE, 2018, pp. 1620–1625.
- [9] E. Cambouropoulos, “Voice and stream: Perceptual and computational modeling of voice separation,” *Music Perception*, vol. 26, no. 1, pp. 75–94, 2008.
- [10] D. Deutsch, “Grouping mechanisms in music,” in *The Psychology of Music*. Elsevier, 2013, pp. 183–248.
- [11] D. Huron, “Tone and voice: A derivation of the rules of voice-leading from perceptual principles,” *Music Perception*, vol. 19, no. 1, pp. 1–64, 2001.
- [12] J. Kilian and H. H. Hoos, “Voice separation—a local optimization approach,” in *Proc. 3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, 2002.
- [13] E. Chew and X. Wu, “Separating voices in polyphonic music: A contig mapping approach,” in *International Symposium on Computer Music Modeling and Retrieval (CMMR)*, U. K. Wiil, Ed. Esbjerg, Denmark: pringer Berlin Heidelberg, 2004, pp. 1–20.
- [14] A. Ishigaki, M. Matsubara, and H. Saito, “Prioritized contig combining to segregate voices in polyphonic music,” in *Proc. Sound and Music Computing Conference (SMC)*, vol. 119, Padova, Italy, 2011, p. 58.
- [15] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé, “Comparing voice and stream segmentation algorithms,” in *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015, pp. 493–499.
- [16] —, “Improving voice separation by better connecting contigs,” in *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 164–170.
- [17] B. Duane and B. Pardo, “Streaming from MIDI using constraint satisfaction optimization and sequence alignment,” in *Proc. of the 2009 International Computer Music Conference (ICMC)*, Montreal, Canada, 2009.
- [18] S. T. Madsen and G. Widmer, “A complexity-based approach to melody track identification in midi files,” in *Proc. of the International Workshop on Artificial Intelligence and Music*, Hyderabad, India, 2007.
- [19] I. Karydis, A. Nanopoulos, A. Papadopoulos, E. Cambouropoulos, and Y. Manolopoulos, “Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data,” in *Proc. of Sound and Music Computing Conference (SMC)*, Lefkada, Greece, 2007, pp. 299–306.
- [20] D. Makris, I. Karydis, and E. Cambouropoulos, “Visa3: Refining the voice integration/segregation algorithm,” in *Proc. Sound and Music Computing Conference (SMC)*, Hamburg, Germany, 2016, pp. 266–273.
- [21] P. B. Kirilin and P. E. Utgoff, “Voise: Learning to segregate voices in explicit and implicit polyphony,” in *Proc. of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, UK, 2005, pp. 552–557.
- [22] D. Rizo, P. J. P. De León, C. Pérez-Sancho, A. Pertusa, and J. M. I. Quereda, “A pattern recognition approach for melody track selection in midi files,” in *Proc. of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, Victoria, Canada, 2006, pp. 61–66.
- [23] R. de Valk, T. Weyde, E. Benetos *et al.*, “A machine learning approach to voice separation in lute tablature,” in *Proc. of the 14th International Society for Music*

Information Retrieval Conference (ISMIR), Curitiba, Brazil, 2013, pp. 555–560.

- [24] T. Weyde and R. de Valk, “Chord- and note-based approaches to voice separation,” in *Computational Music Analysis*. Springer, 2016, pp. 137–154.
- [25] P. Gray and R. Bunescu, “From note-level to chord-level neural network models for voice separation in symbolic music,” *arXiv preprint arXiv:2011.03028*, 2020.
- [26] F. Simonetta, C. E. C. Chacón, S. Ntalampiras, and G. Widmer, “A convolutional approach to melody line identification in symbolic scores,” in *Proc. of the 20th Int. Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 924–931.
- [27] R. de Valk and T. Weyde, “Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations,” in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 281–288.
- [28] P. Gray and R. C. Bunescu, “A neural greedy model for voice separation in symbolic music.” in *Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 782–788.
- [29] A. McLeod and M. Steedman, “Hmm-based voice separation of midi performance,” *Journal of New Music Research (JNMR)*, vol. 45, no. 1, pp. 17–26, 2016.
- [30] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 22, no. 8, pp. 888–905, 2000.
- [31] A. Ng, M. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” Vancouver, Canada, 2001, pp. 849–856.
- [32] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.
- [33] J. Burkardt, “The truncated normal distribution,” *Department of Scientific Computing Website, Florida State University*, pp. 1–35, 2014.
- [34] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.

VOCANO: A NOTE TRANSCRIPTION FRAMEWORK FOR SINGING VOICE IN POLYPHONIC MUSIC

¹Jui-Yang Hsu ²Li Su

¹Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

²Institute of Information Science, Academia Sinica, Taipei, Taiwan
B05901022@ntu.edu.tw, lisu@iis.sinica.edu.tw

ABSTRACT

High variability of singing voice and insufficiency of note event annotation present a huge bottleneck in singing voice transcription (SVT). In this paper, we present VOCANO, an open-source VOCAL NOTE transcription framework built upon robust neural networks with multi-task and semi-supervised learning. Based on a state-of-the-art SVT method, we further consider virtual adversarial training (VAT), a semi-supervised learning (SSL) method for SVT on both clean and accompanied singing voice data, the latter being pre-processed using the singing voice separation (SVS) technique. The proposed framework outperforms the state of the arts on public benchmarks over a wide variety of evaluation metrics. The effects of the types of training models and the sizes of the unlabeled datasets on the performance of SVT are also discussed.

1. INTRODUCTION

Singing voice transcription (SVT), the task to map singing voice to common music notation of note events, is a critical step to drive novel applications in music retrieval, content creation, musicology, education, and human-computer interaction [1]. Similar to many of the automatic music transcription (AMT) tasks, the SVT task typically encompasses several sub-tasks of AMT, which are pitch detection, onset detection, offset detection, as well as sequence-level modeling [2, 3]. In the literature of music information retrieval (MIR), one of the most extensively investigated sub-tasks of SVT might be *vocal melody extraction*, the task to transcribe the *frame-level* instantaneous pitch (i.e., fundamental frequency (F0)) contours of singing voice in either monophonic (no accompaniment) or polyphonic (mixed with accompaniment) audio signals. Specifically, recent endeavours mostly focus on leveraging deep learning techniques to transcribe the singing voice which serves as a predominant melody in polyphonic music [4–7]. Though achieving breakthrough performance, vocal melody extraction is however not yet a complete so-

lution of SVT, as it does not specify *note-level* events distinct from its outputs rendered in time frames.

The challenges of note-level SVT are multi-fold. Vocal signals are highly variable in singing timbre, articulation, intonation, discernible patterns such as *vibrato*, *glissando*, note transitions and ornaments, and even lyrics. These variables blurs the boundaries between notes and notes, and make the note-level data annotation of singing voice an extremely challenging job. It seems to be impossible to compile large-scale, accurate and consistent human-annotated datasets, especially on note transition (e.g., onset and offset) time. Such variability also challenges a model to discriminate local time-frequency patterns. For example, an offset event can be overlapped with another onset event to a flexible overlapping ratio, making the transition a non-Markovian process [8]. This issue is even worsen in polyphonic music in which the note transitions in accompaniments are much denser than in the vocal melody.

In this paper, we propose a novel SVT framework to address these issues. We notice that, as the annotated datasets are limited, using advanced regularized neural networks against overfitting, and semi-supervised learning (SSL) to leverage massive amounts of unlabeled data emerge as an efficient solution. Based on the hierarchical classification approach of transcription [8], we utilize the PyramidNet with ShakeDrop regularization to reduce overfitting [9], and also incorporate it with virtual adversarial training (VAT) [10] for SSL. These techniques have been found useful in the fields of computer vision, while their potential on MIR tasks has not been thoroughly discussed.

The major technical novelty and contribution of this paper are as follows. First, to the best of our knowledge, this paper represents one of the first implementations of note-level SVT considering mixture audio inputs. Second, the proposed SVT method outperforms state-of-the-art methods. Also, the effect of SSL mechanism together with the model choice on SVT performance are discussed. Section 2 will give an overview and paper survey on the SVT problem scenario that will be discussed in this paper. Method and experiment results will be given in Section 3 and 4, respectively. Conclusion will be made in Section 5.

2. PROBLEM SCENARIOS AND BACKGROUND

The problem scenario of SVT is not consistently defined in the literature. First, the transcription results can be in either



frame-level (e.g., vocal melody extraction) or note-level. Second, the input data can be either monophonic singing or with accompaniment. Third, the target of transcription can be either solo voice or multiple concurrent voices (e.g., choir). In this work, we consider SVT of a single voice without or with instrument accompaniment, which will be referred to as the monophonic or polyphonic SVT later on. There are two approaches to deal with the case when the accompaniment is present: 1) train a general SVT model using the singing voice data mixed with accompaniment, and 2) train a specialized SVT model using clean singing voice data, and use singing voice separation (SVS) tools to remove accompaniments of the input before inference. In the second approach, monophonic and polyphonic SVT can be regarded as the same task, based on the fact that SVS is a relatively well developed technology. For simplicity, we will focus on the second approach in this paper. A pilot study comparing the two approaches will also be reported in Section 4.3.

Previous note segmentation works on SVT usually employ state-space machines such as Bayesian models or hidden Markov models (HMM), which consistently detect onset and offset by characterizing the temporal dynamics among the states (attack, sustain, and silence, etc.) of note events [11–14]. Tony [13], a widely-used note transcription software, is also based on this approach. In the deep learning approach, the connectionist temporal classification (CTC) loss [15], self-attention mechanism [3] also play similar roles in temporal decoding of note-level SVT. However, it has been pointed out that the state space in note transition can be ambiguous in several cases when onset and offset events are overlapped or when note pitch are repeated, and this issues can be solved by extending the output dimension of the network to describe the different classes of transition states [8]. Similar ideas such as multi-state note models have also been discussed recently in piano AMT tasks [16].

To our knowledge, a note-level SVT tool specifically for the singing voice signals mixed with accompaniment has rarely been implemented. It is not until 2020 that the task of “Singing Transcription from Polyphonic Music” was proposed in Music Information Retrieval Evaluation eXchange (MIREX), while there was only one submission to this campaign.¹ Related tasks include singing voice separation (SVS) [6,17] and automatic transcription of multiple concurrent singing voices such as *a cappella* [18,19], most of which are restricted to frame-level transcription.

3. METHOD

The proposed SVT framework is shown in Figure 1. In the training stage, data representations are extracted for each frame, and are then fed into two neural network models, one for pitch contour extraction and the other for note segmentation. SSL is performed on the note segmentation network. Note-level transcription results are obtained through

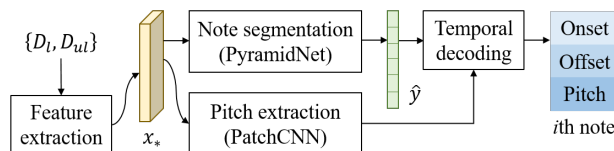


Figure 1. The proposed SVT framework. D_l , D_{ul} , x , and \hat{y} represent labeled dataset, unlabeled dataset, input sample and predicted label, respectively.

a temporal decoding process over the frame-level outputs.

3.1 Data representation

All the input audio signals are sampled at 16 kHz. For all the polyphonic data (e.g., singing plus accompaniment), an SVS algorithm, Demucs [17], is employed to separate out singing voice for use before the feature extraction stage.

Following the previous state-of-the-art method [8], the input of the SVT network is a multi-channel feature consisting in spectrum, generalized cepstrum and the generalized cepstrum of spectrum (GCoS) [20]. Such a combination has been shown effective in enhancing F0 components while suppressing unwanted harmonic components [20]. All channels of feature are mapped into the log-frequency scale with a filterbank containing 174 overlapped triangular filters allocated from 80 Hz to 1 kHz with 48 bins per octave. To adapt to signal-level attributes in different resolution, three windows with different sizes are employed to compute these data representations. As a result, the input feature has 9 channels. For every time step at t , the input $x(t)$ contains the data representations at frame t and also at its previous and future 9 frames, totaling 19 frames. In other word, the shape of $x(t)$ is: (number of channel, height, width) := (9, 174, 19).

3.2 Note segmentation networks

We decompose the SVT process into two parts: frame-level pitch extraction and note segmentation. For pitch extraction, we directly use a vocal melody extraction network, Patch-CNN [21], to obtain frame-level pitch contours. Since the frame-level pitch extraction has been a widely investigated technique (see discussion on vocal melody extraction in Section 1), the proposed network therefore focuses on note segmentation.

The note segment network can be regarded as an re-implementation and extension of [8]. First, while the model in [8] concatenates the 9 data representations as a single-channel inputs, in this work we reshape them into 9 individual channels, as shown in Section 3.1. Second, while [8] was based on the ResNet-18 network [22], we instead consider the PyramidNet with ShakeDrop regularization [23] to reduce overfitting. The PyramidNet improves the performance by gradually increasing the numbers of feature maps through the layers such as to effectively increase the diversity of high-level attributes [9]. ShakeDrop regularization further diversifies the feature maps by assigning different random weights in forward and backward

¹ https://www.music-ir.org/mirex/wiki/2020:Singing_Transcription_from_Polyphonic_Music_Results

stages at each residual layer [23]. In this work, we adopt the PyramidNet-110 architecture, which has 28.49M parameters, a size larger than ResNet-18 by 2.5 times.

Following [8], the output of the note segmentation network is optimized with multiple sub-tasks to capture the complex dynamics of music note transition. For each time step, the network outputs a 6-dimensional vector $\hat{y} := [s, a, o, \bar{o}, f, \bar{f}]$, where s represents the silence state, a represents the activation (i.e. a note is on) state, o represents the onset state, and f represents the offset state. \bar{o} and \bar{f} are the non-onset and non-offset states, respectively. The output at time t is denoted as $\hat{y}(t)$, in which the states are denoted as $s(t)$, $a(t)$, and so on. Each state in \hat{y} represents a probability value between zero and one, and we simply set $\bar{o} := 1 - o$, $\bar{f} := 1 - f$, and $s = 1 - a$. We also define the transition state $t := \max(o, f)$ to describe the state that either an onset of an offset occur. Defining the subspaces $\hat{y}_{\text{tri}} := [s, a, t]$, $\hat{y}_{\text{act}} := [s, a]$, $\hat{y}_{\text{on}} := [o, \bar{o}]$, $\hat{y}_{\text{off}} := [f, \bar{f}]$, then the total objective function for note segmentation is

$$\mathcal{L}_{\text{SEG}}(y, \hat{y}) := \text{BCE}(y_{\text{tri}}, \hat{y}_{\text{tri}}) + \text{BCE}(y_{\text{act}}, \hat{y}_{\text{act}}) + \text{BCE}(y_{\text{on}}, \hat{y}_{\text{on}}) + \text{BCE}(y_{\text{off}}, \hat{y}_{\text{off}}), \quad (1)$$

where y , y_{tri} , y_{act} , y_{on} , and y_{off} are the ground truth, and BCE is binary cross-entropy. In brief, the note segmentation mechanism here is not merely to classify onset and offset individually, but is a combination of four classification sub-tasks over the subspaces in the output space y : one sub-task of multi-class classification over transition, activation, and silence, and three sub-tasks of binary classification (i.e. activation/silence, onset/non-onset, and offset/non-offset). Such design facilitates the discrimination between possibly overlapped events, such as onset and offset (when the offset of a note followed by the onset of its next note) and smooth note transition.

3.3 Semi-supervised learning

The Virtual Adversarial Training (VAT) technique is used for semi-supervised learning on both the labeled and unlabeled training data. VAT can be regarded as an effective data/ label augmentation technique without the needs of prior domain knowledge. Let x_l and x_{ul} be labeled and unlabeled samples sampled from a labeled dataset \mathcal{D}_l and unlabeled dataset \mathcal{D}_{ul} , respectively. Given a sample x_* which is either x_l or x_{ul} , the output distribution can be represented as $p(y|x, \theta)$, in which θ represents the parameters of the note segmentation model. In our case, VAT aims at minimizing the below local distributional smoothness (LDS) function for every x_* :

$$\text{LDS}(x_*, \theta) = \text{BCE}(p(y|x_*, \theta), p(y|x_* + r_{\text{adv}}, \theta)), \quad (2)$$

$$r_{\text{adv}} := \arg \max_{r: \|r\|_2 < \epsilon} \text{BCE}(p(y|x_*, \theta), p(y|x_* + r)).$$

Let N_l and N_{ul} are the number of samples in \mathcal{D}_l and \mathcal{D}_{ul} , respectively, we have the total VAT loss being $\mathcal{L}_{\text{VAT}} := 1/(N_l + N_{ul}) \sum_{x \in \mathcal{D}_l \cup \mathcal{D}_{ul}} \text{LDS}(x_*, \theta)$. Combined with the supervised loss function (Equation (1)), the total loss function is represented as $\mathcal{L} := \mathcal{L}_{\text{SEG}} + \lambda \mathcal{L}_{\text{VAT}}$,

and we set $\lambda = 1$ throughout this work. The note segmentation network is implemented with PyTorch v1.5, and is obtained after 20 epochs of training on an Nvidia TITAN RTX GPU, using the AdamW optimizer with a learning rate of 10^{-4} . Typically, it takes around 8 hours to accomplish training a model.

3.4 Temporal decoding

Post-processing is needed to derive temporally consistent onset/ offset/ activation timestamps from the 6-D distribution (i.e. $\hat{y}(t)$) outputted from the network. We call this process *temporal decoding*. First, we employ a linear filter with impulse response as a 5-tap triangular window to smooth each dimension in $\hat{y}(t)$ in the time axis. Then, we perform peak picking on $\hat{o}(t)$ and $\hat{x}(t)$ with a threshold at 0.5 to determine possible onset and offset positions, respectively. At this stage, there are inevitable mismatches between the predicted onset and offset positions. To ensure that every onset is followed by exactly one offset, additional procedures are used: 1) if there are two onsets having no offset between them, we insert an offset specified to the time when s firstly surpasses a with that interval; 2) similarly, if there are two offsets having no onset between them, the inserted onset is specified to the time when a firstly surpasses s in that interval; and 3) any predicted result violating rules 1) and 2) is removed and is not recognized as an onset or an offset.

After having the onset-offset interval of every predicted note, the pitch of every note is determined by the median value of the pitch contour within that onset-offset interval.

4. EXPERIMENT

4.1 Data

To test the robustness of our model, a cross-dataset scenario (i.e. the training and testing datasets are compiled independently) is employed for the experiments. The dataset used for supervised learning (denoted as \mathcal{D}_l) is the TONAS dataset, which contains 71 flamenco a cappella sung melody, each of which has high-quality note-level annotation [24]. We consider three datasets for semi-supervised learning (denoted as \mathcal{D}_{ul}), which are MIR-1K,² MedleyDB [25] and DALI [26]. MIR1K contains 1,000 excerpts of Chinese karaoke songs sung by amateur singers. MedleyDB is a multi-track dataset, and we select the tracks labeled as ‘female singer’ or ‘male singer’ (76 tracks in total) as our unlabeled training data. The DALI dataset contains a large-scale polyphonic music (mostly Western pop music). In this dataset We select 65 songs from this dataset as unlabeled training data, and this subset is denoted as DALI-train hereafter.

For evaluation, we consider three testing datasets (denoted as $\mathcal{D}_{\text{test}}$), which are ISMIR2014, DALI-test, and Cmedia. ISMIR2014 [27] is a monophonic vocal singing dataset containing singing data from 11 female adults, 13

² <https://sites.google.com/site/unvoicedsoundseparation/mir-1k>

male adults and 14 children. The DALI-test set, also selected from DALI, contains 20 songs with automated annotation of notes. Finally, the Cmedia dataset [28] is used in the MIREX campaign on polyphonic SVT (see footnote 1), and on the list we retrieve 99 pop songs (mostly Chinese songs) with vocal annotation publicly available. The list of the songs selected from DALI and Cmedia are provided on the project website (see Section 5).

4.2 Evaluation metrics

We use the metrics of note transcription in the `mir_eval` library for evaluation [29]. In the evaluation rules, a predicted note is considered as correct (i.e., true positive) for a ground truth note if it fulfills the three rules: 1) the difference in pitch number between the predicted note and the ground truth note is less than a pitch tolerance value δp (in cents), 2) the difference in onset time is less than an onset tolerance value δo (in seconds), and 3) the difference in offset time is less than $\max(\delta o, \delta f \times g)$, where δx is an offset tolerance ratio and g is the duration of the ground truth note (in seconds). The F1-score is the harmonic mean of the precision and recall values obtained from these criteria.

Therefore, the F1-score is parametrized by $(\delta p, \delta o, \delta f)$, and is denoted as $\mathcal{F}_{(\delta p, \delta o, \delta f)}$ in this paper. This incorporates several conventional metrics of note-level transcription. Setting $\delta p = 50$ cents, $\delta o = 50$ ms and $\delta f = 0.2$, we consider the following F1-scores (a tolerance value of ∞ means that it is not considered in the evaluation):

- Onset-only F1-score: $\mathcal{F}_{(\infty, 0.05, \infty)}$
- Offset-only F1-score: $\mathcal{F}_{(\infty, \infty, 0.2)}$
- Onset-offset F1-score: $\mathcal{F}_{(\infty, 0.05, 0.2)}$
- Onset-pitch F1-score: $\mathcal{F}_{(50, 0.05, \infty)}$
- Onset-offset-pitch F1-score: $\mathcal{F}_{(50, 0.05, 0.2)}$

For example, $\mathcal{F}_{(50, 0.05, 0.2)}$ means that a note is considered as a true positive if its pitch deviates from the ground truth pitch by less than 50 cents, its onset deviates from the ground truth onset by less than 0.05s, and its offset deviation is less than 0.2 times the duration of the ground truth note. The F1-scores of only onset (or only offset) events are the cases of $\delta p = \delta f = \infty$ (or $\delta p = \delta o = \infty$).

Besides the note-level F1-scores, we also propose a high-level metric called the sequence-level Note Accuracy (NAcc), which is based on matching the MIDI pitches of predicted and ground truth note sequences rather than the timestamps of onset/offset. More specifically, NAcc is the Levenshtein distance between the ground truth and the predicted MIDI sequences: $\text{NAcc} := 1 - (D + I + S)/N$, where D denotes the number of deletions, I is the number of insertions, S is the number of substitutions, and N is the length of the ground truth sequence. Unlike the F1-score, NAcc can better reveal the performance on the entire pitch sequence, rather than the performance on the time stamps of note events. This evaluation is useful when accurate time stamps of the output are not of primary importance while the global information of pitch sequence is required.

4.3 Results

4.3.1 Effect of singing voice separation

First, as a pilot study, we compare the two SVT approaches for polyphonic audio mentioned in Section 2: 1) a model directly trained with polyphonic \mathcal{D}_{ul} , and 2) a model trained with SVS-processed (i.e. monophonic) \mathcal{D}_{ul} , and requiring SVS in the inference stage. Using MIR1K as \mathcal{D}_{ul} and ISMIR2014 as \mathcal{D}_{test} , results show that the onset-offset-pitch F1-score is 30.04% for the first model, while the second model achieves 68.38%, a much better performance. This is mainly due to the domain difference between \mathcal{D}_l and \mathcal{D}_{ul} (the former is purely monophonic while the latter is polyphonic). We therefore focus on the second approach in evaluating the proposed SVT framework.

4.3.2 Comparison of models

Table 1 compares the performance metrics of two models (ResNet-18 and PyramidNet) trained under a supervised scheme (w/o VAT), and a semi-supervised schemes (w/i VAT) with three different unlabeled datasets (\mathcal{D}_{ul}) having different scales: MIR1K, MIR1K + MedleyDB, and also MIR1K + MedleyDB + DALI-train.

A comparison of the two models is first made from the left three columns of Table 1 (without VAT). ResNet-18 outperforms PyramidNet for onset F1-score, onset-pitch F1-score and NAcc, while PyramidNet prevails on offset detection and gives better onset-offset-pitch F1-scores on the three datasets. In short, PyramidNet, with a larger size of training parameters, performs better on strict note transcription metric such as onset-offset-pitch F1-score.

4.3.3 Effects of semi-supervised learning

By comparing the results without VAT and the ones with VAT, we observe two different trends for the two models. For PyramidNet, using VAT improves the performance for almost all \mathcal{D}_{test} and all the metrics. For example, with MIR1K as \mathcal{D}_{ul} improves the onset-offset-pitch F1-score of the three test datasets by 5.73, 0.08 and 3.01 percentage points, respectively. Since all the methods adopt the same pitch extraction results, such improvement is fully contributed by the improvement of note segmentation network with semi-supervised learning.

For ResNet-18, however, using VAT only improves the performance of offset-related metrics rather than all metrics. This is possibly because VAT performs more effectively on larger models with regularization mechanism. Among the improvement of offset detection metrics, it is worth mentioning that the offset F1-score of the ISMIR2014 dataset is improved by 3.83 percentage points (from 74.68% to 78.51%) with the \mathcal{D}_{ul} being MIR1K+Med+DALI. In summary, although VAT does not improve the performance consistently over all types of models on all performance metrics, it still exhibits a trend to improve more challenging metrics such as offset.

4.3.4 Effects of the unlabeled dataset \mathcal{D}_{ul}

Table 1 also demonstrates that the size, quality, and diversity of the unlabeled dataset (\mathcal{D}_{ul}) affect the performance

	w/o VAT			w/i VAT								
\mathcal{D}_{ul}	-			MIR-1K			MIR-1K + MedleyDB			MIR1K + Med + DALI		
\mathcal{D}_{test}	I	D	C	I	D	C	I	D	C	I	D	C
PyramidNet + ShakeDrop												
Onset-only	78.02	27.79	58.15	84.04	32.87	64.56	81.10	28.98	60.65	82.25	30.40	61.81
Offset-only	75.50	36.32	45.24	80.06	33.95	51.86	78.52	33.95	48.29	78.42	34.87	47.43
Onset-offset	62.92	11.16	25.95	68.60	11.95	33.65	67.06	10.85	29.97	67.12	11.13	29.65
On-off-pitch	62.65	2.69	22.36	68.38	2.76	28.28	66.73	2.65	25.37	66.92	2.68	25.05
Onset-pitch	75.32	5.26	45.57	80.58	5.99	48.33	78.26	5.35	45.95	78.72	5.50	47.28
NAcc	69.76	12.30	50.54	78.68	12.06	48.52	75.51	12.78	47.88	77.41	15.43	52.36
ResNet												
Onset-only	82.01	30.05	60.12	79.39	26.70	55.87	78.85	26.71	55.80	78.38	26.68	55.68
Offset-only	74.68	35.38	45.31	76.76	33.23	44.18	76.11	33.32	46.94	78.51	33.29	46.47
Onset-offset	61.80	10.46	26.60	62.93	9.91	25.10	62.93	9.65	26.53	63.32	9.49	26.32
On-off-pitch	61.71	2.51	22.95	62.76	2.42	21.66	62.77	2.21	22.60	63.04	2.28	22.44
Onset-pitch	77.97	5.89	47.87	75.90	5.19	43.76	74.87	4.93	43.12	74.78	5.11	43.51
NAcc	80.08	16.06	56.24	73.29	4.11	43.08	74.20	11.17	48.28	78.16	10.79	48.38

Table 1. Evaluation results on three test datasets (I: ISMIR2014; D: DALI-test; C: Cmedia). The evaluation metrics are (from top to bottom): onset F1, offset F1, onset-offset F1, onset-offset-pitch (on-off-pitch) F1, and pitch-onset F1. See Section 4.2 for more details on the evaluation metrics. The best performances of each dataset are marked in bold. Upper: PyramidNet with ShakeDrop. Lower: ResNet-18.

with VAT in a quite complicated way. First, it should be noted that a larger-scale of unlabeled dataset (\mathcal{D}_{ul}) does not always imply better performance, and this phenomenon is also model-dependent. First, for PyramidNet, optimal performances mostly occur when only MIR1K is taken as \mathcal{D}_{ul} , and adding MedleyDB and DALI-train does not guarantee better results. For ResNet, it can be observed that a larger unlabeled dataset (MIR1K+Med+DALI) does give better results, but this trend is more obvious only in offset-related metrics. A possible reason is that the genres of the three \mathcal{D}_{ul} are quite different. Both MedleyDB and DALI-train contain a much wider ranges of singing styles, usually with chorus singing, while MIR1K is less diverse and can be better optimized when training in batch. Nevertheless, using MIR1K+MedleyDB+DALI-train still outperforms the case using MIR-1K+MedleyDB, and this indicates that there is still room for improvement if incorporating more unlabeled data for semi-supervised learning.

Among the three testing datasets, DALI-test is obviously the most challenging and is hard to be improved by VAT. This is because that the note event annotation in DALI is obtained automatically from global alignment, and is reported to be error prone [30]. Besides, the chorus singing part, which is commonly seen in the DALI dataset, may confound the result of monophonic pitch extraction. This can be seen from the fact that the performance greatly drops when considering pitch for DALI-test set: its onset-offset-pitch F1-scores are always much lower than its onset-offset F1-scores. These challenging issues might still require solutions from supervised learning rather than the SSL approaches.

4.3.5 Sequence-level vs. note-level evaluation

It is worth noting that NAcc exhibits a trend different from other metrics. A high onset-offset-pitch F1-score does not

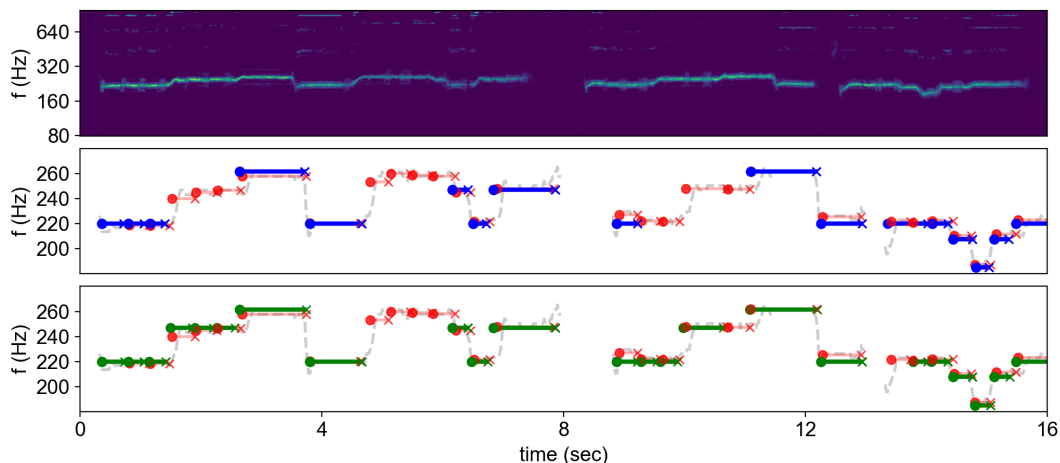
Method	P	R	F
[31]	30.4	31.5	30.8
[24]	43.0	37.3	39.8
[32]	39.7	44.0	41.5
[12]	40.9	43.6	42.1
[13]	51.0	53.4	52.0
[8]	62.5	56.9	59.4
ResNet w/o VAT	63.1	60.6	61.7
ResNet w/i VAT	67.7	58.8	62.8
PyramidNet w/o VAT	68.6	58.1	62.7
PyramidNet w/i VAT	72.2	65.3	68.4

Table 2. Performance comparison (in %) of various SVT methods on the ISMIR2014 dataset ($\mathcal{D}_{ul} = \text{MIR1K}$).

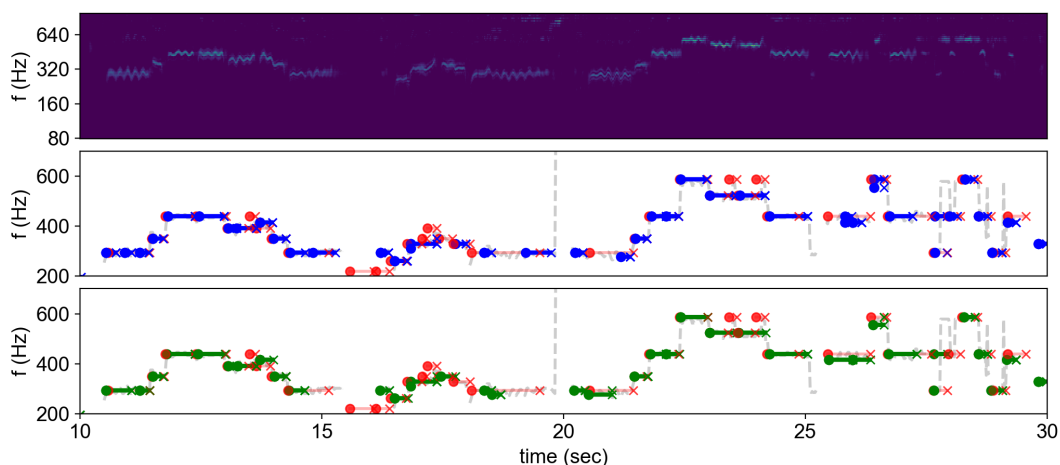
imply a high NAcc. On DALI-test and Cmedia, we observe that using larger scale of \mathcal{D}_{ul} does result in better NAcc using PyramidNet. This can be explained by the high variability of annotation of onset/offset time. With large-scale and high-diversity data, SSL may not be effective in capturing local event, but can improve the performance in the global scale. Besides, given the fact that the onset/offset annotations are not perfectly reliable, sequence-level metrics such NAcc is worth further investigation when the purpose of SVT is to transcribe the music score rather than replicating the music performance.

4.3.6 Comparison to the state of the arts

Table 2 compares the precision, recall, and F1-score of the proposed method to previous work on the ISMIR2014 dataset, the only public dataset systematically evaluated on note-level SVT (this why a comparison on polyphonic music datasets is not made here). Results of previous work are listed in the upper six rows, while the new results are



(a) 'afemale6.wav' in the ISMIR2014 dataset



(b) '10.wav' in the CMedia dataset (10-30 seconds)

Figure 2. Data representations and transcription results using the PyramidNet model. From top to bottom: data representation, transcription results without VAT (blue lines), and transcription results with VAT (green lines). Grey dashed lines are frame-level pitch contours. Red lines are ground truth. Circle dots are onset events, and crosses are offset events.

in the lower four rows. The result of ResNet w/o VAT can be regarded as an improved version of [8], by re-arranging the channels of the input data representations while following the same output dimensions and temporal decoding processes. Such modification entails 2.3 percentage points of improvement from [8]. Besides, using a model larger than ResNet-18 (i.e. PyramidNet) further improves the resulting F1-score by 1 percentage point. Finally, with the assistance of SSL, the PyramidNet model with MIR1K for VAT achieves 68.5% of F1-score, which outperforms the best previous method [8] by 9.0 percentage points.

4.3.7 Illustration

Figure 2 shows the results of two challenging examples of SVT. The first example is challenging because of the repeated notes (consecutive note with the same pitch), while the main challenge of the second example is its wide pitch range. Figure 2(a) shows that it is hard to observe the onset and offset events from the data representation. The purely supervised model fails to capture most of the onset and offset events of repeated notes, and this issue can

be partly solved by utilizing VAT; see the repeated notes captured at around 2 secs and 9 secs of the example. In Figure 2(b), it can be shown that both models without and with VAT fail to transcribe low-pitch notes and high-pitch ornamentation (around 24 secs), partly due to the fact that these events are less visible on the data representations.

5. CONCLUSION

We have validated the effectiveness of leveraging semi-supervised learning on note segmentation in singing voice transcription. State-of-the-art performance has been reported on public benchmarks. The role of semi-supervised learning is found depending on the model and the size, quality and the diversity of the unlabeled training data. These findings provide insights into future semi-supervised MIR research. The source code is available at the project page.³ VOCANO is also available as part of the automatic music transcription library Omnizart [33].⁴

³ <https://github.com/B05901022/VOCANO>

⁴ <https://github.com/Music-and-Culture-Technology-Lab/omnizart>

6. ACKNOWLEDGMENTS

This work was partially supported by MOST Taiwan under the project *Automatic Music Transcription Algorithms for Interactive Music Systems* (Grant No. 106-2218-E-001-003-MY3).

7. REFERENCES

- [1] M. Müller, E. Gómez, and Y.-H. Yang, “Computational methods for melody and voice processing in music recordings (dagstuhl seminar 19052),” in *Dagstuhl Reports*, vol. 9, no. 1, 2019.
- [2] E. Anders, “Modeling music: Studies of music transcription, music perception and music production,” Ph.D. dissertation, KTH Royal Institute of Technology, 2018.
- [3] R. Nishikimi, E. Nakamura, S. Fukayama, M. Goto, and K. Yoshii, “Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 161–165.
- [4] M.-T. Chen, B.-J. Li, and T.-S. Chi, “CNN based two-stage multi-resolution end-to-end model for singing melody extraction,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 1005–1009.
- [5] F. Rigaud and M. Radenen, “Singing voice melody transcription using deep neural networks,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 737–743.
- [6] T. Nakano, K. Yoshii, Y. Wu, R. Nishikimi, K. W. E. Lin, and M. Goto, “Joint singing pitch estimation and voice separation based on a neural harmonic structure renderer,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2019, pp. 160–164.
- [7] S. Kum, J.-H. Lin, L. Su, and J. Nam, “Semi-supervised learning using teacher-student models for vocal melody extraction,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 93–100.
- [8] Z.-S. Fu and L. Su, “Hierarchical classification networks for singing voice segmentation and transcription,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 900–907.
- [9] D. Han, J. Kim, and J. Kim, “Deep pyramidal residual networks,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5927–5935.
- [10] T. Miyato, S. Maeda, M. Koyama, and S. Ishii, “Virtual adversarial training: a regularization method for supervised and semi-supervised learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 41, no. 8, pp. 1979–1993, 2018.
- [11] R. Nishikimi, E. Nakamura, K. Itoyama, and K. Yoshii, “Musical note estimation for F0 trajectories of singing voices based on a Bayesian semi-beat-synchronous HMM,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 461–467.
- [12] L. Yang, A. Maezawa, J. B. Smith, and E. Chew, “Probabilistic transcription of sung melody using a pitch dynamic model,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 301–305.
- [13] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon, “Computer-aided melody note transcription using the tony software: Accuracy and efficiency,” in *Proc. Sound and Music Computing (SMC)*, 2015.
- [14] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, “Bayesian singing transcription based on a hierarchical generative model of keys, musical notes, and F0 trajectories,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, pp. 1678–1691, 2020.
- [15] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, “An end-to-end framework for audio-to-score music transcription on monophonic excerpts,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 34–41.
- [16] T. Kwon, D. Jeong, and J. Nam, “Polyphonic piano transcription using autoregressive multi-state note model,” in *International Society for Music Information Retrieval Conference (ISMIR)*. International Society for Music Information Retrieval, 2020, pp. 454–461.
- [17] A. Défossez, N. Usunier, L. Bottou, and F. Bach, “Music source separation in the waveform domain,” *arXiv preprint arXiv:1911.13254*, 2019.
- [18] R. Schramm and E. Benetos, “Automatic transcription of a cappella recordings from multiple singers,” in *AES International Conference on Semantic Audio*. Audio Engineering Society, 2017.
- [19] H. Cuesta, B. McFee, and E. Gómez, “Multiple F0 estimation in vocal ensembles using convolutional neural networks,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 302–309.
- [20] Y.-T. Wu, B. Chen, and L. Su, “Automatic music transcription leveraging generalized cepstral features and deep learning,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 401–405.

- [21] L. Su, “Vocal melody extraction using patch-based CNN,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 371–375.
- [22] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [23] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, “Shakedrop regularization for deep residual learning,” *IEEE Access*, vol. 7, pp. 186 126–186 136, 2019.
- [24] E. Gómez and J. Bonada, “Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing,” *Computer Music Journal*, vol. 37, no. 2, pp. 73–90, 2013.
- [25] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, “MedleyDB: A multitrack dataset for annotation-intensive mir research.” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 155–160.
- [26] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, “DALI: a large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm.” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 431–437.
- [27] E. Molina, A. M. Barbancho-Perez, L. J. Tardón, I. Barbancho-Perez *et al.*, “Evaluation framework for automatic singing transcription,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2014.
- [28] J.-Y. Wang and J.-S. R. Jang, “On the preparation and validation of a large-scale dataset of singing transcription,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 276–280.
- [29] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, “mir_eval: A transparent implementation of common mir metrics,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 367–372.
- [30] G. Meseguer-Brocal, R. M. Bittner, S. Durand, and B. Brost, “Data cleansing with contrastive learning for vocal note event annotations,” in *Proc. International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 255–262.
- [31] M. P. Rynänen and A. P. Klapuri, “Automatic transcription of melody, bass line, and chords in polyphonic music,” *Computer Music Journal*, vol. 32, no. 3, pp. 72–86, 2008.
- [32] E. Molina, L. J. Tardón, A. M. Barbancho, and I. Barbancho, “SiPTH: Singing transcription based on hysteresis defined on the pitch-time curve,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 23, no. 2, pp. 252–263, 2015.
- [33] Y.-T. Wu, Y.-J. Luo, T.-P. Chen, I. Wei, J.-Y. Hsu, Y.-C. Chuang, and L. Su, “Omnizart: A general toolbox for automatic music transcription,” *arXiv preprint arXiv:2106.00497*, 2021.

DE-CENTERING THE WEST: EAST ASIAN PHILOSOPHIES AND THE ETHICS OF APPLYING ARTIFICIAL INTELLIGENCE TO MUSIC

Rujing Huang

Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Stockholm

Bob L. T. Sturm

Division of Speech, Music and Hearing, KTH Royal Institute of Technology, Stockholm

Andre Holzapfel

Division of Media Technology and Interaction Design, KTH Royal Institute of Technology, Stockholm

{rujing, bobs, holzap}@kth.se

ABSTRACT

Questions about the ethical dimensions of artificial intelligence (AI) become more pressing as its applications multiply. While there is a growing literature calling attention to the ethics of AI in general, sector-specific and culturally sensitive approaches remain under-explored. We thus initiate an effort to establish a framework of ethical guidelines for music AI in the context of East Asia, a region whose rapid technological advances are playing a leading role in contemporary geopolitical competition. We draw a connection between technological ethics and non-Western philosophies such as Confucianism, Buddhism, Shintoism, and Daoism. We emphasize interrelations between AI and traditional cultural heritage and values. Drawing on the IEEE Principles of Ethically Aligned Design, we map its proposed ethical principles to East Asian contexts and their respective music ecosystem. In this process of establishing a culturally situated understanding of AI ethics, we see that the seemingly universal concepts of “human rights”, “well-being”, and potential “misuse” are ultimately fluid and need to be examined in specific cultural contexts.

1. INTRODUCTION

The field of Music Information Retrieval (MIR) involves developing artificial intelligence (AI) technologies for making music accessible, evinced by applications such as music recommendation, identification, analysis and generation. Such technologies augment or replace human efforts by their scalability. The impacts of these technologies on music practices and communities are important subjects of investigation as the MIR research field progresses.

As analyzed in Clancy’s PhD thesis on music AI [1], music takes place in a complex network of “human and non-human (AI) ‘members organisms’ located in civic, industrial or academic domains, who can be considered as stakeholders of the global music community” – a network that Clancy refers to as the *music ecosystem*. One may be tempted to delineate this ecosystem in terms of the organ-

izations and individuals involved in the Western music industry on the one hand, and the “listener” on the other. But the shapes music industries take, and the ways people interact with music change depending on the cultural context [2]. In all of these environments, various music AI applications can transform existing practices in anticipated and unanticipated ways.

Some ethical implications of music AI have been outlined previously [3]. These implications either coincide or extend ethical considerations related to AI in general, as they have been increasingly discussed and documented throughout recent years in various ethical guidelines (e.g. [4]). As with the development of technology, however, most propositions for ethical guidelines are deeply entwined in value judgements of Western societies [5]. But what of the values of non-Western societies? This problem has been recognized [6], and has led to a discussion of how an intercultural information ethics (IIE) may arrive at ethical guidelines that facilitate the development of diverse technologies that enable members of various societies to thrive through interacting with it. With a significant amount of research and development in music AI taking place in East Asia (e.g. [7-12]), we therefore advance the introduction of ideas such as IIE to MIR research. The diversity of cultural backgrounds of music AI stakeholders makes it timely to ask: how can ethical guidelines in MIR encompass this diversity [13]?

We begin this endeavor with a short overview of existing ethical guidelines for AI. We then build a bridge to East Asian philosophies and their relation to technology in Section 3. Section 4 interprets central notions of recent ethical guidelines through the lens of a few East Asian philosophical traditions. Section 5 relates these interpretations to current developments in various fields of study, with the goal to indicate ways to implement intercultural perspectives on ethics of music AI.

While it is not possible to address most East Asian philosophical traditions here, we focus on some schools of thought that are inspiring scholarly discussions around the topic of technological ethics. We juxtapose these existing discussions, develop them, and re-situate them in the context of music AI. We hope this can serve as a starting point for future endeavors to examine the impacts of different philosophical traditions around the world on guiding thinking about the ethics of music AI. This paper, largely theoretical in nature, will complement an upcoming book



chapter by the authors that takes a more applied approach by bringing forth the voices of prominent music AI researchers, developers, and practitioners across Asia, whose reflections over the nuances of practicing ethical music AI will further enrich the current discussion.

2. A REVIEW

Given recent advancements and controversies involving AI technology, the ethical implications of integrating such technology into public, private and commercial spheres have become issues of compelling interest to people, companies, and governments [5]. This has led to the creation of research forums like the ACM Conference on Fairness, Accountability, and Transparency,¹ crowd-sourced initiatives like the AI Incident Database,² the formation of corporate ethics committees, inquiries by government bodies,³ and focus groups of professional global organizations.

The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems⁴ consists of engineers from six continents, and has produced two editions of, “Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems” (EADv2) [4]. This document argues for the development of autonomous systems guided by five ethical principles:

1. *human rights*: “Ensure [these technologies] do not infringe on internationally recognized human rights”
2. *well-being*: “Prioritize metrics of well-being in their design and use”
3. *accountability*: “Ensure that their designers and operators are responsible and accountable”
4. *transparency*: “Ensure they operate in a transparent manner”
5. *misuse*: “Minimize the risks of their misuse”.

EADv2 means designing with values that “put human advancement at the core of development of technical systems, in concert with the recognition that machines should serve humans and not the other way around ... to create intelligent technical systems that enhance and extend human well-being and freedom” [4]. These five principles align with several AI guidelines produced around the world [5], many coming from countries that are economically developed, and thus which could neglect meaningful local knowledge and jeopardize global fairness.

When it comes to AI and music in particular, Clancy’s PhD dissertation [1] surveys the global commercial landscape of music generation with great attention paid to the intellectual property status of music generated by machines, and how this impacts the “music ecosystem” with respect to the “value gap” – the economic disparity between content owners and creators. Clancy proposes self-regulation according to a marking system that identifies and rewards actions sustaining equitable uses of AI technology within the music ecosystem. In this way, consumers might make informed choices in order to support musicians themselves. Clancy also suggests taking a closer look at non-Western approaches to technological ethics in

order to de-center the conversation from Western philosophical thought, which inspires our paper.

3. NON-WESTERN APPROACHES TO TECHNOLOGICAL ETHICS

Discussions of technological ethics have historically been driven by Western thought rooted in Plato and Aristotle. Recent work, however, has begun to pay attention to non-Western influences on Western science and technology. Dusek [14] discusses in *Philosophy of Technology* the power and value of non-Western scientific knowledge systems and their contribution to the development of technology. Hui [15], in putting into question the affirmation of technics⁵ and technologies as anthropologically universal, argues for the urgency of establishing a philosophy of technology that is “properly Chinese”. Among the growing literature that calls attention to the ethical dimensions of AI, few have situated this topic in non-Western contexts. Among them, Hagerty and Rubinov [16], Jobin et al. [5], and Clancy [1] stand out as writings that call for a multicultural shift in addressing (music) AI and its ethics.

To lay the groundwork for our discussions in Section 4 and 5, we now review recent studies that draw a connection between technology (AI), ethics, and such East Asian philosophies as Confucianism, Buddhism, Shintoism, and Daoism. Scholars have turned to Confucianism, a system of thought based on the ancient teachings of Confucius (551-479 B.C.E), as a source that may enrich the ethics of technology. Writing on “ethical pluralism”, Ess [6] juxtaposes contemporary Western ethics with the ethical tradition of Confucian thought, focusing on their shared notions of “resonance” and “harmony” as a way of articulating “pluralistic structures of connection alongside irreducible differences”. Kirk et al. [17] contemplates Confucianism to help guide technology policy, and explores how the Chinese government builds on Confucian notions of harmony, social hierarchy, and legitimacy to inform the nation’s approach to technological governance and ethics, as well as to build public acceptance towards them. The authors emphasize the “stickiness” of Confucian values in South Korean, Japanese, and Chinese societies, as the three nations continue to foreground hierarchy, family, and social order despite their divergence of political ideology. As another example, Wong and Wang [18] argue for a “multicultural turn” in approaches to technological ethics, developing what they call “Confucian ethics of technology”. In this work, scholars investigate such normative Confucian concepts as “*dao*”,⁶ “harmony”, and “personhood” and their application to the philosophy and ethics of technology.

Buddhism is another source for ethical reflection on technology. Throughout history, technology has played an important role in both Buddhist philosophy and religion. Rambelli [20] explores the presence of machines in the Japanese Buddhist tradition, e.g., robotic monks and priests. *The Ethics of AI and Robotics: A Buddhist View-*

¹ <https://factconference.org>

² <http://incidentdatabase.ai>

³ <https://bit.ly/3xgtVZP>

⁴ <https://bit.ly/2QY1vPq>

⁵ Throughout the book, Hui uses the term “technics” to refer to the “general category of all forms of making and practice”.

⁶ One of the most fundamental yet most elusive notions in Daoist thought; Confucians use the term, often translated as “The Way”, to refer to the “organizing and governing principle of the universe” [19].

point [21] is a significant attempt to bridge the ancient tradition of Buddhism with technological ethics. Arguing for a Buddhism-inspired standard of ethical perfection, namely “machine enlightenment”, Hongladarom [21] presents ways in which Buddhism can contribute to the ethics of AI and robotics. The ethical ideal for AI promoted throughout the book grounds itself in two central Buddhist values: the realization that all things are interdependent, and the commitment to alleviate the suffering of all beings.

When studying the technology-friendly nature of Japanese society, scholars often turn to its religion Shintoism to understand the nation’s anthropomorphic view of technology.⁷ In Shinto beliefs, there is no categorical distinction between humans, animals, and inanimate objects, as the religion attributes spirits, or *kami*, to all forms of existences. Juxtaposing this “Shinto-infused techno-animism” with actor-network theory,⁸ Jensen and Blok [24] posit that Shinto cosmic views offer a vantage point for interpreting the contributions of non-humans to “collective life”, and for studying the entanglements of politics, ecology, science, and cosmos in contemporary society.

While some scholars hold that technologies are antithetical to the concept of *ziran*⁹ promoted in Daoist thought [25], a philosophical tradition primarily associated with the texts of ancient thinkers such as Laozi and Zhuangzi, the pioneering work of Joseph Needham [27] uncovers the long history of Daoism engaging with technology. A recent study of Nelson [28] reveals how ideas from Daoist texts have influenced early twentieth-century German thinkers (Buber and Heidegger) and their views on technological rationality and modernity. This further motivates critical engagement with technological ethics traversing geographical, cultural, and philosophical boundaries.

In the broader context of East Asian societies, the preservation and continuation of cultural traditions lie at the heart of conversations surrounding AI and its application. “The BSRC [Bio-Synergy Research Center] is bringing traditional medicines and cutting-edge computer science together”, writes an article [29] that introduces recent South Korean efforts to use AI and biotechnology to explore the therapeutic potentials of traditional medicines. Guo et al. [30] also study the application of AI in traditional Chinese medicine. In *2047 Apologue* [31], a conceptual theater show, the producer creatively fuses AI and robotics with Chinese folk arts to shed light on larger themes such as environmental crises. In Japan, while the Buddhist robot priest “Mindar” delivers sermons inside the 400-year-old Kodaiji temple [32], others have used AI to help sustain near-extinct traditional crafts [33].

On the website of *Aichi’s World Expo* [34], one can locate such claims as “conservation should replace mass production and consumption”. Indeed, in the race towards becoming world leaders in AI, the theme of bridging “cutting-edge technologies” with ancient cultural heritages and traditions is popular in East Asian countries. Writing on Japan’s seamless assembly of science, technology, and

culture, Šabanović explores the ways in which Japan legitimizes its adoption of new technologies through strategic association with traditional practices and cultural continuity [34]. Technologies, in this case, are perceived as culturally situated artifacts. Traditions, on the other hand, are continuously renegotiated and redefined to include emerging technological devices and practices.

4. TOWARD AN ETHICALLY ALIGNED DESIGN FOR AI IN EAST ASIA

We now examine a list of key ethical principles featured in EADv2, and investigate their meanings in East Asian contexts. It should be clarified that it is beyond the scope of this paper to address every principle that appears in major ethical guidelines. For instance, while we do not devote a section to the notion of “privacy,” there is a growing literature that addresses how “privacy” is viewed differently in East Asian societies [6, 16, 17, 35]. As “privacy” is not listed as a separate principle in EADv2, we consider it as an integral component of our discussion on “human rights”, “well-being”, and “awareness of misuse”, three principles listed in EADv2 that are powerful examples of cross-cultural pluralism and are the focus of this paper.

When approaching inter-cultural comparisons of AI ethics, we ground our analysis in a number of theoretical texts and traditions. Ess [6] draws from Platonic and Aristotelian thought to elaborate on what he calls “interpretive pluralism”, where multiple interpretations of an idea can remain “irreducibly different” from each another and yet be connected by “way of their shared point of origin and reference”. Ess relates this form of pluralism to the Confucian idea of “harmony”, where things can “resonate” in spite of fundamental differences.

Our analysis is also guided by poststructuralist ideas of plurality when it comes to the reading of texts from multiple viewpoints [36]. Our attempt to “de-center” the West in discussions of cross-cultural, technological ethics is inspired by the Derridean [37] gesture of multi-centering, that is, the recognition of multiple, simultaneous centers in the absence of one absolute center that renders the *others* unconditionally marginal. Finally, in suggesting the possibility of an East Asia guideline for AI ethics, we juxtapose our pluralistic perspective with the emerging phenomenon of Asian studies in Asia and, particularly, the work of Chen [38] that seek to use Asia, rather than the West, as an “imaginary anchoring point” for critical inquiry.

4.1 “Human Rights”

The first principle in EADv2, “Human Rights”, is fundamental in every major guideline for AI ethics. While there is little debate that the design of ethical AI should not violate human rights, what the term signifies and how that shifts with cultural context are often neglected. An-Na’im et al. [39] argue that much of our viewpoint towards human rights is biased by expectations native to our own

⁷ Scholars, for instance, have written about robots as a kind of “third existence” that can coexist with humans as social agents [22].

⁸ According to Actor-network theory, human subjects and technological artifacts should be studied with the same method, and that no analytical distinction should be made between subjects and objects [23].

⁹ A central concept in Daoist thought variously translated into “self-so”, “spontaneous”, or “natural” [26].

culture. Also central to engaging with contemporary notions of human rights is the question of what it means to be human. Alford [40] draws on Confucian notions of personhood when reflecting on the state of human rights in China. Alford’s emphasis on the social conception of the persons presupposed in Confucianism echoes Wong [19], in which Confucian personhood is characterized as inherently relational, developmental, and virtue-based.

Another question relevant to our consideration of human rights in AI ethics is whether such rights may be possessed by AI “agents”. Eastern philosophies often make little ontological distinction between humans and non-humans. AI ethicist Pak-Hang Wong comments [41] that based on the “role-based” ethics of Confucianism, one can attribute personhood to non-human beings as long as they “play ethically relevant roles and duties as humans”. This may explain why a clause in the Japanese Society for Artificial Intelligence (JSAI) Ethical Guidelines states that AI should abide by all policies described therein in order to “become a *member* or a *quasi-member* of society” [42].

When it comes to emphasizing the rights of all life forms beyond the human race, a connection can be drawn between East Asian philosophies and the emerging field of posthumanities. Braidotti [43] establishes posthuman ethics as a way of rethinking subjectivity as a collective assemblage that encompasses “human and non-human actors, technological mediation, animals, plants, and the planet as a whole”. Pondering the concept of “digital person”, Sjöberg [44] discusses the prospect of treating intelligent agents as legally responsible entities. Bridging these intellectual traditions, an ethically aligned design for AI in the 21st century might move beyond a human-centric approach and consider the rights of all “*beings*”. Such a stance resonates with Indigenous-centered AI Design as proposed by Lewis [45], and is similarly advocated by Floridi [46] who calls for constructing information ethics as a “patient-oriented, ontocentric, and ecological macroethics” that is “as non-anthropocentric as possible”.

4.2 “Well-Being”

EADv2 [4] writes that AI systems should “prioritize metrics of well-being in their design and use”. Different cultures, however, can have different views over the ways in which technologies can best serve mankind and its well-being. Confucianism-based cultures, for one, often do not draw a clear boundary between the self and the community, the individual and the collective, and the private and the public [47]. These cultural characteristics have a profound impact on what “well-being” signifies and how technologies may contribute to or compromise it.

In contemporary China, a community’s collective welfare is typically prioritized over an individual’s well-being, leading to Western criticisms of China’s abuse of human rights. Writing on “global AI ethics”, Hagerty and Rubinov [16] note how in countries such as Singapore and China, AI-driven surveillance technologies do not generate much controversy among citizens as state surveillance seems to be an “acceptable exchange for security and stability”. Such prioritization of societal harmony, Hung [47] argues, implies a paternalistic style of governance that is common in East Asia, where those occupying positions of

power are expected to guide their respective community as would parents for their children. According to Hung, it is for this reason that “collectively mediating technologies” implemented without full, collective consent are more accepted in Confucianism-based societies.

Bringing an East Asian perspective into discussions of AI ethics also enriches the ways in which one can conceptualize the relations between human and technology, and how such relations contribute to human flourishing. Reflecting on a Confucian “ritual technicity”, Wang [48] brings forth the ritual dimensions of artifacts that transcend their sheer practicality and examines how in *performing* (rather than merely *using*) technologies, humans are able to moralize themselves *with* artifacts. This intimate techno-human relationship implied in Confucian theories of self-cultivation aligns with the “embodiment relations” proposed by Don Ihde [49], according to which humans similarly *embody* technologies. In the Confucian context, the embodied, ritualized technologies become integral to pursuits of growth, wellness, and harmony with the world.

4.3 “Awareness of Misuse”

EADv2 emphasizes the need to minimize the risks of potential misuse of AI. While constructing a “Confucian Ethics of Technology”, Wong [19] probes into the concept of “harmony”, which we argue is essential when implementing responsible AI. It should be noted that the word “harmony” has more than once appeared as a separate principle in major guidelines for ethical AI published recently in China [50, 51]. We juxtapose this concept of “harmony” with the *Mepham Ethical Matrix* proposed by O’Neil and Gunn [52], which requires that one consider the interests of a range of stakeholders with reference to specific moral principles when designing AI. Bringing in a Confucian perspective is helpful in that it specifies how one may “harmonize” these diverging interests while preserving their irreducible differences.

Hongladarom [21] believes a Buddhist perspective can contribute to designing AI that can achieve both “ethical” and “technical” excellence. The author argues that when “harmonizing” the interests of diverse stakeholders, AI (and its manufacturer) must first consider the interest of others before their own, with the ultimate goal of relieving all beings of suffering. Here, any AI device that would cause suffering in “sentient beings” would be considered a case of misuse. This Buddhist vision of AI ethics is in line with Floridi’s definition of Information Ethics as an ecological ethics that seeks to ensure the “existence and flourishing of all entities and their global environment” with the goal of freeing them of “entropy”, a state “more fundamental than suffering” that refers to any form of “impoverishment of *being*” [46].

Just as the concept of “well-being” is culturally specific, the notion of “misuse” varies across contexts. As “cultural change” is described as a major threat to Japanese society, culturally trained robots are seen as a possible solution to this challenge and a way to conserve Japan’s assumed cultural continuity and homogeneity [34]. In this context, any AI system that “breaks” traditions and steers society away from its conservative social agenda may be viewed as a case of misuse.

5. MUSIC AI IN EAST ASIA: A SECTOR-SPECIFIC, CULTURALLY SENSITIVE APPROACH

Clancy [1] calls for a sector-specific (music ecosystem) application of AI ethics while encouraging researchers to consider contributions from non-Western traditions. We now attempt a sector-specific *and* culturally sensitive approach to thinking about ethical music AI in the context of East Asia, extending the principles analyzed in Section 4 to the domain of music.

5.1 “Human and Posthuman Rights”

According to the report “Ethics Guidelines for Trustworthy AI” (EGTAI), “AI systems need to be human-centric, resting on a commitment to their use in the service of humanity and the common good” [53]. A majority of AI ethics discussions are guided by such a human-centric frame. In this section, we move beyond this anthropocentric perspective to consider the environmental impact of “musicking”¹⁰ [54] in the age of AI. We first expand the notion of “human rights” so that it includes the rights of the deceased. We then bridge our work with the burgeoning field of “ecomusicology” to reflect on an AI-informed, political ecology of music [55].

In the case of music AI, Supertone, a South Korea-based music technology start-up, states in the “Ethical AI” section of their website that the firm “never monetize[s] any synthetic voice without the permission of the right holder” [56]. While the website does not further specify who may qualify as “right holders” across scenarios, we argue that we must think beyond the rights of *living* human beings. In the East Asian music industry, experiments are “reviving” deceased musicians: from the collaboration between virtual pop icon Teresa Teng with Taiwanese singer Jay Chou [57] to Big Hit Entertainment’s investment in Supertone to clone the voice of deceased South Korean superstars [58]. It is critical to reflect on the potential violation of “human” rights as well as the constant renegotiation of moral boundaries in such practices.

Section 4.1 establishes that it is necessary to consider both “human” and “*post*-human” rights when discussing AI ethics. De-centering the human resonates with the field of “ecomusicology”, which Titon [59] defines as “the study of music, culture, sound and nature in a period of environmental crisis”. Early efforts to connect human and non-human sound worlds came from soundscape studies and acoustic ecology, founded by Schafer with the *World Soundscape Project*. In “The Music of the Environment” [60], Schafer advocates the “recovery of positive silence”, arguing for the “reduction” rather than the “production” of sound. Turning to classical Chinese philosophies, in *Daodejing*, the fundamental text of Daoism, Laozi [61] writes that “the great note sounds *faint*”, promoting the “quiet” and the “silent” in music. Here, similarly, less is more. Meanwhile, Mozi, the founder of the philosophical school of Mohism, strongly condemns wasteful productions and performances of music [61]. The connection to

AI systems that “fart out” billions of songs just because they can is clear. Devine [55], for instance, illustrates how the carbon footprint of the music industry did not decrease in the age of streaming. We argue that such “ecomusicological” concerns become even more critical with the rapid advance of large, energy-consuming neural networks [62] and, in this context, AI-generated music.

From the ancient philosophies of Laozi and Mozi to the modern scholarship of Schafer, these stances can inspire imagining what “posthuman rights” may consist of in the context of music AI. Writing against artificial creativity, Mersch [63] addresses his concerns over how AI art may result in an overly crowded sonic and visual space marked by “overproduction”, “excess”, and the “more-than”. We thus argue that it is the responsibility of music AI developers to consider how their products impact the health of our soundscape in the middle of environmental crises. Meaningful attempts have been made by researchers using neural-network soundscapes to protect natural environments [64], or using AI to help one tune into island soundscapes to determine the level of seabird recovery [65].

5.2 Music and “Well-Being”

EADv2 refers to the Aristotelian concept of *eudaimonia*, “a practice that defines human well-being as the highest virtue for a society” [4]. While EADv2 does not explicitly address music AI and its tie to human well-being, included is a subsection titled “Affective Computing” that discusses issues related to emotion-like control in both humans and AI systems with a cross-cultural perspective. Considering the role of music as a culturally dependent regulator of emotions, one could propose that any AI music system be considered as a form of “affective computing” and should follow the guidelines detailed in this subsection of EADv2.

To help build an ethical application of AI to music, one that can foster well-being, it would be productive for AI developers to study and understand what makes a certain kind of music aesthetically pleasing and culturally appropriate in a particular musical ecosystem. In China, not all music is thought to carry the potential of contributing to “human flourishing”. “The Master [Confucius] said, ‘Find inspiration in the *Odes*, take your place through ritual, and achieve perfection with music’” [61]. This quote from the *Analects* of Confucius uncovers the essential place the trinity of poetry, ritual, and (ceremonial) music occupies in the growth of Confucian personhood. For Confucius, ritual music is fundamental to the moralization of mind, while entertainment music only corrupts – a claim that would not be unfamiliar to Plato. Today, this ancient link between music, morality, and well-being has in many ways become Chinese government’s rationale for music censorship as a way of promoting art and only “moral art” [66].

Finally, we argue that the propensity of East Asian societies to align AI with agendas of traditional culture preservation provides an important insight on how one may ethically deploy AI technology in these communities to maximize societal “harmony”. Šabanović [34] records

¹⁰ Christopher Small, in 1998, coined the term “musicking”, a verb that highlights music as a process (rather than an object) and that encompasses all musical activities from composing to performing to listening.

how robots in Japan were used to preserve *aizu bandaisan*, a Japanese folk dance, when there are no longer human inheritors to carry them out. Similar revivalist programs can be initiated with the design of AI systems that work to revitalize traditional repertoire. Such systems can potentially be used, for example, to help generate music for the hundreds of poems in the ancient text of the *Book of Odes*, the musical component of which is lost. This is the subject of much revivalist effort, and will require vast human labor.

5.3 Music, AI, and Cases of “Misuse”

To prevent potential misuses of AI technology, Clancy [1] argues the owners and designers of AI should make explicit statements about the intended consequences of these technologies. Taking into consideration the Confucian notion of “harmony” and the Buddhist concern for alleviating all sentient beings of suffering, AI researchers and designers might consider applying the *Mephram Ethical Matrix* [52] in order to “harmonize” the interests of different actors and, eventually, achieve holistic decision making.

According to Lamtharn (Hanoi) Hantrakul (a research scientist at TikTok/ByteDance in Shanghai, China), when designing *Tone Transfer* – a web app that uses Google’s machine learning AI to realize timbre conversion between different sound sources – he was met with the challenge of having to balance the interests of developers, target users, and – very importantly – those of local music practitioners as well as cultural insiders who might oppose having the unique timbre of their musical instruments taken out of context. In a private interview with us, Hantrakul describes the team’s decision to not include *guqin*, the ancient Chinese zither, in the application so as not to misrepresent the music instrument in front of an audience with limited knowledge about its original sound [67]. Hantrakul contrasts *Tone Transfer* with *Sounds of India*, an AI-powered app that transforms sounds into specific Indian music instruments. Hantrakul explains how the developers of that app were more confident to use those instruments because they knew the app would be used by communities already familiar with the sounds, hence reducing risks of misrepresentation and cultural appropriation.

What might other cases of misuse look like in the context of music AI, when factoring in the intellectual traditions reviewed so far? Not unlike the thoughts of Schafer, the Daoist tradition is likely to oppose an overly crowded soundscape that leaves no room for silence, as “only by relying on what is not there, do we have use of the room” [61]. Developers of music AI systems should in this sense pay attention to aspects of data ethics such as data management and “recycling”, so as to avoid flooding our already overloaded info- and sound-scape with algorithmically-generated music.¹¹ Similarly, the Mohist (that condemns wasteful music) and the Shinto tradition (that makes no categorical distinction between humans and their environments) will likely argue against AI systems that require too much computing power, embracing instead the idea of “green” music AI.

As addressed, for much of today’s Japanese society, any AI system that may disrupt its conservative social agenda

and cultural continuity would be viewed as a case of misuse. The same is true for music AI that may harm traditional art. It should be noted that creative experiments have been made by researchers to apply AI to the realm of traditional music, as in the case of “folk-rnn” [68], an AI system showing surprising success in generating plausible transcriptions in traditional dance music styles of Ireland and Scandinavia. These experiments, however, bring forth another set of questions regarding data ethics [69, 70, 71]: can any AI developer freely use and exploit materials in the public domain, which includes most of traditional music, that are not “protected” by copyright? In the cases that these materials are used in the training of a particular model, there should be much more conversation involving practitioners of such living musical traditions.

6. CONCLUSION

This paper contributes a discussion of several ethical dimensions of AI, and specifically AI applied to music, drawing in particular on non-Western philosophies such as Confucianism, Buddhism, Shintoism, and Daoism. In investigating such normative concepts in Confucian ethics as “personhood” and “harmony”, for instance, one may begin to reimagine the kind of relations humans may have with technologies (and thus enrich the existing framework established by Ihde in the 90s [49]). In juxtaposing these philosophical traditions with the critical perspectives of “posthumanism” and “ecomusicology” without disregarding their “irreducible differences”, we put into practice what Ess [6] advocates as “ethical pluralism”, while extending it to the less-visited domain of music AI.

To answer Clancy’s calls [1] for a sector-specific and culturally sensitive application of AI ethics, we take a close look in Section 5 at three ethical principles proposed in the IEEE Principles of Ethically Aligned Design and investigate their significances when applied to the music ecosystems of East Asia. Throughout, we ask what fresh perspectives researchers and practitioners of music AI today might gain by thinking beyond the dominant Western scientific knowledge systems that have been guiding our approaches to technological ethics. We begin such an experiment by turning to a number of influential East Asian philosophies. We recognize, however, that each of these philosophical traditions is extremely intricate and highly heterogenous within, and our discussions can only scratch the surface of this complex topic.

To date, such trans-cultural explorations are absent from existing work discussing ethics and MIR research in general [3]. Since East Asia is playing a leading role in the development and application of such technology [1], the perspectives forwarded in this paper are important to consider in order to draw culturally informed conclusions. This not only illuminates these issues for AI and music, but also applications of technology in general, and subtle differences in how established ethical principles can be (re)interpreted, e.g., “human rights”, “well-being”, and the potential “misuse” of AI technology.

¹¹ For instance, see Boomy (<https://boomy.com>).

7. ACKNOWLEDGEMENTS

This paper is an outcome of a project that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (Grant agreement No. 864189); Andre Holzapfel was supported by the Swedish Research Council (2019-03694) and the Marianne and Marcus Wallenberg Foundation (MMW 2020.0102).

8. REFERENCES

- [1] M. Clancy, “Reflections on the financial and ethical implications of music generated by artificial intelligence,” Ph.D. Dissertation, Trinity College, Dublin, 2021.
- [2] G. Born, Ed., *Music, Sound and Space: Transformations of Public and Private Experience*. Cambridge: Cambridge University Press, 2013.
- [3] A. Holzapfel, B. L. Sturm, and M. Coeckelbergh, “Ethical Dimensions of Music Information Retrieval Technology,” *Trans. Int. Soc. Music Information Retrieval*, vol. 1, no. 1, Art. no. 1, 2018.
- [4] The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems*, 2nd ed. IEEE, 2017.
- [5] A. Jobin, M. Lenca, and E. Vayena, “The global landscape of AI ethics guidelines,” *Nat. Mach. Intell.*, pp. 389–399, 2019.
- [6] C. Ess, “Ethical pluralism and global information ethics,” *Ethics and Information Technology*, vol. 8, no. 4, pp. 215–226, 2006.
- [7] M. Hamanaka, “Melody Slot Machine: A Controllable Holographic Virtual Performer,” *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 2468–2477, 2019.
- [8] Y.-J. Luo, K. W. Cheuk, T. Nakano, M. Goto, and D. Herremans, “Unsupervised disentanglement of pitch and timbre for isolated musical instrument sounds,” *Proceedings of the 21st International Society for Music Information Retrieval Conference ISMIR*, pp. 10–16, 2020.
- [9] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, “Deep Music Analogy Via Latent Representation Disentanglement,” *arXiv preprint arXiv:1906.03626*, 2019.
- [10] K. Watanabe and M. Goto, “A Chorus-section Detection Method for Lyrics Text,” *Proc. of the 21st Int. Society for Music Information Retrieval Conf.*, 2020.
- [11] S. Khoo, Z. Man, and Z. Cao, “Automatic Han Chinese Folk Song Classification Using Extreme Learning Machines,” in *AI 2012: Advances in Artificial Intelligence*, M. Thielscher and D. Zhang, Eds. Berlin, Heidelberg: Springer, 2012, pp. 49–60.
- [12] J. Kim, H. Choi, J. Park, M. Hahn, S. Kim, and J.-J. Kim, “Korean Singing Voice Synthesis Based on an LSTM Recurrent Neural Network,” *Proc. Interspeech 2018*, pp. 1551–1555, 2018.
- [13] G. Born, “Diversifying MIR: Knowledge and real-world challenges, and new interdisciplinary futures,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 193–204, 2020.
- [14] V. Dusek, *Philosophy of Technology: An Introduction*. Malden, MA; Oxford: Blackwell Publishing, 2006.
- [15] Yuk Hui, *The Question Concerning Technology in China: An Essay in Cosmotechnics*. Falmouth, United Kingdom: Urbanomic Media Ltd, 2016.
- [16] A. Hagerty and I. Rubinov, “Global AI Ethics: A Review of the Social Impacts and Ethical Implications of Artificial Intelligence,” *arXiv:1907.07892*, 2019.
- [17] H. R. Kirk, K. Lee, and C. Micallef, “The Nuances of Confucianism in Technology Policy: An Inquiry into the Interaction Between Cultural and Political Systems in Chinese Digital Ethics,” *International J. Politics, Culture, and Soc.*, 2020.
- [18] P.-H. Wong and T. X. Wang, Eds., *Harmonious Technology: A Confucian Ethics of Technology*. Abingdon, Oxon; New York, NY: Routledge, 2021.
- [19] P.-H. Wong, “Dao, Harmony, and Personhood: Toward a Confucian Ethics of Technology,” in *Harmonious Technology: A Confucian Ethics of Technology*, P.-H. Wong and T. X. Wang, Eds. Abingdon, Oxon; New York, NY: Routledge, 2021, pp. 10–28.
- [20] F. Rambelli, “Dharma Devices, Non-Hermeneutical Libraries, and Robot-Monks: Prayer Machines in Japanese Buddhism,” *J. Asian Humanities at Kyushu University*, no. 3, pp. 57–75, 2018.
- [21] S. Hongladarom, *The Ethics of AI and Robotics: A Buddhist Viewpoint*. Lanham: Lexington Books, 2020.
- [22] S. Hashimoto and K. Yabuno, *The Book of Wabot 2: The Dream of Making Robots*. Tokyo, Japan: Chuokoron-Shinsha, Inc., 2003.
- [23] T. D. Taylor, *Strange Sounds: Music, Technology and Culture*. New York: Routledge, 2001.
- [24] C. B. Jensen and A. Blok, “Techno-animism in Japan: Shinto Cosmograms, Actor-network Theory, and the Enabling Powers of Non-human Agencies,” *Theory, Culture & Society*, vol. 30, no. 2, pp. 84–115, 2013.
- [25] B. Allen, “A Dao of Technology?,” *Dao: A Journal of Comparative Philosophy*, vol. 9, pp. 151–160, 2010.
- [26] B. Ziporyn, “The Self-So and Its Traces in the Thought of Guo Xiang,” *Philosophy East and West*, vol. 43, no. 3, pp. 511–539, 1993.
- [27] J. Needham, *Science in Traditional China: A Comparative Perspective*. Cambridge, Massachusetts: Harvard University Press, 1981.

- [28] E. S. Nelson, “Technology and the Way: Buber, Heidegger, and Lao-Zhuang ‘Daoism,’” *J. Chinese Philosophy*, vol. 41, no. 3–4, pp. 307–327, 2014.
- [29] Nature Research Custom, “Traditional medicine meets artificial intelligence,” *Nature Portfolio*, p. 10, Mar. 2019.
- [30] Y. Guo, X. Ren, Y. Chen, and T. Wang, “Artificial Intelligence Meets Chinese Medicine,” *Chinese J. Integrative Medicine*, vol. 25, pp. 648–653, 2019.
- [31] andyrobot, “2047 APOLOGUE: Live performance collaboration with Zhang Yimou,” *andyrobot.com*, 2020. Accessed: Apr. 16, 2021. [Online]. Available: <https://www.andyrobot.com/2047-apologue>
- [32] R. R. Nair, “Mindar The Robot Teaches Heart Sutra At Japan’s Buddhist Temple,” *International Business Times*, Aug. 20, 2019. [Online]. Available: <https://bit.ly/37dbjhu>
- [33] Y. Fukunaga, “AI can help sustain traditional crafts,” *The Japan Times*, Mar. 13, 2020. Accessed: Apr. 16, 2021. [Online]. Available: <https://bit.ly/3f5N73X>
- [34] S. Šabanović, “Inventing Japan’s ‘robotics culture’: The repeated assembly of science, technology, and culture in social robotics,” *Social Studies of Science*, vol. 44, no. 3, pp. 342–367, 2014.
- [35] B. S. McDougall and A. Hansson, Eds., *Chinese Concepts of Privacy*. Leiden, Boston, and Köln: Brill, 2002.
- [36] R. Barthes, “The Death of the Author,” in *The Norton Anthology of Theory and Criticism*, 2nd ed., New York: W.W. Norton & Company, Inc., 2010, pp. 1322–1326.
- [37] J. Derrida, “Structure, Sign and Play in the Discourse of the Human Sciences,” in *Writing and Difference*, Chicago: University of Chicago Press, 1978, pp. 278–293.
- [38] K.-H. Chen, *Asia as Method: Toward Deimperialization*. Durham and London: Duke University press, 2010.
- [39] A. A. An-Na’im., Ed., *Human Rights in Cross-Cultural Perspectives: A Quest for Consensus*. Philadelphia: University of Pennsylvania Press, 2010.
- [40] W. P. Alford, “Making a Goddess of Democracy from Loose Sand: Thoughts on Human Rights in the People’s Republic of China,” in *Human Rights in Cross-Cultural Perspectives: A Quest for Consensus*, A. A. An-Na’im., Ed. Philadelphia: University of Pennsylvania Press, 2010.
- [41] T. Cassauwers, “How Confucianism Could Put Fears about Artificial Intelligence to Bed,” *OZY*, Mar. 27, 2019. Accessed: Apr. 15, 2021. [Online]. Available: <https://bit.ly/3txWGxJ>
- [42] The Ethics Committee, JSAI, “The Japanese Society for Artificial Intelligence Ethical Guidelines.” 2017. Accessed: Apr. 23, 2021. [Online]. Available: <https://bit.ly/3bfGLy6>
- [43] R. Braidotti, “Posthuman Critical Theory,” *J. Posthuman Studies*, vol. 1, no. 1, pp. 9–25, 2017.
- [44] C. M. Sjöberg, “The Digital Person—A New Legal Entity? On the Role of Law in an AI-Based Society,” in *Legal Tech and the New Sharing Economy*, M. C. Compagnucci, N. Forgó, T. Kono, S. Teramoto, and E. P. M. Vermeulen, Eds. Singapore: Springer, pp. 81–91.
- [45] J. E. Lewis, Ed., *Indigenous Protocol and Artificial Intelligence Position Paper*. Honolulu, Hawai‘i: The Initiative for Indigenous Futures and the Canadian Institute for Advanced Research (CIFAR), 2020.
- [46] L. Floridi, “Foundations of Information Ethics,” in *The Handbook of Information and Computer Ethics*, 2008, pp. 1–23.
- [47] C. Hung, “Technological Mediation In and For Confucianism-Based Cultures,” in *Harmonious Technology: A Confucian Ethics of Technology*, P.-H. Wong and T. X. Wang, Eds. Abingdon, Oxon; New York, NY: Routledge, 2021, pp. 50–65.
- [48] T. X. Wang, “Confucian Ritual Technicity and Philosophy of Technology,” in *Harmonious Technology: A Confucian Ethics of Technology*, P.-H. Wong and T. X. Wang, Eds. Abingdon, Oxon; New York, NY: Routledge, 2021, pp. 10–28.
- [49] D. Ihde, *Technology and the lifeworld: From garden to earth*. Bloomington: Indiana University Press, 1990.
- [50] L. Zhang, “China: AI Governance Principles Released,” *Library of Congress*, Sep. 09, 2019. <https://www.loc.gov/item/global-legal-monitor/2019-09-09/china-ai-governance-principles-released/>
- [51] Beijing Academy of Artificial Intelligence (BAAI), “Beijing AI Principles,” *baai.ac.cn*, May 25, 2019. <https://www.baai.ac.cn/news/beijing-ai-principles-en.html>
- [52] C. O’Neil and H. Gunn, “Near-Term Artificial Intelligence and the Ethical Matrix,” in *Ethics of Artificial Intelligence*, S. M. Liao, Ed. New York: Oxford University Press, 2020.
- [53] High-Level Expert Group on Artificial Intelligence (AI HLEG), “Ethics guidelines for trustworthy AI.” European Commission, 2019. Accessed: May 10, 2021. [Online]. Available: <https://bit.ly/3uA6N6N>
- [54] C. Small, *Musicking: the meanings of performing and listening*. Hanover: University Press of New England, 1998.
- [55] K. Devine, *Decomposed: The Political Ecology of Music*. Cambridge, Massachusetts: MIT Press, 2019.

- [56] Supertone, “Ethical AI,” *supertone.ai*. <https://supertone.ai/eng/synthesis/synthesis.php?focus=ai> (accessed Jul. 31, 2021).
- [57] L. Hu, “AI-Powered Digital People,” *Synced: AI Technology & Industry Review*, Apr. 11, 2020. [Online]. Available: <https://bit.ly/3bbyv1Z>
- [58] M. Stassen, “Big Hit Invests \$3.6M in Supertone, an AI Firm that Just Cloned a Dead Superstar’s Voice,” *Music Business Worldwide*, Feb. 25, 2021. [Online]. Available: <https://bit.ly/3vSJ1mE>
- [59] J. T. Titon, “The Nature of Ecomusicology,” *Música e Cultura*, vol. 8, no. 1, pp. 8–18, 2013.
- [60] R. M. Schafer, “The Music of the Environment,” in *Audio Culture: Readings in Modern Music*, C. Cox and D. Warner, Eds. New York and London: The Continuum International Publishing Group Inc, 2006, pp. 29–39.
- [61] B. W. Van Norden and P. J. Ivanhoe, Eds., *Readings in Classical Chinese Philosophy*, 2nd ed. Indianapolis: Hackett Publishing Company, Inc., 2005.
- [62] E. Strubell, A. Ganeish, and A. McCallum, “Energy and policy considerations for deep learning in NLP,” presented at the The 57th Annual Meeting of the Association for Computational Linguistics (ACL), 2019.
- [63] D. Mersch, “(Un)creative Artificial Intelligence: A Critique of ‘Artificial Art,’” *ResearchGate*, 2020, doi: 10.13140/RG.2.2.20353.07529.
- [64] B. Yirka, “Using neural-network soundscapes to protect natural environments,” *Phys.org*, Jul. 17, 2020. Accessed: Apr. 27, 2021. [Online]. Available: <https://bit.ly/3ezvnPB>
- [65] J. R. Learn, “Artificial intelligence helps researchers tune into island soundscapes,” *The Wildlife Society*, Dec. 20, 2019. Accessed: Apr. 27, 2021. [Online]. Available: <https://bit.ly/3ty05fZ>
- [66] P. Li and A. Jourdan, “China takes aim at hip-hop, saying ‘low-taste content’ must stop,” *Reuters*, Jan. 22, 2018. Accessed: Apr. 28, 2021. [Online]. Available: <https://reut.rs/33AsjMw>
- [67] L. (Hanoi) Hantrakul, “Interview with Rujing Huang (Online),” Jul. 09, 2021.
- [68] B. L. T. Sturm and O. Ben-Tal, “Folk the Algorithms: (Mis)Applying Artificial Intelligence to Folk Music,” in *Handbook of Artificial Intelligence for Music*, E. R. Miranda, Ed. Springer, 2021.
- [69] A. Seeger, “I found it, how can I use it? Dealing with the ethical and legal constraints of information access,” *Proc. Int. Symp. Music Info. Retrieval Conf.*, 2003.
- [70] E. Drott, “Copyright, compensation, and commons in the music AI industry,” *Creative Industries Journal*, pp. 1–18, Oct. 2020, doi: 10.1080/17510694.2020.1839702.
- [71] B. L. Sturm, M. Iglesias, O. Ben-Tal, M. Miron, and E. Gómez, “Artificial Intelligence and Music: Open Questions of Copyright Law and Engineering Praxis,” *MDPI Arts*, vol. 8, no. 3, Art. no. 3, 2019.

A BENCHMARKING INITIATIVE FOR AUDIO-DOMAIN MUSIC GENERATION USING THE FREESOUND LOOP DATASET

Tun-Min Hung^{1,2}

Bo-Yu Chen¹

Yen-Tung Yeh^{1,2}

Yi-Hsuan Yang^{1,3}

¹ Academia Sinica, ² National Taiwan University, ³ Taiwan AI Labs

r09946015@ntu.edu.tw, bernie40916@gmail.com, b06611042@ntu.edu.tw, yang@citi.sinica.edu.tw

ABSTRACT

This paper proposes a new benchmark task for generating musical passages in the audio domain by using the drum loops from the FreeSound Loop Dataset, which are publicly re-distributable. Moreover, we use a larger collection of drum loops from Looperman to establish four model-based objective metrics for evaluation, releasing these metrics as a library for quantifying and facilitating the progress of musical audio generation. Under this evaluation framework, we benchmark the performance of three recent deep generative adversarial network (GAN) models we customize to generate loops, including StyleGAN, StyleGAN2, and UNAGAN. We also report a subjective evaluation of these models. Our evaluation shows that the one based on StyleGAN2 performs the best in both objective and subjective metrics.

1. INTRODUCTION

Audio-domain music generation involves generating musical sounds either directly as audio waveforms or as time-frequency representations such as the Mel spectrograms. Besides modeling musical content in aspects such as pitch and rhythm, it has the additional complexity of modeling the spectral-temporal properties of musical sounds, compared to its symbolic-domain music generation counterpart. In recent years, deep learning models have been proposed for audio-domain music generation, starting with simpler tasks such as generating instrumental single notes [1–4], a task also known as *neural audio synthesis*. Researchers have also begun to address the more challenging setting of generating sounds of longer duration [5–12]. For example, Jukebox [11] aims to generate realistic minutes-long singing voices conditioned on lyrics, genre, and artists; and UNAGAN [9] aims to generate musical passages of finite yet arbitrary duration for singing voices, violin, and piano, in an unconditional fashion.

The focus of this paper is on the evaluation of audio-domain music generation. We note that, for model training

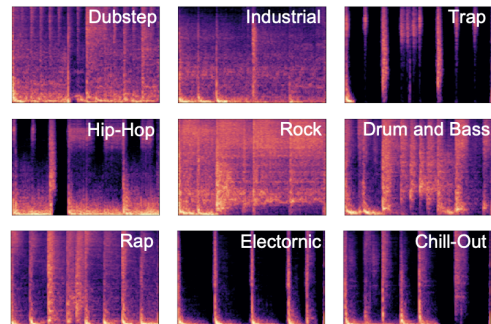


Figure 1. The mel-spectrograms of some random drum loops generated by the StyleGAN2 model [13] trained on the looperman dataset, with the genre labels predicted by the short-chunk CNN [14] classifier (see Section 4.1).

and evaluation, research on generating single notes quite often adopts NSynth [1], a large public dataset consisting of individual notes from different instruments. The use of a common dataset for evaluation ensures the validity of performance comparison between different models. Such a standardized dataset for benchmarking, however, is not available when it comes to generating longer musical passages, to our best knowledge. Oftentimes private in-house datasets are employed in existing works; for example, both UNAGAN [9] and Jukebox [11] employ audio recordings scrapped from the Internet, which cannot be shared publicly. The only exception is MAESTRO, a public dataset with over 172 hours of solo piano performances, employed by MelNet [8], UNAGAN [9], and MP3net [12]. However, MAESTRO is piano-only so not diverse enough in timbre.

We see new opportunities to address this gap with the recent release of the FreeSound Loop Dataset (FSLD) [15], which contains 9,455 production-ready, public-domain loops distributed under Creative Commons licenses.¹ We therefore propose to use *audio-domain loop generation*, a task seldom reported in the literature, to set a benchmark for musical audio generation research.

We deem loops as an adequate target for audio generation for their following merits. First, loops are audio excerpts, usually of short duration, that can be played in seamless manner [15, 16]. Hence, the generated loops can be played repeatedly. Second, loops are fundamental units

¹ <https://zenodo.org/record/3967852>

in the production of many contemporary dance music genres. A loop can usually be associated with a single genre or instrument label [15], and a certain “role” (e.g., percussion, FX, melody, bass, chord, voice) [17]. Third, loops are fairly diverse in their music content and timbre, as sound design has been a central part in making loops.

A primary contribution of this paper is therefore the proposal and implementation of using FSLD as a benchmark for audio generation. In particular, we adapt three recent deep generative adversarial network (GAN) [18] models and train them on the *drum-loop* subset of FSLD, and report thorough evaluation of their performance, both objectively and subjectively. This includes UNAGAN [9] and two state-of-the-art models for image generation, StyleGAN [19] and StyleGAN2 [13].

Drum loop generation is interesting in its own right due to its applications in automatic creation of loop-based music [20]. As [21] indicates, drum beats represent one of the most critical and fundamental elements that form the style of EDM. Moreover, drum loops are already fairly diverse in musical content, as demonstrated in Figure 1. Although we only consider drum loops here for the sake of simplicity, this benchmark can be easily extended to cover all the loops from FSLD in the near future.

Our secondary contribution lies in the development of standardized objective metrics for evaluating audio-domain loop generation, which can be equally important as having a standardized dataset. We collect a larger drum loop dataset from an online library called *looperman*,² with roughly 9 times more drum loops than FSLD, and use this *looperman* dataset to build four model-based metrics (e.g., inception score [22])³ to evaluate the acoustic *quality* and *diversity* of the loops generated by the GAN models. While this *looperman* dataset cannot be released publicly due to copyright concerns, we release the metrics and the trained GAN models for drum loop generation at the following GitHub repo: <https://github.com/allenhung1025/LoopTest>.

Moreover, we put some of the generated drum loops on an accompanying **demo website**,⁴ which we recommend readers to visit and listen to. We also present the result where we use the method of *style-mixing* of StyleGAN2 to generate “interpolated” versions of loops.

Below, we review related work in Section 2, present the datasets in Section 3, the proposed objective metrics in Section 4, the benchmarked models in Section 5, and the evaluation result in Section 6.

2. RELATED WORK

Existing work on audio-domain music generation can be categorized in many ways. First, an **unconditional** audio generation model takes as input a vector $\mathbf{z} \in \mathbb{R}^{N_z}$ of a fixed number of random variables (or a sequence of such vectors; see below) and generates an audio piece from scratch.

² <https://www.looperman.com/>

³ We refer to them as *model-based* metrics because we need to build a classifier or a clustering model to calculate the metrics; see Section 4.

⁴ <https://loopgen.github.io/>

When side information of the target audio to be generated is available, we can feed such prior information as another vector $\mathbf{c} \in \mathbb{R}^{N_c}$ and use it as an additional input to the generative model, making it a **conditional** generation model. For example, GANSynth [2] uses the pitch of the target audio as a condition. While we focus on unconditional generation in our benchmarking experiments presented in Section 6, it is straightforward to extend all the models presented in Section 5 to take additional conditions.

Second, some existing models can only generate **fixed-length** output, while others can do **variable-length** generation. One approach to realize variable-length generation is by using as input to the generative model a sequence of latent vectors $\mathbf{z}_1, \mathbf{z}_2, \dots$, instead of just one latent vector \mathbf{z} . This is the approach taken by UNAGAN [9], Jukebox [11], and VQCPC-GAN [23].

Third, existing models for generating single notes are typically **non-autoregressive** models [2–4], i.e., the target is generated at one shot. When it comes to generating longer phrases, **autoregressive** models, that generate the target piece one frame or one time sample at a time in the chronological order, might perform better [5, 11], as the output of such models depends explicitly on the previous frames (or samples) that have been generated.

Existing models have been trained and evaluated to generate different types of musical audio, including singing voice [9–11], drum [3, 4, 24, 25], violin [9], and piano [5, 8, 9, 12]. The only work addressing loop generation is the very recent LoopNet model from Chandna *et al.* [26]. They also use loops from *looperman* but not anything from FSLD or other public datasets, hence not constituting a benchmark for audio generation.

For drum generation in particular, work has been done in the symbolic domain to generate drum patterns [27–30] and a drum track as part of a symbolic multi-track composition [31–33]. For example, DeepDrummer [29] employs human-in-the-loop to produce drum patterns preferred by a user. In the audio domain, DrumGAN [3] and the model proposed by Ramires *et al.* [4] both work on only single hits, i.e., one-shot drum sounds. They both use the Audio Commons models [34] to extract high-level timbral features to condition the generation process. DrumNet [35] is a model that generates a sequence of (monophonic) kick drum hits, not the sounds of an entire drum kit.

3. DATASETS

Two datasets are employed in this work. The first one is a subset of drum loops from the public dataset FSLD [15], which is used to train the generative models for benchmarking. FSLD comes with detailed manual labeling of the loops with tags such as instrumentation, rhythm, tone and genre. As stated in the FSLD paper [15], FSLD is balanced in terms of musical genre. By picking loops which are tagged with the keywords “drum”, “drums” or “drum-loop”, we are able to find 2,608 drum loops out of the 9,455 loops available in FSLD. We do not need to hold out any of them as test data but use all these bars for training our generative models, since we focus on unconditional generation

in this paper; i.e., each generative model will generate a set of loops randomly for evaluation.

The second dataset is a larger, private collection of drum loops we collect from *looperman*, a website hosting free music loops.⁵ We are able to collect in total 23,983 drum loops, which is much more than the drum loops in FSLD. We use the *looperman* dataset mainly for establishing the model-based objective metrics for evaluation (see Section 4). For instance, we train an audio-based genre classifier using *looperman* to set up the drum-loop version of the “inception score” [22, 36] to measure how likely a machine-generated loop sounds like a drum loop. Figure 2 shows the number of tracks per genre tag in *looperman*, which exhibits a typical long-tail distribution. We can see that “Trap” is the most frequent genre, with 5,903 loops.

We use *looperman* instead of FSLD to set up such objective metrics, since a larger dataset increases the validity and generalizability of the metrics. Moreover, although we cannot re-distribute the loops from *looperman* according to its terms, we can share checkpoints of the pre-trained models for computing the proposed objective metrics.

3.1 Data Pre-processing

As we are interested in benchmarking the performance of one-bar loop generation, we perform downbeat tracking using the state-of-the-art recurrent neural network (RNN) model available in the *Madmom* library [37, 38] to slice every audio file into multiple one-bar loops.⁶ After this processing, we have in total 13,666 and 128,122 one-bar samples from FSLD and *looperman*, respectively. We refer to these two collections of one-bar drum loops as the **freesound** and **looperman** datasets hereafter. We note that all these one-bar samples are of four beats.

As shown in Figure 3, the one-bar samples in either the *freesound* or *looperman* datasets have different tempos and hence different lengths. To unify their length to facilitate benchmarking, we use *pyrubberband*⁷ to temporally stretch each of them to 2-second long, namely to have 120 BPM (beat-per-minute) as their tempo. We listened to some of the stretched samples in both datasets and found most sounded plausible with little perceptible artifacts.⁸

All the loops are in 44,100 Hz sampling rate. We down-mix the stereo ones into mono. After that, we follow the setting of UNAGAN [9] to compute the Mel spectrograms of these samples, with 1,024-point window size hann window and 275-point hop size for short-time Fourier Transform (STFT), and 80 Mel channels.

⁵ As stated on <https://www.looperman.com/help/terms>, “All samples and loops are free to use in commercial and non commercial projects.” But, “You may NOT use or re-distribute any media from the loops section of *looperman.com* as is either for free or commercially on any other web site.” (Accessed August 1, 2021)

⁶ The downbeat tracker in *Madmom* is fairly accurate for percussive audio such as the drum loops. For example, it reaches F1-score of 0.863 on the Ballroom dataset [39], according to [38].

⁷ <https://pypi.org/project/pyrubberband/>

⁸ This, however, may not be the case if the loops are not drum loops. Some data filtering might be needed then, e.g., to remove those whose tempo are much away from 120 BPM.

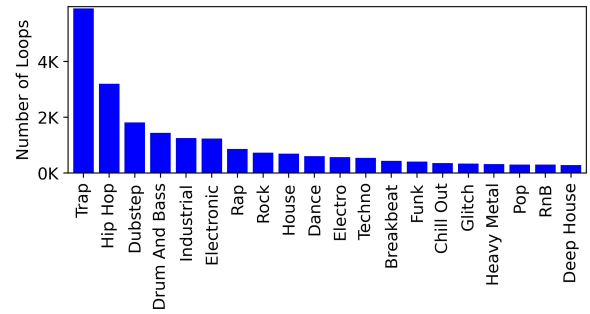


Figure 2. Genre distribution of the drum loops from *looperman*; we display only the top 20 out of 66 genres.

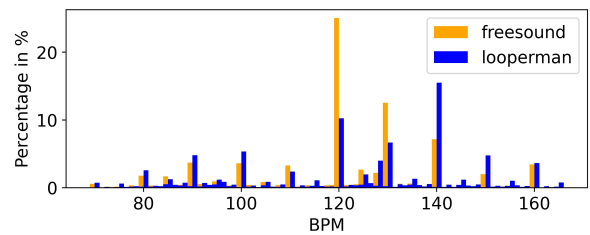


Figure 3. Tempo distribution of the two sets of loops. Y axis represents the percentage of all loops in the dataset.

4. EVALUATION METRICS

We consider four metrics in our benchmark, developing the drum-loop version of them using the *looperman* dataset.

4.1 Inception Score (IS)

IS [22, 36] measures the quality of the generated data and detects whether there is a mode collapse by using a pre-trained domain-specific classifier. It is computed as the KL divergence between the conditional probability $p(y|x)$ and marginal probability $p(y)$,

$$IS = \exp \left(E_x [\text{KL}(p(y|x) || p(y))] \right), \quad (1)$$

where $x \in \mathcal{X}$ denotes a data example (e.g., a generated loop), and $y \in \mathcal{Y}$ is a pre-defined class. Specifically, the calculation of IS involves building a classifier over the type of data of interest, and it achieves high score (namely, the *higher* the better) score when 1) each of the generated data can be classified to any of the predefined classes with high confidence, and 2) the generated data as a whole has close to uniform distribution over the predefined classes.

We use *looperman* to establish such a classifier, using its genre labels for training a 66-class classifier over the Mel spectrograms of one-bar samples. Specifically, we split the data by 100,000/10,000/18,111 as the training, validation, and test sets, and use the state-of-the-art music auto-tagging model *short-chunk CNN* [14]⁹ for model training. The classifier achieves 0.748 accuracy on the test set.

4.2 Fréchet Audio Distance (FAD)

The idea of FAD, as proposed by Kilgour *et al.* [40], is to measure the closeness of the data distribution of the real

⁹ github.com/minzwon/sota-music-tagging-models

data versus that of the generated data, in a certain embedding space. Specifically, they pre-train a VGGish-based audio classifier on a large collection of YouTube videos for classifying 300+ audio classes and sound events, and then use the second last 128-dimension layer (i.e., prior to the final classification layer) for this embedding space [40]. The data distributions of *real* and *generated* data in this space are modeled as a multi-variate normal distribution characterized by (μ_r, Σ_r) and (μ_g, Σ_g) respectively. The FAD score is then computed by the following equation,

$$\text{FAD} = \|\mu_r - \mu_g\|^2 + \text{tr}(\Sigma_r + \Sigma_g - 2\sqrt{\Sigma_r \Sigma_g}), \quad (2)$$

and is the *lower* the better (down to zero). We use the open source code and pre-trained classifier¹⁰ to compute the FAD, using the looperman data as the real data and the output of a generative model as the generated data.

4.3 Diversity Measurement

Following [9], we measure diversity with the number of statistically-different bins (NDB) and Jensen-Shannon divergence (JSD) metrics proposed by Richardson *et al.* [41], via the official open source code.¹¹ We firstly run K -means clustering over normalized Mel spectrograms of 10 thousands one-bar samples randomly picked from looperman to get $K = 100$ clusters, and count the number of samples per cluster, n_k , for each k . Then, given a collection of loops randomly generated by a generative model, we fit the loops into the clustering and also count the number of fitted samples per cluster, \widehat{n}_k . We can then measure the difference between the two distributions $\{n_k\}$ and $\{\widehat{n}_k\}$ by either the number of statistically-different bins (among the K bins; the *lower* the better) and their JSD (the *lower* the better; down to zero). Richardson *et al.* [41] recommend reporting the value of NDB divided by K , saying that if the two samples do come from the same distribution, NDB/K should be equal to the significance level of the statistical test, which we set to 0.05.

5. BENCHMARKED GENERATIVE MODELS

We develop and evaluate in total three recent deep generative models, all of which happen to be GAN-based [18]. The first model is **StyleGAN2** [13], which represents the state-of-the-art in image generation, included here intending to test its applicability for musical audio generation (which has not been reported elsewhere, to our best knowledge). The second model, **StyleGAN** [19], is a precursor of StyleGAN2, tested on spoken digit generation before (akin to single note generation in music) [43] but not on musical audio generation. Both StyleGAN and StyleGAN2 generate only fixed-length output, which is fine here since our samples have constant length. The last model, **UNAGAN** [9], represents a state-of-the-art in musical audio generation, capable of generating variable-length output. For fair comparison, we only require UNAGAN to

generate two-second samples as the other two. Schematic plots of the three models can be found in Figure 4.

All these three models are trained to generate Mel spectrograms, with phase information missing. But, the Mel spectrograms can later be converted into audio waveforms by a separate *neural vocoder*, such as WaveNet [5], WaveGlow [44], DiffWave [45], or MelGAN [46]. We are in favor of MelGAN for it is non-autoregressive and therefore fast in inference time, and for there is official open source code that is easy to use.¹² We train MelGAN on the looperman dataset and use it in all our experiments.

5.1 StyleGAN

StyleGAN and StyleGAN2 are both non-autoregressive models for generating images. They take a constant tensor of size $4 \times 4 \times 512$ as input, and use a mapping network $f(\cdot)$ consisting of eight linear layers to map a random latent vector \mathbf{z} to an intermediate latent vector \mathbf{w} , which affects the generation process by means of adaptive instance normalization (AdaIN) operations in every block of the generator [19]. Each block progressively upsamples its input to a larger tensor, until reaching the target size of 1024×1024 by the end with in total eight such blocks.

The input tensor of StyleGAN and StyleGAN2 can be interpreted as $512 \times 4 \times 4$ tiny images. This tensor is learned and then fixed during the inference stage while generating new images, using different \mathbf{z} each time. We modify it to be a $5 \times 20 \times 512$ tensor in our work, to generate a 80×320 Mel spectrogram through four upsampling blocks.

Our implementation of StyleGAN is based on an open source code.¹³ For model training, StyleGAN employs the non-saturating loss with R_1 regularization [47] and a progressive-growing training strategy [42]. We use 0.9 mixing regularization ratio [42], and set the batch size to 32, 16, 8, 4 in the respective scale, from low to high resolution. In every scale, we train with 1.2M samples. We deployed Adam optimization algorithm and set the learning rate to $1e-3$. The total training time is 120 hr on an NVIDIA GTX1080 GPU with 8GB memory.

5.2 StyleGAN2

StyleGAN2 [13] is an improved version of StyleGAN with many structural changes, including replacing AdaIN by a combination of “modulation” and “demodulation” layers, processing the input tensor differently, adding the Gaussian noise outside of the style blocks etc. The weights in the 3×3 convolution layers are scaled with $f(\mathbf{z})$ in the Modulation block and normalized by L2 norm in the DeModulation block. We refer readers to the original paper [13] for details. Our implementation of StyleGAN2 is based on another open source code,¹⁴ with similar training strategies as the StyleGAN case, but two times larger learning rate, no progressive growing, and a constant batch size of 8 for 1M samples. The total training time is 100 hr on a GTX1080.

¹⁰ github.com/google-research/google-research/tree/master/frechet_audio_distance

¹¹ github.com/eitanrich/gans-n-gmms

¹² github.com/descriptinc/melgan-neurips

¹³ github.com/rosinality/style-based-gan-pytorch

¹⁴ github.com/rosinality/stylegan2-pytorch

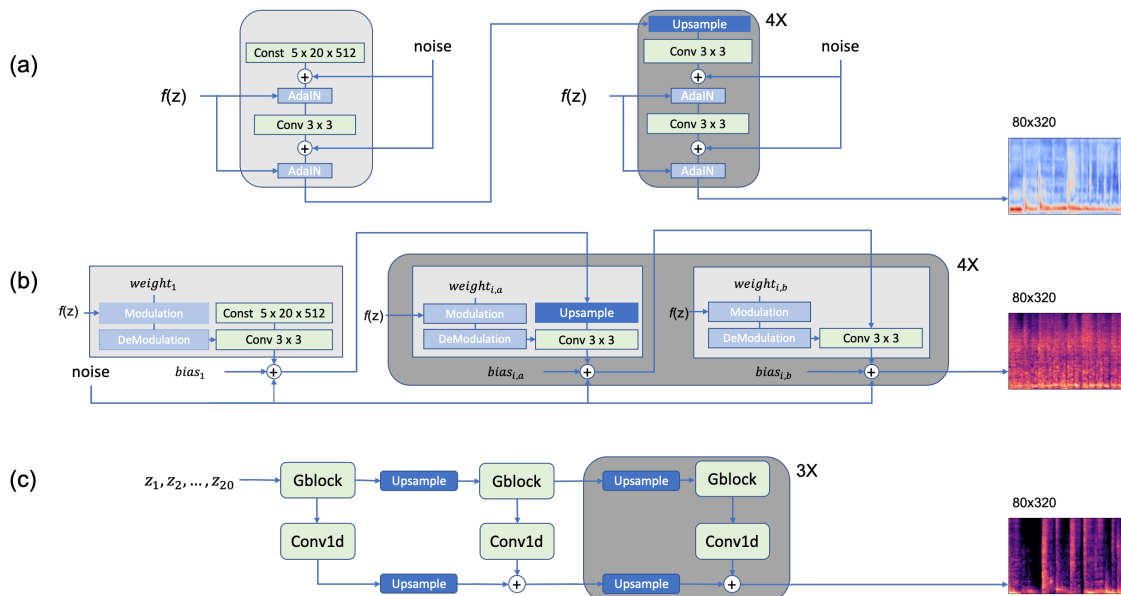


Figure 4. Schematic plots of the adapted (a) StyleGAN [19], (b) StyleGAN2 [42], and (c) UNAGAN [9] in our benchmark. Only a single latent vector \mathbf{z} is used as the input for the fully convolutional models in (a) and (b), while a sequence of 20 latent vectors are used in model (c), which uses a stack of gate recurrent unit (GRU) layer and grouped convolution layer in each of its ‘Gblocks’ [9, 10]. In (b), $weight_i$ and $bias_i$ are parameters of the 3×3 convolution layers to be learned.

5.3 UNAGAN

UNAGAN [9] is a non-autoregressive model originally designed for generating variable-length singing voices in an unconditional fashion. The authors also demonstrate its effectiveness in learning to generate passages of violin, piano, and speech. What makes UNAGAN different from existing models such as StyleGAN, WaveGAN [48], DrumGAN [3], and GANSynth [2] is that UNAGAN takes a sequence of latent vectors $\mathbf{z}_1, \mathbf{z}_2, \dots$ as input, instead of just a single one. This sequence of latent vectors, together with the recurrent units inside its ‘Gblocks’ [9, 10] (see Figure 4(c)), facilitates UNAGAN to generate variable-length audio with length proportional to the length of the input latent sequence. UNAGAN adopts a hierarchical architecture that generates Mel spectrograms in a coarse-to-fine fashion similar to the progressive upsampling blocks in StyleGAN and StyleGAN2. UNAGAN uses the BEGAN-based adversarial loss [49], and an additional cycle consistency loss [50] to stabilize training and for increasing diversity. Our implementation of UNAGAN is based on the official open source code.¹⁵ We fix the number of input latent vectors to 20 and train the model with Adam, $1e-4$ learning rate, and a batch size of 16 for 100k iterations, amounting to 40 hr on a GTX1080.

6. EVALUATION

6.1 Objective Evaluation Result

Table 1 presents the objective evaluation result of models trained on the *freesound* dataset. Each model generates 2,000 random loops to compute the scores. We also compute these metrics on the two real datasets and add

¹⁵ <https://github.com/ciaua/unagan>

the results to Table 1, to offer an oracle reference. We see that the IS of StyleGAN2 is the closest to that of the *freesound* dataset, followed by UNAGAN and then StyleGAN. Student’s *t*-test shows that the performance edge of StyleGAN2 over either UNAGAN or StyleGAN is statistically significant (p -value <0.01). This reveals the efficacy of StyleGAN2 for generating fixed-length audio.

The scores in JS and NDB further support the superiority of StyleGAN2, showing that its output is the most diverse among the three.

The scores in FAD, however, shows that UNAGAN performs better than StyleGAN2 here. The contrast between IS and FAD suggests that UNAGAN learns to generate samples whose embeddings have similar distribution as the real data, but its output cannot be easily associated with a genre class by the short-chunk CNN classifier. We also see that StyleGAN has fairly high FAD, showing that its generation hardly resemble the real data distribution.

Out of curiosity, we also train the models on the private, yet larger, *looperman* dataset and redo the evaluation. Table 2 shows that StyleGAN2 achieves even higher IS and much lower NDB here. Furthermore, its FAD is now lower than that of UNAGAN. Together with the result in JS and NDB, we see from this table that StyleGAN2 is more effective in learning to cover the modes in a large dataset. Figure 1 demonstrates the mel-spectrograms of some random drum loops generated by this StyleGAN2 model.

6.2 Subjective Evaluation & Its Result

We run additionally an online listening test to evaluate the models subjectively. Each subject is presented with the a randomly-picked human-made loop from the *freesound* dataset, and one randomly-generated loop by each of the

	IS \uparrow	FAD \downarrow	JS \downarrow	NDB/K \downarrow
Looperman	11.9 \pm 3.21	0.11	0.01	0.01
Freesound	6.30 \pm 1.82	0.72	0.08	0.46
StyleGAN	1.31 \pm 1.95	13.78	0.43	0.94
StyleGAN2	5.24\pm1.84	7.91	0.09	0.59
UNAGAN	3.33 \pm 1.65	4.32	0.16	0.73

Table 1. Objective evaluation result for the three models trained on the *freesound* dataset. We also display the IS of the two sets of real data. (\downarrow/\uparrow : the lower/higher the better).

	IS \uparrow	FAD \downarrow	JS \downarrow	NDB/K \downarrow
StyleGAN	1.30 \pm 2.00	12.98	0.41	0.87
StyleGAN2	6.08\pm2.26	2.22	0.01	0.08
UNAGAN	3.83 \pm 1.72	3.36	0.29	0.89

Table 2. Objective evaluation result for the three models trained on instead the private *looperman* dataset.

three models trained on *freesound*, with the ordering of these four loops randomized. Then, the subject is asked to rate each of these one-bar loops in terms of the following metrics, the first three on a three-point scale, and the last one on a five-point Likert scale:

- **Drumness:** whether the sample contains drum sounds ('no'/'yes but vague'/'yes and clear');
- **Loopness:** whether the sample can be played repeatedly in a seamless manner ('no'/'yes but not so good'/'yes');
- **Audio quality:** whether the sample is free of unpleasant noises or artifacts ('no'/'no but not so bad'/'yes');
- **Preference:** how much you like it (1–5).

To evaluate loopness, we actually repeat each sample four times in the audio recording presented to the subjects. And, since the output of the models go through the MelGAN vocoder to become waveforms, we compute the Mel spectrograms of the human-made loops and render them to audio with the same vocoder for fair comparison.

140 anonymous subjects from Taiwan participated in this test,¹⁶ with in total six unique samples by each model evaluated. Overall, the responses indicated an acceptable level of reliability (Cronbach’s $\alpha = 0.709$). We see from Figure 5 that the result of this subjective evaluation is well aligned with that of the objective evaluation, with StyleGAN2 performing the best and StyleGAN the worst, demonstrating the effectiveness of the objective metrics to some extent. Interestingly, we see no statistical difference in the ratings of the StyleGAN2 loops and the (MelGAN-vocoded) freesound loops in Drumness and Preference.

Finally, we correlate the scores of the objective metrics and subjective metrics for the 18 samples evaluated in the listening test (i.e., six samples by each GAN model). We found 0.25–0.37 correlation between IS and the four subjective metrics, and 0.01–0.16 negative correlation between FAD and the subjective metrics. The strongest correlation (0.37) is found between IS and Preference.

¹⁶ The subjects have no ideas about our models beforehand; they neither know that one of the loops they hear is human-made.

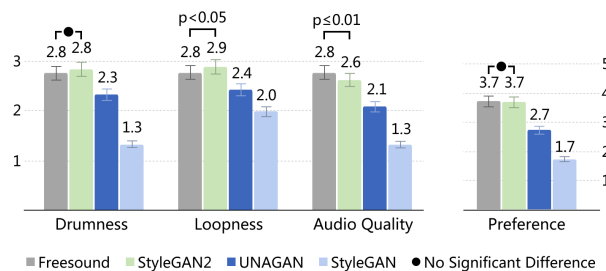


Figure 5. Subjective evaluation result for the three models trained on *freesound*. The performance difference between any pair of models in any metric is statistically significant (p -value < 0.001) under the Wilcoxon signed-rank test, except for the pairs that are explicitly highlighted.

7. CONCLUSION AND FUTURE WORK

In this paper, we have proposed using loop generation as a benchmarking task to provide a standardized evaluation of audio-domain music generation models, taking advantage of the public availability of the large collection of loops in FSLD. Moreover, we developed customized metrics to objectively evaluate the performance of such generative models for the particular case of one-bar drum loops with 120 BPM. As references, we implemented and evaluated three recent model architectures using the dataset, and discovered that StyleGAN2 works quite well. The list of models we have evaluated is short and by no means exhaustive. We wish researchers can find this benchmark useful and consider it as part of their evaluation of new models.

This work can be extended in many other directions. First and foremost, we can extend the benchmark to cover all the loops in FSLD (and looperman). The major complexity here could be the challenge to build a model that fits it all; we may need separate generative models and vocoders for different types of loops.

Second, we are certainly interested in the case of generating loops that have different tempos, rather than a fixed tempo at 120. This will require the generative models to be capable of generating variable-length output, which seems more realistic in musical audio applications.

We can also extend the benchmark to generate four-bar loops (which are not simply repeating a one-bar loop quadruple times), as there are actually a big collection of 6,656 four-bar drum loops in the looperman dataset. We do not evaluate this in this paper, as the public freesound dataset does not contain many such four-bar loops.

We also want to include more objective metrics in the future, such as using the Audio Commons Audio Extractor [34] to evaluate the “loopness” of the generated samples, or using an automatic drum transcription model [51–53] to assess the plausibility of the created percussive patterns.

Besides the benchmarking initiative, we are interested in further improving audio-domain loop generation itself and exploring new use cases, e.g., to have a conditional generation model that gives users some control (in similar veins to [3, 4, 26]), or to aim at generating novel loops by means of a creative adversarial network (CAN) [30].

8. ACKNOWLEDGEMENTS

This research work is supported by the Ministry of Science and Technology (MOST), Taiwan, under grant number 109-2628-E-001-002-MY2.

9. REFERENCES

- [1] J. Engel *et al.*, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proc. Int. Conf. Machine Learning*, 2017.
- [2] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, “GANSynth: Adversarial neural audio synthesis,” in *Proc. Int. Conf. Learning Representations*, 2019.
- [3] J. Nistal, S. Lattner, and G. Richard, “DrumGAN: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks,” in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020.
- [4] A. Ramires, P. Chandna, X. Favory, E. Gómez, and X. Serra, “Neural percussive synthesis parameterised by high-level timbral features,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2020.
- [5] A. van den Oord *et al.*, “WaveNet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [6] S. Mehri *et al.*, “SampleRNN: An unconditional end-to-end neural audio generation model,” in *Proc. Int. Conf. Learning Representations*, 2017.
- [7] C. J. Carr and Z. Zukowski, “Generating albums with SampleRNN to imitate metal, rock, and punk bands,” *arXiv preprint:1811.06633*, 2018.
- [8] S. Vasquez and M. Lewis, “MelNet: A generative model for audio in the frequency domain,” *arXiv preprint arXiv:1906.01083*, 2019.
- [9] J.-Y. Liu, Y.-H. Chen, Y.-C. Yeh, and Y.-H. Yang, “Unconditional audio generation with generative adversarial networks and cycle regularization,” in *Proc. INTERSPEECH*, 2020.
- [10] —, “Score and lyrics-free singing voice generation,” in *Proc. Int. Conf. Computational Creativity*, 2020.
- [11] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint:2005.00341*, 2020.
- [12] K. van den Broek, “MP3net: coherent, minute-long music generation from raw audio with a simple convolutional GAN,” *arXiv preprint arXiv:2101.04785*, 2021.
- [13] T. Karras *et al.*, “Analyzing and improving the image quality of StyleGAN,” in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2020.
- [14] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, “Evaluation of CNN-based automatic music tagging models,” in *Proc. Sound and Music Computing Conf.*, 2020.
- [15] A. Ramires *et al.*, “The Freesound Loop Dataset and annotation tool,” in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020.
- [16] G. Stillar, “Loops as genre resources,” *Folia Linguistica*, vol. 39, no. 1-2, pp. 197 – 212, 2005. [Online]. Available: <https://www.degruyter.com/view/journals/flin/39/1-2/article-p197.xml>
- [17] J. Ching, A. Ramires, and Y.-H. Yang, “Instrument role classification: Auto-tagging for loop based music,” in *Proc. Joint Conference on AI Music Creativity*, 2020.
- [18] I. J. Goodfellow *et al.*, “Generative adversarial nets,” in *Proc. Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [19] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2019, pp. 4396–4405.
- [20] B.-Y. Chen, J. Smith, and Y.-H. Yang, “Neural loop combiner: Neural network models for assessing the compatibility of loops,” in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020.
- [21] R. Vogl and P. Knees, “An intelligent drum machine for Electronic Dance Music production and performance,” in *Proc. Int. Conf. New Interfaces for Musical Expression*, 2017.
- [22] T. Salimans *et al.*, “Improved techniques for training GANs,” in *Proc. Conf. Neural Information Processing Systems*, 2016, pp. 2226–2234.
- [23] J. Nistal, C. Aouameur, S. Lattner, and G. Richard, “VQCPC-GAN: Variable-length adversarial audio synthesis using vector-quantized contrastive predictive coding,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2021.
- [24] C. Aouameur, P. Esling, and G. Hadjeres, “Neural drum machine: An interactive system for real-time synthesis of drum sounds,” *arXiv preprint arXiv:1907.02637*, 2019.
- [25] J. Drysdale, M. Tomczak, and J. Hockman, “Adversarial synthesis of drum sounds,” in *Proc. Int. Conf. Digital Audio Effects*, 2020.
- [26] P. Chandna, A. Ramires, X. Serra, and E. Gómez, “LoopNet: Musical loop synthesis conditioned on intuitive musical parameters,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2021.
- [27] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Baman, “Learning to groove with inverse sequence transformations,” in *Proc. Int. Conf. Machine Learning*, 2019.

- [28] V. Thio, H.-M. Liu, Y.-C. Yeh, and Y.-H. Yang, “A minimal template for interactive web-based demonstrations of musical machine learning,” in *Proc. Workshop on Intelligent Music Interfaces for Listening and Creation*, 2019.
- [29] G. Alain, M. Chevalier-Boisvert, F. Osterrath, and R. Piche-Taillefer, “DeepDrummer: Generating drum loops using deep learning and a human in the loop,” *arXiv preprint arXiv:2008.04391*, 2020.
- [30] N. Tokui, “Can GAN originate new electronic dance music genres? – Generating novel rhythm patterns using GAN with genre ambiguity loss,” *arXiv preprint: 2011.13062*, 2020.
- [31] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks,” in *Proc. AAAI Conf. Artificial Intelligence*, 2018.
- [32] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, “Learning a latent space of multitrack measures,” *arXiv preprint arXiv:1806.00195*, 2018.
- [33] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, “PopMAG: Pop music accompaniment generation,” in *Proc. ACM Multimedia Conf.*, 2020.
- [34] F. Font, T. Brookes, G. Fazekas, M. Guerber, A. La Burthe, D. Plans, M. Plumbley, W. Wang, and X. Serra, “Audio Commons: Bringing Creative Commons audio content to the creative industries,” in *Proc. AES Int. Conf. Audio for Games*, 2016.
- [35] S. Lattner and M. Grachten, “High-level control of drum track generation using learned patterns of rhythmic interaction,” in *Proc. IEEE Work. Applications of Signal Processing to Audio and Acoustics*, 2019.
- [36] S. Barratt and R. Sharma, “A note on the inception score,” in *Proc. ICML Works. Theoretical Foundations and Applications of Deep Generative Models*, 2018.
- [37] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new Python audio and music signal processing library,” in *Proc. ACM Multimedia Conf.*, 2016.
- [38] S. Böck, F. Krebs, and G. Widmer, “Joint beat and downbeat tracking with recurrent neural networks,” in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2016.
- [39] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, “An experimental comparison of audio tempo induction algorithms,” *IEEE Trans. Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.
- [40] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet Audio Distance: A metric for evaluating music enhancement algorithms,” *arXiv preprint arXiv: 1812.08466*, 2019.
- [41] E. Richardson and Y. Weiss, “On GANs and GMMs,” in *Proc. Conf. Neural Information Processing Systems*, 2018.
- [42] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of GANs for improved quality, stability, and variation,” in *Proc. Int. Conf. Learning Representations*, 2018.
- [43] K. Palkama, L. Juvela, and A. Ilin, “Conditional spoken digit generation with StyleGAN,” in *Proc. INTER-SPEECH*, 2020.
- [44] R. Prenger, R. Valle, and B. Catanzaro, “WaveGlow: A flow-based generative network for speech synthesis,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2019.
- [45] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, “DiffWave: A versatile diffusion model for audio synthesis,” in *Proc. Int. Conf. Learning Representations*, 2021.
- [46] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, “MelGAN: Generative adversarial networks for conditional waveform synthesis,” *arXiv preprint arXiv:1910.06711*, 2019.
- [47] L. Mescheder, A. Geiger, and S. Nowozin, “Which training methods for gans do actually converge?” *arXiv preprint arXiv:1801.04406*, 2018.
- [48] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” in *Proc. Int. Conf. Learning Representations*, 2019.
- [49] D. Berthelot, T. Schumm, and L. Metz, “BEGAN: Boundary equilibrium generative adversarial networks,” *arXiv preprint:1703.10717*, 2017.
- [50] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, 2017.
- [51] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch, “A review of automatic drum transcription,” *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, 2018.
- [52] K. Choi and K. Cho, “Deep unsupervised drum transcription,” in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020.
- [53] L. Callender, C. Hawthorne, and J. H. Engel, “Improving perceptual quality of drum transcription with the expanded groove MIDI dataset,” *arXiv preprint:2004.00188*, 2020.

EMOPIA: A MULTI-MODAL POP PIANO DATASET FOR EMOTION RECOGNITION AND EMOTION-BASED MUSIC GENERATION

Hsiao-Tzu Hung^{1,2} Joann Ching¹ Seungheon Doh³
Nabin Kim⁴ Juhan Nam³ Yi-Hsuan Yang¹

¹ Academia Sinica, Taiwan

² Department of CSIE, National Taiwan University

³ Graduate School of Culture Technology, KAIST, South Korea

⁴ Georgia Institute of Technology, United States

r08922a20@csie.ntu.edu.tw, joann8512@citi.sinica.edu, seungheondoh@kaist.ac.kr

ABSTRACT

While there are many music datasets with emotion labels in the literature, they cannot be used for research on symbolic-domain music analysis or generation, as there are usually audio files only. In this paper, we present the EMOPIA (pronounced ‘yee-mò-pi-uh’) dataset, a shared multi-modal (audio and MIDI) database focusing on perceived emotion in pop piano music, to facilitate research on various tasks related to music emotion. The dataset contains 1,087 music clips from 387 songs and clip-level emotion labels annotated by four dedicated annotators. Since the clips are not restricted to one clip per song, they can also be used for song-level analysis. We present the procedure for building the dataset, covering the song list curation, clip selection, and emotion annotation processes. Moreover, we prototype use cases on clip-level music emotion classification and emotion-based symbolic music generation by training and evaluating corresponding models using the dataset. The result demonstrates the potential of EMOPIA for being used in future exploration on piano emotion-related MIR tasks.

1. INTRODUCTION

The affective aspect of music has been a major subject of research in the field of music information retrieval (MIR), not only for music analysis and labeling [1–9], but also for music generation or editing [10–14]. Accordingly, there have been quite a few public music datasets with emotion, as listed in Table 1. These datasets are different in many ways, including the musical genres considered, data modality and data size, and the way emotion is described.

With the growing interest in symbolic-domain music analysis and generation in recent years of ISMIR [15–19], it is desirable to have an emotion-labeled symbolic music

dataset to add emotion-related elements to such research. However, among the datasets listed in Table 1, only two provide MIDI data, and they are both small in size. Moreover, the majority of the audio-only datasets contain songs of multiple genres, making it hard to apply automatic music transcription algorithms, which currently work better for piano-only music [20–23], to get MIDI-like data from the audio recordings.

To address this need, we propose a new emotion-labeled dataset comprising of three main nice properties:

- **Single-instrument.** We collect audio recordings of piano covers and creations from YouTube, with fair to high audio and musical quality, and a diverse set of playing styles and perceived emotions. Focusing on only piano music allows for the use of piano transcription algorithms [20, 21], and facilitates disentanglement of musical composition from variations in timbre, arrangement, and other confounds seen in multi-instrument music.
- **Multi-modal.** Both the audio and MIDI versions of the music pieces can be found from the Internet (see Section 3.5 for details). The MIDI files are automatically transcribed from the audio by a state-of-the-art model [21].
- **Clip-level annotation.** The audio files downloaded from YouTube are full songs. As different parts of a song may convey different emotions, the first four authors of the paper manually and carefully pick emotion-consistent short clips from each song and label the emotion of these clips using a four-class taxonomy derived from the Russell’s valence-arousal model [32]. This leads to clip-level emotion annotations for in total 1,087 clips from 387 songs (i.e., 2.78 clips per song on average), with the number of clips per emotion class fairly balanced.

Given these properties, EMOPIA has versatile use cases in MIR research. For music labeling, EMOPIA can be used for clip-level music emotion recognition or music emotion variation detection [2], in both the audio and symbolic domains. For music generation, EMOPIA can be used for emotion-conditioned piano music generation or style transfer, to create emotion-controllable new compositions, or variations of existing pieces, again in both domains.

We present details of the dataset and the way we com-



© H.T.Hung and J. Ching, and S.H Doh. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: H.T.Hung and J. Ching, and S.H Doh, “EMOPIA: A Multi-modal Pop Piano Dataset for Emotion Recognition and Emotion-based Music Generation”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

Name	Label type	Genre (or data source)	Size	Modality
Jamendo Moods [24]	adjectives	multiple genres	18,486	Audio
DEAM [25]	VA values	(from FMA [26], Jamendo, MedleyDB [27])	1,802	Audio
EMO-Soundscapes [28]	VA values	(from FMA)	1,213	Audio
CCMED-WCMED [29]	VA values	classical (both Western & Chinese)	800	Audio
emoMusic [30]	VA values	pop, rock, classical, electronic	744	Audio
EMusic [31]	VA values	experimental, 8 others	140	Audio
MOODetector [6]	adjectives	multiple genres (AllMusic)	193	Audio+MIDI
VGMIDI [10]	valence	video game	95	MIDI
EMOPIA (ours)	Russell’s 4Q	pop (piano covers)	1,078	Audio+MIDI

Table 1. Comparison of some existing public emotion-labeled music datasets and the proposed EMOPIA dataset.

pile it in Section 3, and report a computational analysis of the dataset in Section 4. Moreover, to demonstrate the potential of the new dataset, we use it to train and evaluate a few clip-level emotion classification models using both audio and MIDI data in Section 5, and emotion-conditioned symbolic music generation models in Section 6. The latter involves the use of a recurrent neural network (RNN) model proposed by Ferreira *et al.* [10], and our own modification of a Transformer-based model [33–35] that takes emotion as a conditioning signal for generation.

We release the EMOPIA dataset at a Zenodo repo.¹ Source code implementing the generation and classification models can be found on GitHub.^{2 3} Examples of generated pieces can be found at a demo webpage.⁴

2. RELATED WORK

Emotion Recognition in Symbolic Music. Symbolic music representations describe music with the note, key, tempo, structure, chords, instruments. To understand the relationship between music and emotion, various researchers have investigated machine learning approaches with handcrafted features. Grekow *et al.* [36] extract in total 63 harmony, rhythmic, dynamic features from 83 classic music MIDI files. Lin *et al.* [37] compare audio, lyric, and MIDI features for music emotion recognition, finding that MIDI features lead to higher performance in valence dimension. Panda *et al.* [6] also proposed multi-model approaches, combining audio and MIDI features for emotion recognition using a small dataset of 193 songs.

Emotion-conditioned Symbolic Music Generation. Only few work has started to address this task recently. Ferreira *et al.* [10] compile a small dataset of video game MIDI tracks with manual annotations of valence values, named VGMIDI (cf. Table 1), and use it to train a long short term memory network (LSTM) in tandem with a genetic algorithm (GA) based elite selection mechanism to generate positive or negative music. Makris *et al.* [12] approach the same task by using designated chord progression sequence in a sequence-to-sequence architecture

trained with the VGMIDI dataset. Zhao *et al.* [38] use LSTM to generate music with four different emotions. Madhok *et al.* [13] use human facial expressions as the condition to generate music. More recently, Tan & Herremans demonstrate that their FaderNets [16] can achieve arousal-conditioned symbolic music style transfer with a semi-supervised clustering method that learns the relation between high-level features and low-level representation. Their model modifies the emotion (specifically, only the arousal) of a music piece, instead of generating new pieces from scratch.

3. THE EMOPIA DATASET

3.1 Song Selection and Segmentation

EMOPIA is a collection of 387 piano solo performances of popular music segmented manually into 1,087 clips for emotion annotation. Two authors of the paper curated the song list of the piano performances by scanning through playlists on Spotify for its consistently high quality, then downloading the recordings from YouTube. A song is included when it is played by a professional conveying a clear emotion, and the recording has not been heavily engineered during post-production. The genres of songs include Japanese anime, Korean and Western pop song covers, movie soundtracks, and personal compositions.

In an effort to extend the usefulness of the dataset for future research, at the best of our ability, the songs are intentionally segmented (with the help of the Sonic Visualizer [40]) only at cadential arrivals to make it an *emotionally-consistent clip* and a valid *musical phrase* at the same time. Accordingly, EMOPIA contains information for full songs, extracted phrases, and emotion labels.

3.2 Emotion Annotation

Different emotion taxonomies have been adopted in the literature for emotion annotation, with no standard so far [2]. For EMOPIA, we consider a simple four-class taxonomy corresponding to the four quadrants of the Russell’s famous Circumplex model of affect [32], which conceptualizes emotions in a two-dimensional space defined by valence and arousal. The four classes are: HVHA (high valence high arousal); HVLA (high valence low arousal);

¹ <https://zenodo.org/record/5090631>

² <https://github.com/annahung31/EMOPIA>

³ https://github.com/SeungHeonDoh/EMOPIA_cls

⁴ <https://annahung31.github.io/EMOPIA/>

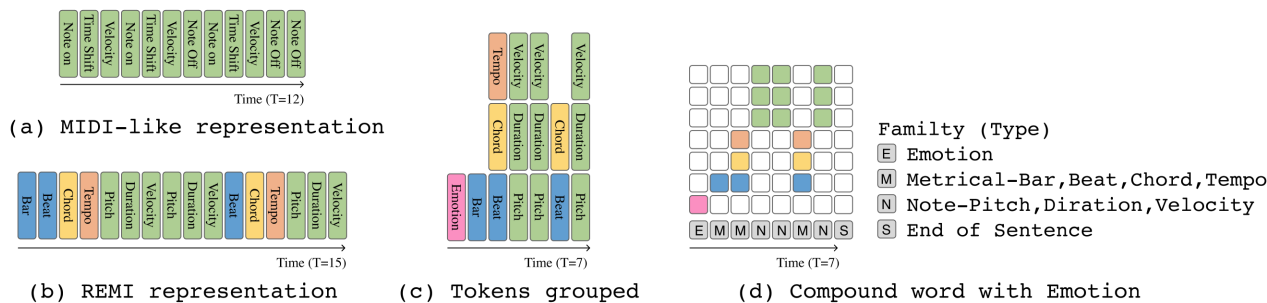


Figure 1. Illustration of different token-based representation for symbolic music: (a) MIDI-like [39], (b) REMI [34], and (d) CP [35] plus emotion token. Sub-figure (c) is an intermediate representation of the CP one.

Quadrant	# clips	Avg. length (in sec / #tokens)
Q1	250	31.9 / 1,065
Q2	265	35.6 / 1,368
Q3	253	40.6 / 771
Q4	310	38.2 / 729

Table 2. The number of clips and their average length in seconds, or in the number of REMI tokens, for each quadrant of the Russell’s model in the EMOPIA dataset.

LVHA (low valence high arousal); and LVLA (low valence low arousal). We refer to this taxonomy as Russell’s 4Q.

As pointed out in the literature [2, 3, 41], various factors affect the perceived emotion of music, including cultural background, musical training, gender, etc. Consensus on the perception of emotion is challenging accordingly. Therefore, the annotations were made only among the first four authors, all coming from similar cultural backgrounds and collaborating closely during the annotation campaign, to ensure mutual standards for high or low valence/arousal. As it is time-consuming and laborious to choose clips from a song and label the emotion, each song was taken care of by only one annotator. Yet, cross validation of the annotations among the annotators was made several times during the annotation campaign (which spans 2.5 months), to ensure all annotators work on the same standard.

Table 2 shows the number of clips and the average length (in seconds) for each class. The clips amount to approximately 11 hours’ worth of data.

3.3 Transcription

We transcribe the selected clips automatically with the help of the high-resolution piano transcription model proposed by Kong *et al.* [21], which is open source and represents the state-of-the-art for this task. We have manually checked the transcription result for a random set of clips and find the accuracy in note pitch, velocity, and duration satisfactory. The transcription might be fragmented and undesirable for cases such as when the audio recording is engineered to have unnatural ambient effects; we drop such songs from our collection. The model also transcribes pedal information, which we include to EMOPIA but do not use in our experiments.

3.4 Pre-processing and Encoding

For building machine learning models that deal with symbolic data, we need a data representation that can be used as input to the models. For example, MusicVAE [42] adopts the *event-based* representation that encodes a symbolic music piece as a sequence of “event tokens” such as note-on and note-off, while MuseGAN [43] employs a timestep-based, *piano roll*-like representation. Since there is no standard on the symbolic representation thus far, we adopt the following event-based ones in our experiments. Specifically, we use MIDI-like and REMI in Section 5 and CP in Section 6. See Figure 1 for illustrations.

- The **MIDI-like** representation [39] encodes information regarding a MIDI note with a “note-on” token, a “note-off”, and a “velocity” token. Moreover, the “time shift” token is used to indicate the relative time gap (in ms) between two tokens.
- **REMI** [34] considers a beat-based representation that instead uses “bar” and “subbeat” tokens to represent the time information. A “bar” token signifies the beginning of a new bar, and “subbeat” points to one of the constant number of subbeat divisions in a bar. Additionally, REMI uses “tempo” tokens to control the pace of the music, and replaces “note-off” with “note-duration”. Table 2 also shows the average number of REMI tokens for clips in each emotion class.
- **CP** [35]. Both MIDI-like and REMI view events as individual tokens. In CP, tokens belonging to the same *family* are grouped into a *super token* and placed on the same timestep (see Figure 1(c)). CP considers by default three families: *metrical*, *note*, and *end-of-sequence*. We additionally consider the “emotion” tokens and make it a new family, as depicted in Figure 1(d). The *prepending* approach is motivated by CTRL [44], a state-of-the-art controllable text generation model in natural language processing (NLP) that uses global tokens to affect some overall properties of a sequence.

3.5 Dataset Availability

In EMOPIA, each sample is accompanied with its corresponding metadata, segmentation annotations, emotion annotation, and transcribed MIDI, which are all available

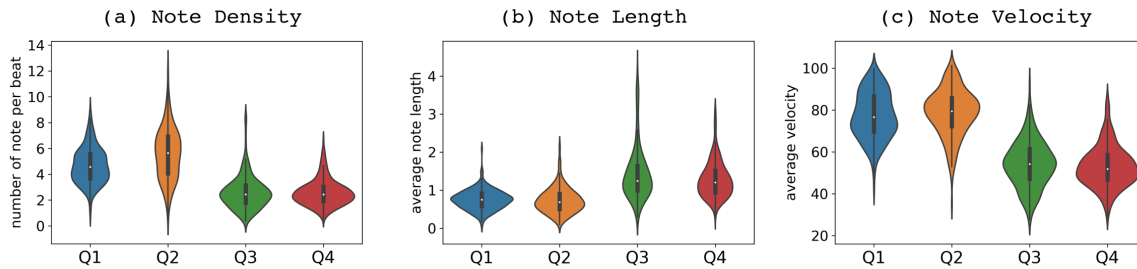


Figure 2. Violin plots of the distribution in (a) note density, (b) length, and (c) velocity for clips from different classes.

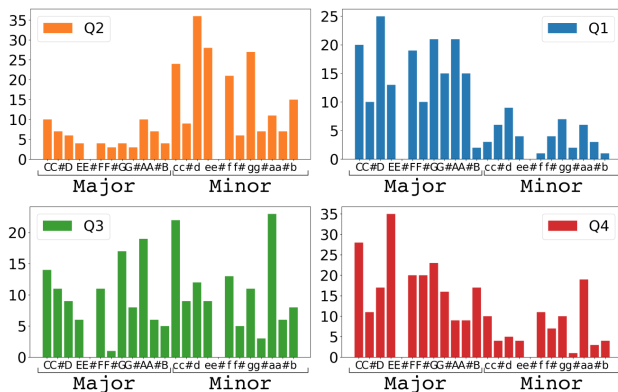


Figure 3. Histogram of the keys (left / right: major / minor keys) for clips from different emotion classes.

Label pair	Different arousal		Different valence	
	Q1 vs. Q4	Q2 vs. Q3	Q1 vs. Q2	Q3 vs. Q4
JS Div.	0.236	0.250	0.434	0.280

Table 3. Jensen-Shannon divergence between the key histograms of a few emotion quadrant pairs.

in the Zenodo repository. Moreover, we have added the MIDI data to the MusPy library [18] to facilitate its usage. Due to copyright issues, however, we can only share audio through YouTube links instead of sharing the audio files directly; the availability of the songs are subject to the copyright licenses in different countries and whether the owners will remove them.

4. DATASET ANALYSIS

The emotion that listeners perceive is determined by a wide array of composed and performed features of music [45]. To observe the emotional correlate of the musical attributes in EMOPIA, we extract various MIDI-based features and examine the distributions over the four quadrants of emotion. We present here the most discriminative features among many choices we examined in our analysis.

Note Density, Length, and Velocity. The arousal of music can be easily observed based on the frequency of note occurrences and their strength [46]. We measure them by note density, length and velocity. The note density is defined as the number of notes per beat, and the note

length is defined as the average note length in beat unit. The note velocity is obtained directly from MIDI. Figure 2 shows three violin plots of the three features. The general trend shows that the high-arousal group (Q1, Q2) and low-arousal group (Q3, Q4) are distinguished well in all three plots. The results of note density and velocity are expected considering the nature of arousal. However, it is quite interesting that the note lengths are generally longer in the low-arousal group (Q3, Q4). Between the same-arousal quadrants, the differences are subtle. In note density, Q2 has more dynamics than Q1, whereas Q3 is not distinguishable from Q4. In note length, Q1 has slightly longer notes than those of Q2, whereas Q3 is again not distinguishable from Q4. In velocity, Q2 have louder notes than those of Q1, and Q3 has slightly louder notes than Q4.

Key Distribution. The valence of music is often found to be related to the major-minor tonality [7]. For simplicity, we measure the tonality from the musical key. We extract the key information using the Krumhansl-Kessler algorithm [47] in the MIDI ToolBox [48]. Figure 3 shows the key distributions on 12 major-minor pitch classes for the four emotion quadrants. They assure that the major-minor tonality is an important clue in distinguishing valence. In the high valence group (Q1, Q4), the distribution is skewed to the left (major), while, in the low valence group (Q2, Q3), the trend is the opposite. We also measured the distance between the key distributions using the Jensen-Shannon divergence, which has the property of symmetry. Table 3 summarizes the pair-wise distances, indicating that the valence difference group has a larger difference than the arousal difference group.

5. MUSIC EMOTION RECOGNITION

We report our baseline research on both symbolic- and audio-domain emotion recognition using EMOPIA, which defines the task as classifying a song into four categories. The clips in EMOPIA are divided into train-validation-test splits with the ratio of 7:2:1 in a stratified manner.

Symbolic-domain Classification. We evaluate two methods: one is based on hand-crafted features with a simple classifier, and the other is on a deep neural network model. For the former, we use the analysis features in the previous section and a logistic regression classifier as a baseline. Specifically, we use average values of note density, note

Model	4Q	A	V
Logistic regression	.581	.849	.651
LSTM-Attn [49]+MIDI-like [39]	.684	.882	.833
LSTM-Attn [49]+REMI [34]	.615	.890	.746

Table 4. Symbolic-domain classification performance.

Model	4Q	A	V
Logistic regression	.523	.919	.558
Short-chunk ResNet [50]	.677	.887	.704

Table 5. Audio-domain classification performance.

length, velocity, and represent the key as a one-hot vector. For the latter, we use two different symbolic note representation methods introduced in Section 3.4, MIDI-like [39] and REMI [34]. For the learning model, we use the combination of bidirectional LSTM and a self-attention module, or *LSTM-Attn* for short, proposed originally for sentiment classification in NLP [49]. The LSTM extracts temporal information from the MIDI note events, while the self-attention module calculates different weight vectors over the LSTM hidden states with multi-head attentions. The weighted hidden states are finally used for classification.

Audio-domain Classification. We evaluate two audio-domain classification methods in a similar manner to the symbolic-domain ones. In the first method, we use an average of 20 dimensions of mel-frequency cepstral coefficient (MFCC) vectors and a logistic regression classifier. In the second method, we use the short-chunk ResNet following [50], which is composed of 7-layer CNNs with residual connections. The output is summarized as a fixed 128-dimensional vector through max pooling, which is followed by two fully connected layers with the ReLU activation for classification. The input audio to the short-chunk ResNet is 3-second excerpts represented as a log-scaled mel-spectrogram with 128 bins with 1024-size FFT (Hanning window), and 512 size hop at 22,050 Hz sampling rate. We randomly sample three seconds of the audio chunk as an input size to the classifier.

Evaluation. We calculate 4-way classification accuracy over the four different emotion quadrants and 2-way classification accuracy over either arousal and valence. Tables 4 and 5 show the results in the symbolic and audio domains, respectively. Except for arousal, we can see that the deep learning approaches generally outperform the logistic regression classifiers using hand-crafted features. In audio domain arousal classification, MFCC vectors averaging the entire song sequence showed better performance than the deep learning approach with 3-second input. It seems that wider input sequence has the strength in emotion recognition. In both domains, valence classification is a more difficult task compared to arousal classification. For valence classification, MIDI-domain classifiers yield better result

than audio-domain classifiers (0.883 vs. 0.704). Among the two token representations, MIDI-like seems to outperform REMI for valence classification.

6. EMOTION-CONDITIONED GENERATION

We build the Transformer and LSTM models for emotion-conditioned symbolic music generation using EMPOIA. For the former, we adopt the Compound Word Transformer [35], the state-of-the-art in unconditioned symbolic music generation. We employ the CP+emotion representation presented in Section 3.4 as the data representation.

For the LSTM model, we consider the approach proposed by Ferreira *et al.* [10], which represents the state-of-the-art in emotion-conditioned music generation. Our implementation follows that described in [10], with the following differences: 1) train on EMOPIA rather than VG-MIDI; 2) use 512 neurons instead of 4,096 due to our limited computational resource; 3) use the same linear logistic regression layer for classification but we classify four classes instead of two.

As the size of EMOPIA might not be big enough, we use additionally the *AILabs1k7* dataset compiled by Hsiao *et al.* [35] to pre-train the Transformer. *AILabs1k7* contains 1,748 samples and is also pop piano music, but it does not contain emotion labels. Most of the clips in *AILabs1k7* are longer than EMOPIA, so to keep the consistency of the input sequence length, the length of the token sequence is set to be 1,024. We pre-train the Transformer with $1e-4$ learning rate on *AILabs1k7*, take the checkpoint with negative log-likelihood loss 0.30, and then fine-tune it on EMOPIA with $1e-5$ learning rate. During pre-training, the emotion token is always set to be “ignore,” while in fine-tuning it is set to the emotion of that sample.

Evaluation. We use the following three sets of metrics.

- **Surface-level objective metrics.** We use the following three metrics proposed by Dong *et al.* [43] to evaluate whether the generated samples fit the training data: pitch range (PR), number of unique pitch classes used (NPC), and number of notes being played concurrently (POLY). We use MusPy [18] to compute these metrics.
- **Emotion-related objective metrics.** Since both REMI and CP adopt a beat-based representation of music, we employ the LSTM-Attn+REMI emotion classifier (cf. Section 5) here to quantify how well the generation result is influenced by the emotion condition. We first use the generative model to generate 100 samples per class, and use the assigned label as the target class of the sample. The trained classifier is then used to make prediction on the generated samples. Similar to the classification task, apart from 4Q classification, we also conduct 2-way classification of both Arousal and Valence aspect.
- **Subjective metrics.** As the classifiers are not 100% accurate, we also resort to a user survey to evaluate the emotion-controllability of the models. Specifically, we deploy an online survey to collect responses to the music generated by different models. A subject has to lis-

Model	Objective metrics						Subjective metrics		
	PR	NPC	POLY	4Q	A	V	Humanness	Richness	Overall
EMOPIA (i.e., real data)	51.0	8.48	5.90	—	—	—	—	—	—
LSTM+GA [10]	59.1	9.27	3.39	.238	.500	.498	2.59±1.16	2.74±1.12	2.60±1.07
CP Transformer [35]	53.4	9.20	3.48	.418	.690	.583	2.61±1.03	2.81±1.03	2.78±1.03
CP Transformer w/ pre-training	49.6	8.54	4.40	.403	.643	.590	3.31±1.18	3.22±1.23	3.26±1.15

Table 6. Performance comparison of the evaluated models for emotion-conditioned symbolic music generation in *surface-level objective metrics* (Pitch Range, Number of Pitch Classes used, and POLYphony; the closer to that of the real data the better), *emotion-related objective metrics* (4Q classification, Arousal classification, Valence classification; the higher the better), and *subjective metrics* (all in 1–5; the higher the better); bold font highlights the best result per metric.

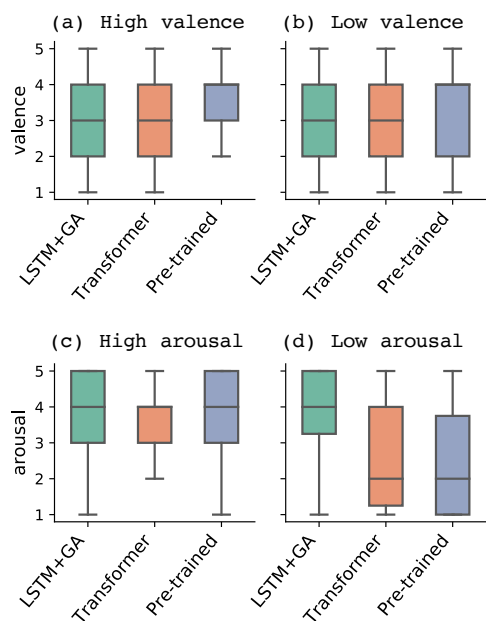


Figure 4. The subjective emotional scores (in 1–5) for the generative results when the target emotion is (a) high valence, (b) low valence, (c) high arousal, (d) low arousal.

ten to 12 random-generated samples, one for each of the three models and each of the four emotion classes, and rate them on a five-point Likert scale with respect to 1) Valence: is the audio negative or positive; 2) Arousal: is low or high in arousal; 3) Humanness: how well it sounds like a piece played by human; 4) Richness: is the content interesting; and, 5) Overall musical quality. In total 25 subjects participated in the survey.

Table 6 tabulates some of the results. We see that the CP Transformer with (‘w/’) pre-training performs the best in most of the objective metrics and the three subjective metrics listed here. Nevertheless, the scores of the CP Transformer with pre-training in the three emotion-related objective metrics are much lower than that reported in Table 4, suggesting that either the generated pieces are not emotion-laden, or the generated pieces are too dissimilar to the real pieces to the classifier.

Figure 4 shows the human assessment of the emotion-controllability of the models. To our surprise, while the CP

Transformer with pre-training does not score high in the emotion related objective metrics, the subjective test shows that it can actually control the emotion of the generated pieces to a certain extent, better than the two competing models. In particular, the valence of the samples generated by the Transformer with pre-training has a median rating of 4 when the goal is to generate positive-valence music (i.e., Figure 4(a)), while the scores of the other two models are around 3. Moreover, the arousal of the samples generated by the Transformer with pre-training has a median rating of 4 when the goal is to generate high-arousal music (Figure 4(c)), which is higher than that of the non pre-trained Transformer. This suggests that the LSTM-Attn classifier employed for computing the emotion-related objective metrics may not be reliable enough to predict the emotion perceived by human, and that the Transformer with pre-training is actually effective in controlling the emotion of the music it generates to certain extent. But, the model seems not good enough for the cases of generating low-valence (i.e., negative) music, as shown in Figure 4(b).

7. CONCLUSION

In this paper, we have proposed a new public dataset EMOPIA, a medium-scale emotion-labeled pop piano dataset. It is a multi-modal dataset that contains both the audio files and MIDI transcriptions of piano-only music, along with clip-level emotion annotations in four classes. We have also presented prototypes of models for clip-level music emotion classification and emotion-based symbolic music generation trained on this dataset, using a number of state-of-the-art models in respective tasks. The result shows that we are able to achieve high accuracy in both four-quadrant and valence-wise emotion classification, and that our Transformer-based model is capable of generating music with a given target emotion to a certain degree.

In the future, in-depth importance analysis can be conducted to figure out features that are important for emotion classification, and to seek ways to incorporate those features to the generation model. Many ideas can also be tried to further improve the performance of emotion conditioning, e.g., the Transformer-GAN approach [51].

We share not only the dataset itself but the code covering all our implemented models in a GitHub repo. We hope that researchers will find this contribution useful in future emotion-related MIR tasks.

8. ACKNOWLEDGEMENT

This research is supported by Year 2021 Culture Technology R&D Program by Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (Project Name: Research Talent Training Program for Emerging Technologies in Games, Project Number: R2020040211, Contribution Rate: 50%)

9. REFERENCES

- [1] Y. Kim *et al.*, “Music emotion recognition: A state of the art review,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2010.
- [2] Y.-H. Yang and H. H. Chen, *Music Emotion Recognition*. CRC Press, 2011.
- [3] J. S. Gómez-Cañón, E. Cano, P. Herrera, and E. Gómez, “Joyful for you and tender for us: the influence of individual characteristics and language on emotion labeling and classification,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [4] J. S. Gómez-Cañón, E. Cano, Y.-H. Yang, P. Herrera, and E. Gómez, “Let’s agree to disagree: Consensus entropy active learning for personalized music emotion recognition,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2021.
- [5] J. de Berardinis, A. Cangelosi, and E. Coutinho, “The multiple voices of musical emotions: Source separation for improving music emotion recognition models and their interpretability,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [6] R. Panda, R. Malheiro, B. Rocha, A. Oliveira, and R. P. Paiva, “Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis,” in *Proc. Int. Symposium on Computer Music Multidisciplinary Research*, 2013.
- [7] R. Panda, R. M. Malheiro, and R. P. Paiva, “Audio features for music emotion recognition: A survey,” *IEEE Trans. Affective Computing*, pp. 1–20, 2020.
- [8] S. Chowdhury, A. Vall, V. Haunschmid, and G. Widmer, “Towards explainable music emotion recognition: The route via mid-level features,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2019.
- [9] C. Cancino-Chacón, S. Peter, S. Chowdhury, A. Aljanaki, and G. Widmer, “On the characterization of expressive performance in classical music: First results of the con espresione game,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [10] L. N. Ferreira and J. Whitehead, “Learning to generate music with sentiment,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2019.
- [11] K. Chen, C.-I. Wang, T. Berg-Kirkpatrick, and S. Dubnov, “Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [12] D. Makris, K. R. Agres, and D. Herremans, “Generating lead sheets with affect: A novel conditional seq2seq framework,” in *Proc. Int. Joint Conf. Neural Networks*, 2021.
- [13] R. Madhok, S. Goel, and S. Garg, “SentiMozart: Music generation based on emotions,” in *Proc. Int. Conf. Agents and Artificial Intelligence*, 2018.
- [14] A. M. Grimaud and T. Eerola, “EmoteControl: an interactive system for real-time control of emotional expression in music,” *Personal and Ubiquitous Computing*, 2020.
- [15] T.-P. Chen, S. Fukayama, M. Goto, and L. Su, “Chord Jazzification: Learning Jazz interpretations of chord symbols,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [16] H. H. TAN and D. Herremans, “Music Fadernets: Controllable music generation based on high-level features via low-level feature modelling,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [17] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, “POP909: A pop-song dataset for music arrangement generation,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [18] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, “MusPy: A toolkit for symbolic music generation,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [19] A. McLeod, J. Owers, and K. Yoshii, “The MIDI Degradation Toolkit: Symbolic music augmentation and correction,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2020.
- [20] C. Hawthorne *et al.*, “Onsets and Frames: Dual-objective piano transcription,” *arXiv preprint arXiv:1710.11153*, 2018.
- [21] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, “High-resolution piano transcription with pedals by regressing onsets and offsets times,” *arXiv preprint arXiv:2010.01815*, 2020.
- [22] C. Hawthorne *et al.*, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in *Proc. Int. Conf. Learning Representations*, 2019.
- [23] K. W. Cheuk, D. Herremans, and L. Su, “ReconVAT: A semi-supervised automatic music transcription framework for low-resource real-world data,” in *Proc. ACM Multimedia*, 2021.

- [24] D. Bognadov, A. Porter, P. Tovstogan, and M. Won, “Mediaeval 2019: Emotion and theme recognition in music using Jamendo,” in *MediaEval 2019 Workshop*.
- [25] A. Alajanki, Y.-H. Yang, and M. Soleymani, “Benchmarking music emotion recognition systems,” *PLOS ONE*, 2016.
- [26] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “FMA: A dataset for music analysis,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2017.
- [27] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, “MedleyDB: A multitrack dataset for annotation-intensive mir research,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2014.
- [28] J. Fan, M. Thorogood, and P. Pasquier, “Emo-Soundscapes: A dataset for soundscape emotion recognition,” in *Proc. Int. Conf. Affective Computing and Intelligent Interaction*, 2017.
- [29] J. Fan, Y.-H. Yang, K. Dong, and P. Pasquier, “A comparative study of Western and Chinese classical music based on soundscape models,” in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing*, 2020.
- [30] M. Soleymani, M. Caro, E. Schmidt, C.-Y. Sha, and Y.-H. Yang, “1000 songs for emotional analysis of music,” in *Proc. ACM Int. Workshop on Crowdsourcing for Multimedia*, 2013.
- [31] J. Fan, K. Tatar, M. Thorogood, and P. Pasquier, “Ranking-based emotion recognition for experimental music,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2017.
- [32] J. Russell, “A circumplex model of affect,” *Journal of Personality and Social Psychology*, vol. 39, pp. 1161–1178, December 1980.
- [33] A. Vaswani *et al.*, “Attention is all you need,” in *Proc. Advances in Neural Information Processing Systems*, 2017.
- [34] Y.-S. Huang and Y.-H. Yang, “Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions,” in *Proc. ACM Multimedia*, 2020.
- [35] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, “Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs,” in *Proc. AAAI*, 2021.
- [36] J. Grekow and Z. W. Raś, “Detecting emotions in classical music from MIDI files,” in *Proc. Int. Symposium on Methodologies for Intelligent Systems*, 2009.
- [37] Y. Lin, X. Chen, and D. Yang, “Exploration of music emotion recognition based on MIDI,” in *Proc. Int. Society for Music Information Retrieval Conf.*, 2013.
- [38] K. Zhao, S. Li, J. Cai, H. Wang, and J. Wang, “An emotional symbolic music generation system based on LSTM networks,” in *Proc. IEEE Information Technology, Networking, Electronic and Automation Control Conf.*, 2019.
- [39] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, “This time with feeling: Learning expressive musical performance,” *Neural Computing and Applications*, 2018.
- [40] C. Cannam, C. Landone, and M. Sandler, “Sonic Visualiser: An open source application for viewing, analysing, and annotating music audio files,” in *Proc. ACM Multimedia*, 2010, pp. 1467–1468, <https://www.sonicvisualiser.org/>.
- [41] L.-L. Balkwill and W. Thompson, “A cross-cultural investigation of the perception of emotion in music: Psychophysical and cultural cues,” *Music Perception*, vol. 17, pp. 43–64, October 1999.
- [42] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, “A hierarchical latent vector model for learning long-term structure in music,” in *Proc. Int. Conf. Machine Learning*, 2018.
- [43] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Proc. AAAI*, 2018.
- [44] N. S. Keskar *et al.*, “CTRL - a conditional Transformer language model for controllable generation,” *arXiv preprint arXiv:1909.05858*, 2019.
- [45] P. Juslin and E. Lindstorm, “Musical expression of emotions: Modelling listeners’ judgements of composed and performed features,” *Music Analysis*, vol. 29, pp. 334 – 364, 2011.
- [46] S. R. Livingstone, R. Muhlberger, A. R. Brown, and W. F. Thompson, “Changing musical emotion: A computational rule system for modifying score and performance,” *Computer Music Journal*, vol. 34, no. 1, pp. 41–64, 2010.
- [47] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 2001.
- [48] T. Eerola and P. Toiviainen, *MIDI Toolbox: MATLAB Tools for Music Research*. Jyväskylä, Finland: University of Jyväskylä, 2004.
- [49] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, “A structured self-attentive sentence embedding,” in *Proc. Int. Conf. Learning Representations*, 2017.
- [50] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, “Evaluation of CNN-based automatic music tagging models,” in *Proc. Sound and Music Conf.*, 2020.
- [51] A. Muhamed *et al.*, “Symbolic music generation with Transformer-GANs,” in *Proc. AAAI*, 2021.

PIANO SHEET MUSIC IDENTIFICATION USING MARKETPLACE FINGERPRINTING

Kevin Ji Daniel Yang TJ Tsai

Harvey Mudd College

kji, dhyang, ttsai@hmc.edu

ABSTRACT

This paper studies the problem of identifying piano sheet music based on a cell phone image of all or part of a physical page. We re-examine current best practices for large-scale sheet music retrieval through an economics perspective. In our analogy, the runtime search is like a consumer shopping in a store. The items on the shelves correspond to fingerprints, and purchasing an item corresponds to doing a fingerprint lookup in the database. From this perspective, we show that previous approaches are extremely inefficient marketplaces in which the consumer has very few choices and adopts an irrational buying strategy. The main contribution of this work is to propose a novel fingerprinting scheme called marketplace fingerprinting. This approach redesigns the system to be an efficient marketplace in which the consumer has many options and adopts a rational buying strategy that explicitly considers the cost and expected utility of each item. We also show that deciding which fingerprints to include in the database poses a type of minimax problem in which the store and the consumer have competing interests. On experiments using all solo piano sheet music images in IMSLP as a searchable database, we show that marketplace fingerprinting substantially outperforms previous approaches and achieves a mean reciprocal rank of 0.905 with sub-second average runtime.

1. INTRODUCTION

This paper tackles the problem of identifying piano sheet music based on a cell phone picture of all or part of a physical page. This is the camera-based sheet music identification task. Such a system could be used to conveniently retrieve Youtube videos of relevant performances, compare different scores for a particular passage of music, or – more generally – explore representations of sheet music that are useful for alignment and retrieval.

Previous work on retrieval tasks involving sheet music fall into three groups. The first group of related works study audio–sheet alignment and retrieval. The earliest works used Optical Music Recognition (OMR) systems to

convert sheet music to a symbolic format like MIDI, extracted chroma features from both MIDI and audio, and then performed alignment or retrieval using dynamic time warping (DTW). This approach has been used to synchronize audio and sheet music [1–4] and perform audio–sheet retrieval [5, 6]. More recent works have explored the use of convolutional neural networks (CNNs) to project both sheet music and audio into an embedding space where similarity can be computed directly. This approach has been applied to various forms of audio–sheet music alignment [7–11] and audio–sheet retrieval [7, 12, 13]. The second group of related works study symbolic–sheet retrieval. Several recent works studying MIDI–sheet retrieval have used the bootleg score feature representation [14], which encodes the positions of noteheads relative to staff lines. This representation has been used for MIDI–sheet passage retrieval [14, 15] and to find matches between the Lakh MIDI dataset and IMSLP sheet music [16, 17]. Other approaches find matches with symbolic queries by performing OMR or object recognition on the sheet music, and then doing an n-gram lookup [18, 19], string matching [20], or keyword spotting [21]. The third group of related works — and the works that are most directly relevant to our present study — explore sheet–sheet retrieval. Hajic et al. [22] use OMR to convert sheet music to MIDI and then use DTW on the pitch sequences. Waloschek et al. [23] use a CNN to project entire measures into an embedding space to align different sheet music editions of the same piece. Yang and Tsai [24] propose a dynamic n-gram fingerprint derived from bootleg score features to identify sheet music based on cell phone images of physical pages.

This paper re-examines current practices for large-scale sheet–sheet retrieval from an economics perspective. This perspective makes clear what the weaknesses of current approaches are and suggests ways to improve them. We will focus our analysis on the dynamic n-gram approach [24], since it is the largest-scale study (using all solo piano sheet music images in IMSLP) and achieves robust sub-second retrieval (0.85 MRR). The dynamic n-gram approach has three steps: (1) it extracts a sequence of bootleg score features x_1, x_2, \dots, x_L from the query image, (2) it constructs either a 1-gram (x_i), 2-gram (x_i, x_{i+1}), 3-gram (x_i, x_{i+1}, x_{i+2}), or 4-gram ($x_i, x_{i+1}, x_{i+2}, x_{i+3}$) fingerprint at each offset $i = 1, 2, \dots, L$, where the size of the n-gram at offset i is selected at runtime to ensure that the number of fingerprint matches in the database is below a certain threshold, and (3) the n-gram fingerprints are used



© K. Ji, D. Yang, and T. Tsai. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Ji, D. Yang, and T. Tsai, “Piano Sheet Music Identification Using Marketplace Fingerprinting”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

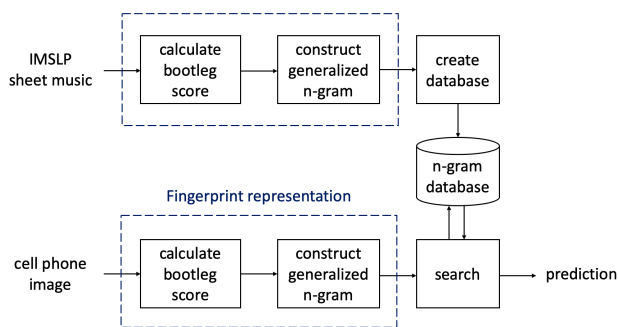


Figure 1. Overview of the marketplace fingerprinting approach. The upper half describes the offline process of constructing the database, and the lower half describes the online process of performing a real-time search.

with an inverted file index to identify the database item containing the most fingerprint matches.

Consider the following analogy. Imagine that the run-time search is like a consumer shopping in a store. Every offset $i = 1, 2, \dots, L$ in the bootleg score sequence is like an aisle in the store. The 1-gram, 2-gram, 3-gram, and 4-gram fingerprints are like items on the shelves. In the dynamic n-gram approach, the store has exactly four items on every aisle, and the consumer always purchases the most expensive item in each aisle that is below a maximum acceptable price. From an economics perspective, this setup is horrible for the consumer, and the consumer’s purchasing strategy is irrational.

The main contribution of this paper is to propose a novel fingerprinting scheme called marketplace fingerprinting. Marketplace fingerprinting is the result of redesigning the system above using principles of economics to produce a more efficient marketplace. One of the key principles in this approach is that more options and choices are good for the consumer. We generalize the notion of an n-gram in order to produce a much larger set of n-gram types, which corresponds to offering many more items in each aisle. Furthermore, we adopt a much more rational purchasing strategy in which the consumer explicitly considers the cost and the expected utility of each item, purchases the items with the highest utility-to-cost ratio, and is allowed to purchase multiple items in each aisle as long as they stay under budget. We also show that the database design problem (i.e. which n-gram fingerprints to include in the database) presents a type of minimax problem in which the store and the consumer have competing interests. On experiments involving all solo piano sheet music images in IMSLP, we show that the marketplace fingerprinting method substantially improves retrieval accuracy compared to the dynamic n-gram method.¹

2. SYSTEM DESCRIPTION

Figure 1 shows an overview of our proposed approach. We will describe the system in three parts: computing the fin-

¹Code can be found at <https://github.com/HMC-MIR/ImprovedSheetID>.

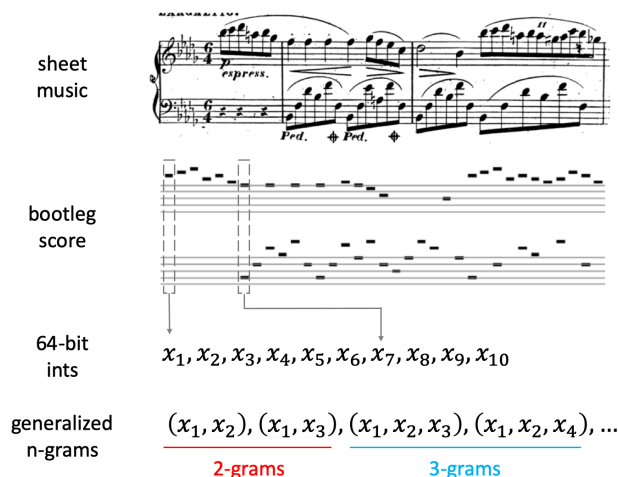


Figure 2. Description of the generalized n-gram fingerprint representation. The sheet music is first converted to a bootleg score, each column of the bootleg score is represented as a 64-bit integer, and n-grams are constructed from various groupings of integers.

gerprint representation (Section 2.1), creating the database (Section 2.2), and searching the database (Section 2.3). The bootleg score representation is adopted from previous work, but the generalized n-gram, database construction, and search mechanism are novel contributions.

2.1 Fingerprint Representation

Figure 2 shows how n-gram fingerprints are computed. This process is described in the next two paragraphs.

The first step is to extract bootleg score features. The bootleg score is a mid-level feature representation that encodes the positions of filled noteheads relative to staff lines in piano sheet music [14]. The bootleg score itself is a $62 \times N$ binary matrix, where 62 indicates the total number of distinct staff line positions in both the left and right hand staves and N indicates the number of grouped note events in the sheet music (e.g. a chord containing four notes played simultaneously would constitute a single grouped note event). Note that this representation throws away a significant amount of information such as key signature, time signature, accidentals, note duration, octave markings, clef changes, and non-filled noteheads. Nonetheless, it has been successfully used in several applications involving sheet music, including sheet music identification [16, 24], sheet-MIDI retrieval [14, 15, 17], and sheet-audio alignment [25, 26]. We represent each column (containing 62 bits) as a 64-bit integer, so that the bootleg score is encoded as a sequence of integers x_1, x_2, \dots, x_N .

The second step is to construct generalized n-grams. The concept of n-grams comes from linguistics [27], where the frequency of word sequences in a large corpus was historically used for language modeling. Many previous works have likewise used n-grams for language modeling with music data (e.g. [28, 29]). Here, we use a generalization of n-grams that is specifically useful for indexing and retrieval. When constructing generalized n-grams at

offset i for a bootleg score sequence x_1, x_2, \dots, x_N , we consider any combination of n elements that satisfies two conditions: (1) the leftmost element must be x_i , and (2) the elements must be selected from a fixed context window of length C . For example, when $C = 4$ and we consider up to 3-grams, there is one 1-gram $\{(x_i)\}$, three 2-grams $\{(x_i, x_{i+1}), (x_i, x_{i+2}), (x_i, x_{i+3})\}$, and three 3-grams $\{(x_i, x_{i+1}, x_{i+2}), (x_i, x_{i+1}, x_{i+3}), (x_i, x_{i+2}, x_{i+3})\}$, resulting in a total of 7 generalized n-grams. In our experiments, we consider up to 3-grams with $C = 6$, which results in a total of $T = 16$ different generalized n-gram types at each offset i . We will denote the generalized n-grams at offset i as $y_{i1}, y_{i2}, \dots, y_{iT}$.

2.2 Database Construction

If we had an infinite amount of RAM available, the database construction problem would be trivial: we would simply include all of the generalized n-grams in the database. However, because the IMSLP dataset is large and we have many different types of n-grams, the total amount of memory required to store everything in the database quickly becomes exorbitant. This forces us to be strategic in choosing which n-grams to include in the database and which to exclude.

The database is simply a reverse index in which the key is the n-gram fingerprint and the value is a list of all instances in the IMSLP dataset where the n-gram occurs. This list consists of (PDF, offset) tuples that specify the IMSLP PDF and bootleg score offset where the fingerprint occurs. Note that the n-gram *type* and n-gram *value* must both match in order to be included in the reverse index (i.e. a (x_i, x_{i+1}) 2-gram and (x_i, x_{i+2}) 2-gram will never collide).

We approach this problem from the perspective of a store manager who has a limited amount of shelf space and wants to fill the shelves with products that maximize some utility function. There are two different types of resources in this scenario. The first resource is shelf space, which in our case corresponds to the amount of RAM available. In our experiments, we work with a machine that has 128 GB of RAM. The second resource of interest is the utility function. Our utility function is the expected number of match points that will be added to the true matching PDF at runtime. Consider a single n-gram fingerprint y_{ij} at offset i in a query. If y_{ij} is not in the database or if the bootleg score representation has errors (e.g. it fails to detect a notehead in the sheet music), the true matching item in the database will accumulate 0 match points. On the other hand, if y_{ij} is included in the database and the bootleg score computation is correct, the true matching item in the database will accumulate 1 match point. Therefore, the expected (added) utility for including y_{ij} in the database is the probability that its constituent bootleg score representation is correct. We estimated this probability of correctness as a fixed constant for each of the 16 generalized n-gram types based on the training data, as shown in Figure 3. Unsurprisingly, the 1-gram has the highest probability of correctness, and the 3-grams had the lowest (4-grams were even lower). Using

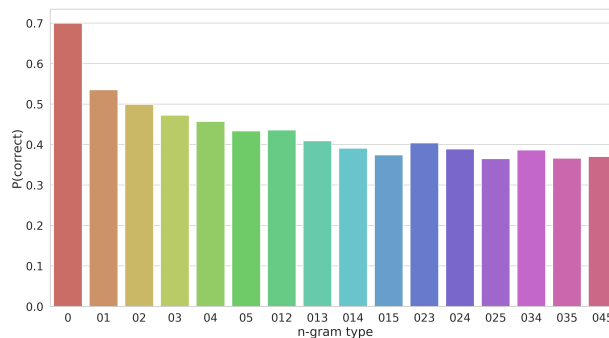


Figure 3. Probability of correctness for various generalized n-gram types, as estimated on training data. Each bar indicates the fraction of n-grams of that type in the training queries whose underlying bootleg score representation matched the database.

this approach, the total utility of an n-gram fingerprint y_{ij} is $U(y_{ij}) = N(y_{ij})P_{correct}(j)$, where $N(y_{ij})$ indicates the total number of times the fingerprint occurs in the IMSLP data and where $P_{correct}(j)$ indicates the probability that the underlying bootleg score representation is correct (as shown in Figure 3).

At this point, we *could* simply sort all of the unique n-gram fingerprints by their utility (largest to smallest), and then add them to the database in order until the memory is used up. However, this ignores the fact that the store manager (i.e. the database construction problem) and the consumer (i.e. the runtime search problem) have competing interests. Note that the utility function rewards fingerprints that occur very frequently.² For example, the n-gram with the highest utility is a 1-gram (i.e. single bootleg score column) with a single notehead present, which occurs more than a million times in the database. This fingerprint offers the store manager the highest return, but is an awful proposition for the consumer. From the search perspective, the primary constraining resource is runtime, not memory. Given two different n-grams to choose from — one that occurs 1 million times in the database and one that occurs once in the database — the latter is far more desirable at runtime: both offer the possibility of adding 1 match point to the true matching item in the database, but one requires processing 1 million matches and the other only requires processing 1 match. Therefore, the database construction problem is a type of minimax problem, in which the store manager wants to offer the most frequently occurring fingerprints but the consumer wants to purchase the least frequently occurring fingerprints.

Based on these considerations, the database is constructed in the following way. All of the unique n-gram fingerprints in IMSLP are sorted by their utility value $U(y_{ij})$ from largest to smallest. We discard any fingerprints that occur more than $\gamma = 10,000$ times in the database. This value of γ is set conservatively to ensure that all n-gram

² Because there is a memory overhead for adding a new entry to the reverse index, it is possible to fit more matches in memory by adding frequently occurring fingerprints than by adding lots of rarely occurring fingerprints.

fingerprints that are actually selected at runtime on the training data are included in the database. We then add the remaining fingerprints to the database in order of utility value until the memory has been used up.

At the end of the database construction step, we have a reverse index that contains a subset of the generalized n-grams found in the IMSLP data. This subset excludes extremely common n-grams (that occur $> \gamma$ times) as well as extremely rare n-grams (due to memory limits).

2.3 Search

In our analogy, the runtime search is akin to a consumer shopping in the store. We first process the cell phone image query to compute a sequence of bootleg score features $\tilde{x}_1, \dots, \tilde{x}_L$, and then construct a set of $T = 16$ generalized n-grams y_{i1}, \dots, y_{iT} at each offset $i = 1, \dots, L$. This set of (approximately) TL candidate n-grams are the items on the store shelves that the consumer can purchase. Each offset i corresponds to an aisle in the store, and each aisle contains T items.³ The consumer uses a strategy (described in the next few paragraphs) to purchase selected candidate n-grams from each aisle of the store. These selected n-grams are then used to search the database. The consumer purchasing strategy and the search mechanism are described in the next three paragraphs.

The consumer adopts a disciplined greedy purchasing strategy. They first decide on a budget for the whole store (B_{tot}), divide it by the number of aisles to determine a budget per aisle ($B_{aisle} = B_{tot}/L$), and then purchase as many items in each aisle as they can in order to maximize utility while staying under their budget for the aisle. Any unused funds from an aisle carry over to the next aisle. During a search, the primary constrained resource is runtime – we do not want a query to take too long to process. The runtime is directly correlated with the number of fingerprint matches in the database that are processed. The total budget B_{tot} is therefore specified in terms of the maximum total number of matches we are willing to process for each query. B_{tot} is a hyperparameter that can be selected to achieve a desired runtime. In our experiments, we set $B_{tot} = 65000$ in order to achieve ≈ 1 second average runtime per query on the training set.

How should the consumer decide which items (i.e. which n-grams) to purchase in each aisle? We assume that the consumer is rational and wants to maximize their own utility. We define the utility as the expected number of match points added to the true matching item in the database, which (as explained in Section 2.2 paragraph 3) is the probability that the underlying bootleg score representation is correct. We again approximate this probability based on the n-gram type and the statistics on the training data (as shown in Figure 3). The runtime cost for each item is proportional to the number of fingerprint matches in the database, since processing many fingerprint matches will require more runtime. Putting this all together, the consumer tries to maximize utility

by adopting the following strategy: they purchase items in each aisle in decreasing order of their utility-to-cost ratio $R(y_{ij}) = P_{correct}(j)/N(y_{ij})$ until the budget for the aisle (B_{tot}/L plus any carryover from the previous aisle) has been spent. Note that $P_{correct}(j)$ can be determined based only on the n-gram type and $N(y_{ij})$ can be determined quickly without needing to actually process the fingerprint matches. This information allows the system to dynamically adjust which n-grams to select in an informed and rational manner.

The search is performed using the histogram of offsets method [30]. This method provides an efficient way to search a large database in order to find a sequence of matching fingerprints aligned in time. For each fingerprint y_{ij} (i.e. the n-gram of type j at offset i in the query) that is purchased by the consumer, the system processes the list of fingerprint matches in the database from the reverse index. Each fingerprint match is specified by two pieces of information: the PDF and the offset k in the bootleg score where the fingerprint occurs. Processing a fingerprint match (i.e. a (PDF, k) tuple) means adding the relative offset $k - i$ to the PDF's histogram. Note that a sequence of matching fingerprints aligned in time will result in a histogram with a large spike at the true relative offset (i.e. where the query occurs in the PDF). Therefore, we can use the maximum bin count in each histogram as a match score for the PDF. Because IMSLP often contains multiple PDFs for a single piece, we calculate the *piece* match score as the maximum score among its constituent PDFs. Finally, we sort all pieces in the database by their piece score. The resulting ranked list is the final output of the system.

2.4 Comparison to Dynamic N-gram

It is instructive to compare the proposed marketplace fingerprinting approach to the dynamic n-gram approach. We will compare these two along the three axes described above: the fingerprint representation, the database construction, and the search mechanism.

Fingerprint representation. The dynamic n-gram approach considers four types of standard n-grams at each offset i : (x_i) , (x_i, x_{i+1}) , (x_i, x_{i+1}, x_{i+2}) , and $(x_i, x_{i+1}, x_{i+2}, x_{i+3})$. The marketplace fingerprinting approach generalizes the notion of an n-gram and offers a much wider selection of n-gram types to the consumer.

Database construction. The dynamic n-gram approach is to simply add all 1-grams to the database, then all 2-grams, then all 3-grams, etc. until memory runs out. For 128 GB of RAM, this approach maxes out at 4-grams. This results in a database that has complete representation of four different types of n-grams. The marketplace fingerprinting approach, on the other hand, considers 16 different types of n-grams and selects a subset of the most useful fingerprints from each n-gram type in a principled way.

Search. In our analogy, the dynamic n-gram approach corresponds to a store in which every aisle has four items, and the consumer purchases exactly one item per aisle. The consumer purchasing strategy is to set a fixed budget for

³ The last few aisles may contain less than T items due to a lack of context.

each aisle and to purchase the most expensive item in each aisle that is under the budget. The marketplace fingerprinting approach corresponds to a store in which every aisle has 16 (or more) items, and the consumer can purchase multiple items per aisle. The consumer purchasing strategy is to set a fixed total budget for the whole store, determine a budget per aisle to control their total spending, and purchase the items in each aisle with highest utility-to-cost ratio until the aisle budget has been spent. The key idea behind the marketplace fingerprinting approach is that options and choices are good for the consumer. Once the consumer has purchased a set of fingerprints, both approaches use the histogram of offsets method to search the database.

3. EXPERIMENTAL SETUP

The experimental setup is identical to [24] for fair comparison with the dynamic n-gram approach. We describe the data and evaluation metrics below for completeness.

The queries in our system are cell phone images taken of physical pages of piano sheet music. There are 10 cell phone images of 200 different piano pieces across 25 different composers, resulting in a total of 2000 queries. The pictures were taken with four different cell phone models. The cell phone pictures are spread across the length of the piece and are taken in a variety of different physical locations, lighting conditions (including both flash and no flash), and levels of zoom (between 1 and 5 lines of sheet music). The proposed marketplace fingerprinting system and the baseline systems do not have any trainable parameters (only hyperparameters), so we use only 40 pieces for training (400 queries) and the remaining 160 pieces for testing (1600 queries).

The database consists of all solo piano sheet music in IMSLP. In total, there are 31,384 PDFs, 29,310 pieces, and 374,758 pages of sheet music. [24] provides a pre-computed dataset of bootleg score features on all solo piano sheet music in IMSLP, and we use this dataset without modification.

We evaluate system performance along two axes: retrieval accuracy and runtime. Because each query matches exactly one unique *piece* in IMSLP (with multiple different PDF versions), we use mean reciprocal rank (MRR) as a measure of retrieval accuracy. MRR is computed as

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{R_i} \tag{1}$$

where N indicates the number of queries and R_i indicates the rank of the true matching piece. For the IMSLP piano dataset, R_i ranges between 1 and 29,310. Note that MRR ranges between 0 and 1, where 1 corresponds to perfect performance. We also measure the runtime required to process each query and report the average and standard deviation of the runtimes. All experiments were performed on a 2.1 GHz Intel Xeon processor with 128 GB RAM.

System	MRR	Runtime	
		avg	std
1-gram	.709	21.5s	12.5s
2-gram	.845	2.76s	1.11s
3-gram	.808	1.99s	.36s
4-gram	.755	1.12s	.25s
5-gram	.688	1.07s	.13s
dynamic n-gram	.853	.98s	.12s
marketplace	.905	.95s	.14s

Table 1. Comparison of system performance on the camera-based piano sheet music identification task. The middle column indicates the retrieval accuracy in terms of mean reciprocal rank (MRR), and the rightmost column indicates the average and standard deviation of runtimes. The bottom row shows the performance of the proposed marketplace fingerprinting system.

4. RESULTS

We compare the marketplace fingerprinting approach to six other baseline systems. The first five baselines are fixed n-grams with $n = 1, 2, 3, 4, 5$. These approaches use a single fixed-size n-gram as the fingerprint representation. For example, given a sequence of bootleg score features x_1, \dots, x_L , the fixed 2-gram baseline would construct fingerprints of the form (x_i, x_{i+1}) . The fixed n-gram approach for large-scale retrieval was explored in [17]. The sixth baseline is the dynamic n-gram method [24], which represents the state-of-the-art in sheet music identification. For a given sequence of query bootleg score features x_1, \dots, x_L , the dynamic n-gram constructs one fingerprint at each offset i of the form $(x_i, x_{i+1}, \dots, x_{i+L_i})$ where $0 \leq L_i \leq 3$. The size of the n-gram is selected dynamically for each offset i in order to ensure that the number of fingerprint matches in the database is below a specified threshold.

Table 1 shows the retrieval accuracy and runtime for all 7 systems. We can see that the marketplace fingerprinting system has the highest retrieval accuracy by a large margin (.905 vs .853). We can roughly interpret this gap as a reduction in “errors” by about $\frac{1}{3}$ compared to the dynamic n-gram approach. Furthermore, this improvement in retrieval accuracy does not come at the expense of runtime: the runtime budget (B_{tot}) for the marketplace system was selected to achieve < 1 second average runtime per query on the training set. The improvement in retrieval accuracy instead comes from utilizing the runtime more *efficiently* – processing many more fingerprints with a higher utility-to-cost ratio compared to the dynamic n-gram approach.

5. ANALYSIS

In this section we explore two questions of interest to gain deeper intuition into the performance of the marketplace fingerprinting system.

The first question of interest is, “What is the effect of the runtime budget (B_{tot})?” This is a hyperparameter specify-

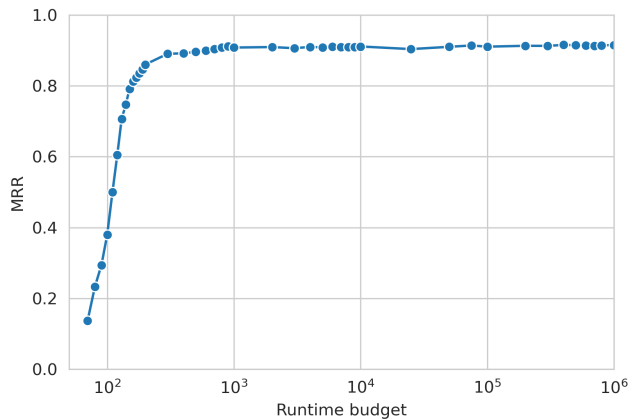


Figure 4. Relationship between runtime budget (B_{tot}) and retrieval accuracy. The runtime budget is a hyperparameter that specifies the maximum number of fingerprint matches in the database that can be processed for a single query.

ing the total number of fingerprint matches in the database that we are willing to process for each query. The total runtime budget determines the budget for each aisle, which in turn determines how many n-gram fingerprints will be processed. By setting B_{tot} appropriately, we can thus trade off runtime for retrieval accuracy: if we’re willing to wait longer to process a query, we can get higher quality results.

Figure 4 shows the retrieval accuracy of the marketplace fingerprinting system across a range of runtime budget values. The retrieval accuracy improves dramatically as the runtime budget increases from 1 to 500, but then reaches a plateau and remains approximately constant for runtime budget values greater than 1000. This plateau strongly suggests that we have reached an upper bound on performance through the use of redundancy in the fingerprint lookups. Thus, we would not expect that adding more n-gram types would improve results further. Any significant additional improvements to retrieval accuracy would likely need to come from a more accurate bootleg score (or alternative) representation. Perhaps the most surprising finding in Figure 4 is how *low* the runtime budget is when it reaches the plateau. With only a budget of 1000 matches – chosen strategically, of course – it is possible to already achieve a MRR of .908 with an average runtime of 0.71 seconds. This is quite remarkable considering that the dynamic n-gram model performs lookups on fingerprints with up to 10,000 matches in the database, so that a single lookup would likely use up the entire runtime budget.

The second questions of interest is, “What is the distribution of fingerprints in the database?” Figure 5 shows the distribution of fingerprints in the databases for all seven systems.⁴ The fingerprints are sorted from most frequent (left) to least frequent (right), and their frequency of occurrence in the IMSLP dataset is shown on the y-axis. Note that both axes are shown on a log scale. The ideal distribution for optimal hashing performance is a uniform (flat)

⁴ The curves for the fixed n-gram systems match those of Figure 3 in [24]. However, we have confirmed that the curve for the dynamic n-gram system in Figure 3 of [24] is incorrect. The brown curve in Figure 5 above shows the corrected distribution.

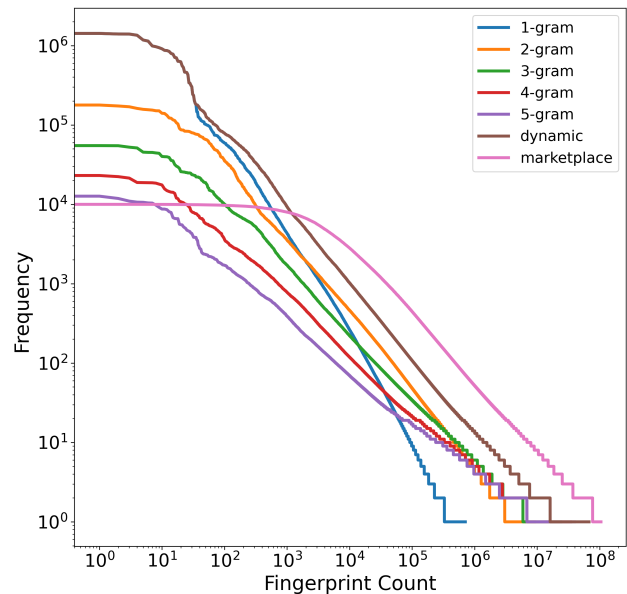


Figure 5. Fingerprint distribution for different systems. For each system, the set of unique fingerprints in the database are sorted from most frequent (left) to least frequent (right), and the y-axis indicates the frequency of occurrence. Note that both axes are on a log scale.

distribution, in which all fingerprints occur the same number of times. We can see that the marketplace fingerprinting system has the flattest distribution, avoiding extremely common fingerprints and minimizing extremely rare fingerprints. By considering many more types of n-grams, it is able to achieve a flatter distribution that is closer to the ideal uniform distribution.

6. CONCLUSION

This paper proposes a way to identify sheet music using a novel fingerprinting scheme called marketplace fingerprinting. Our approach considers the retrieval problem through the lens of an economic marketplace in which a consumer (the search) with a finite budget (runtime) purchases items (fingerprints) in a store (the database). Building off of previous work that uses n-grams of bootleg score features as fingerprints, we generalize the notion of n-grams to greatly expand the number of different types of fingerprints in order to give the consumer more options to choose from. We show that choosing which fingerprints to include in the database presents a type of minimax problem in which the consumer (the runtime search problem) and the store (the database design problem) have competing interests. At runtime, the consumer (the search) is presented with many different options (fingerprint types) to choose from and tries to maximize utility by purchasing the items (i.e. doing database lookups on fingerprints) that have maximum utility-to-cost ratio while staying under a fixed budget (runtime). With experiments using all solo piano sheet music images in IMSLP as a searchable database, we show that the marketplace fingerprinting approach substantially outperforms previous approaches.

7. REFERENCES

- [1] V. Thomas, C. Fremerey, M. Müller, and M. Clausen, "Linking sheet music and audio – challenges and new approaches," in *Dagstuhl Follow-Ups*, M. Müller, M. Goto, and M. Schedl, Eds. Schloss Dagstuhl–Leibniz- Zentrum für Informatik, Dagstuhl, Germany, 2016, vol. 3, pp. 1–22.
- [2] C. Fremerey, M. Müller, , and M. Clausen, "Handling repeats and jumps in score-performance synchronization," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2010, pp. 243–248.
- [3] D. Damm, C. Fremerey, F. Kurth, M. Müller, and M. Clausen, "Multimodal presentation and browsing of music," in *Proceedings of the International Conference on Multimodal Interfaces*, 2008, pp. 205–208.
- [4] F. Kurth, M. Müller, C. Fremerey, Y.-H. Chang, and M. Clausen, "Automated synchronization of scanned sheet music with audio recordings," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2007, pp. 261–266.
- [5] C. Fremerey, M. Müller, F. Kurth, and M. Clausen, "Automatic mapping of scanned sheet music to audio recordings," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2008, pp. 413–418.
- [6] C. Fremerey, M. Clausen, S. Ewert, and M. Müller, "Sheet music-audio identification," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2009, pp. 645–650.
- [7] M. Dorfer, A. Arzt, and G. Widmer, "Learning audio-sheet music correspondences for score identification and offline alignment," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2017, pp. 115–122.
- [8] —, "Towards score following in sheet music images," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2018, pp. 784–791.
- [9] M. Dorfer, F. Henkel, and G. Widmer, "Learning to listen, read, and follow: Score following as a reinforcement learning game," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2018, pp. 784–791.
- [10] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, "Score following as a multi-modal reinforcement learning problem," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 67–81, 2019.
- [11] F. Henkel, R. Kelz, and G. Widmer, "Learning to read and follow music in complete score sheet images," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2020, pp. 780–787.
- [12] M. Dorfer, J. Hajič, A. Arzt, H. Frostel, and G. Widmer, "Learning audio-sheet music correspondences for cross-modal retrieval and piece identification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 22–23, 2018.
- [13] M. Dorfer, J. Schlüter, A. Vall, F. Korzeniowski, and G. Widmer, "End-to-end cross-modality retrieval with cca projections and pairwise ranking loss," *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 117–128, 2018.
- [14] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, "Midi passage retrieval using cell phone pictures of sheet music," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 916–923.
- [15] T. Tsai, D. Yang, M. Shan, T. Tanprasert, and T. Jenrungrot, "Using cell phone pictures of sheet music to retrieve midi passages," *IEEE Transactions on Multimedia*, vol. 22, no. 12, pp. 3115–3127, 2020.
- [16] D. Yang and T. Tsai, "Piano sheet music identification using dynamic n-gram fingerprinting," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 42–51, 2021.
- [17] T. Tsai, "Towards linking the lakh and imslp datasets," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 546–550.
- [18] J. Thompson, A. Hankinson, and I. Fujinaga, "Searching the liber usualis: Using CouchDB and ElasticSearch to query graphical music documents," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2011.
- [19] S. P. Achankunju, "Music search engine from noisy omr data," in *International Workshop on Reading Music Systems*, 2018, pp. 23–24.
- [20] R. Malik, P. P. Roy, U. Pal, and F. Kimura, "Handwritten musical document retrieval using music-score spotting," in *IEEE International Conference on Document Analysis and Recognition*, 2013, pp. 832–836.
- [21] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Probabilistic music-symbol spotting in handwritten scores," in *IEEE International Conference on Frontiers in Handwriting Recognition*, 2018, pp. 558–563.
- [22] J. Hajič, M. Kolárová, A. Pacha, and J. Calvo-Zaragoza, "How current optical music recognition systems are becoming useful for digital libraries," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, 2018, pp. 57–61.
- [23] S. Waloschek, A. Hadjakos, and A. Pacha, "Identification and cross-document alignment of measures in music score images," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 137–146.

- [24] D. Yang and T. Tsai, “Camera-based piano sheet music identification,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2020, pp. 481–488.
- [25] M. Shan and T. Tsai, “Improved handling of repeats and jumps in audio-sheet image synchronization,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2020, pp. 62–69.
- [26] —, “Automatic generation of piano score following videos,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 29–41, 2021.
- [27] P. F. Brown, V. J. Della Pietra, P. V. Desouza, J. C. Lai, and R. L. Mercer, “Class-based n-gram models of natural language,” *Computational Linguistics*, vol. 18, no. 4, pp. 467–480, 1992.
- [28] M. Hontanilla, C. Pérez-Sancho, and J. M. Inesta, “Modeling musical style with language models for composer recognition,” in *Iberian Conference on Pattern Recognition and Image Analysis*, 2013, pp. 740–748.
- [29] J. Wołkowicz and V. Kešelj, “Evaluation of n-gram-based classification approaches on classical music corpora,” in *International Conference on Mathematics and Computation in Music*, 2013, pp. 213–225.
- [30] A. Wang, “An industrial strength audio search algorithm,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2003, pp. 7–13.

LEARNING A CROSS-DOMAIN EMBEDDING SPACE OF VOCAL AND MIXED AUDIO WITH A STRUCTURE-PRESERVING TRIPLET LOSS

Keunhyoung Luke Kim¹ Jongpil Lee¹ Sangeun Kum¹ Juhan Nam²

¹ Neutune Research, Seoul, South Korea

² Graduate School of Culture Technology, KAIST, South Korea

{khlukekim, jongpillee, keums}@neutune.com, juhan.nam@kaist.ac.kr

ABSTRACT

Recent advances of music source separation have achieved high quality of vocal isolation from mix audio. This has paved the way for various applications in the area of music informational retrieval (MIR). In this paper, we propose a method to learn a cross-domain embedding space between isolated vocal and mixed audio for vocal-centric MIR tasks, leveraging a pre-trained music source separation model. Learning the cross-domain embedding was previously attempted with a triplet-based similarity model where vocal and mixed audio are encoded by two different convolutional neural networks. We improve the approach with a structure-preserving triplet loss that exploits not only cross-domain similarity between vocal and mixed audio but also intra-domain similarity within vocal tracks or mix tracks. We learn vocal embedding using a large-scaled dataset and evaluate it in singer identification and query-by-singer tasks. In addition, we use the vocal embedding for vocal-based music tagging and artist classification in transfer learning settings. We show that the proposed model significantly improves the previous cross-domain embedding model, particularly when the two embedding spaces from isolated vocals and mixed audio are concatenated.

1. INTRODUCTION

Vocal is the key component in popular music, as it is usually tied to the artist and melody of a song. Research reports that vocal is the most salient part in music listening experience of streaming service [1] and the most effective factor in hit song prediction [2]. A number of MIR tasks are also focused on the singing voice, for example, singer identification [3], melody extraction [4], singing transcription [5], and query-by-humming [6]. However, vocal sound sources are usually available in mixed form with instrumental sounds in popular music and isolated vocal tracks are scarcely available. This has been a barrier in singing voice research.

Recent advances of music source separation achieved a significant level of competency [7]. Pre-trained source separation models are freely available for practical applications [8]. This has opened up a great potential for various down-streaming tasks. Among others, vocal source separation found immediate uses in many relevant tasks including vocal melody extraction [9, 10], vocal tagging [11], singer identification [12, 13], and automatic drum mixing [14].

In this paper, we apply the vocal source separation to learn a cross-domain embedding space between isolated vocal and mix audio. The goal of this research is learning a discriminative vocal embedding space agnostic to mixing with instrumental sounds. This idea of cross-domain embedding space was first proposed in [15]. However, the previous work secured the isolated vocal and mix audio by a simple music mash-up, an artificial mix between two heterogeneous datasets by musical matching (i.e., tempo, beat, and key). This is not a realistic setting for obtaining a practical vocal embedding space. Furthermore, they considered only the correspondence between monophonic vocal and mixed audio in training the model using metric learning, missing vocal similarity within the same domain.

We improve the cross-domain embedding space in two ways. First, we employ a structure-preserving triplet loss that exploits not only cross-domain similarity between vocal and mixed audio but also intra-domain similarity within vocal tracks or mix tracks. Second, we conduct a large-scale cross-domain vocal embedding learning by leveraging a pre-trained vocal separation model to extract isolated vocals with high-quality. We show that the proposed method achieves significant improvements in singing identification, compared to the previous work. In addition, we evaluate the cross-domain vocal embedding for vocal tagging and artist classification in transfer learning settings and show the generalization capability.

2. RELATED WORKS

2.1 Singer Identification in Polyphonic Music

The main challenge of singer identification in polyphonic music is to extract voice features from music signals mixed with instrumental sounds. The most straightforward way to handle this issue is to separate the vocal sound sources from mixed audio as a pre-processing step. Early approaches relied on vocal melody extraction (or predominant pitch estimation) with a combination of voice re-synthesis and voice detection algorithms to obtain en-



hanced vocal sources [16–18]. They then extracted various hand-engineered features such as mel-frequency cepstral coefficient (MFCC), linear prediction mel-frequency cepstral coefficient (LPMCC) as input features for a classifier.

Recently, music source separation algorithms based on deep learning have significantly advanced [7, 19] and some of them focused more on vocal source separation [20]. This has provided a great chance to improve singer identification by allowing to use high-quality isolated vocals [13] or augmenting training data by remixing of vocals and accompaniment [12]. However, the scope of research was limited to achieving high accuracy on a small size dataset without scaling up to general vocal audio embedding.

2.2 Representation Learning

Representation learning allows unorganized input data to be mapped into an structured space [21]. Previously, this was done by a shallow network with hand-crafted features [22], and more recently deep representation learning where the deep neural networks directly learn the mapping from the raw input data became a dominant methodology. Once the representation space is structured, we can utilize it to solve the related domain problems (or downstream tasks), known as *transfer learning* [23].

Among many techniques for deep representation learning, metric learning with a triplet loss became popular because of its flexibility in training the model with less strict distance metrics [24]. This triplet loss based network is trained with three sampled examples such that the two examples are defined as similar, but not the rest one. Then, the embedding network is trained to locate the two similar examples to be closer than the dissimilar example in the representation space. This similarity-based learning can be easily extended to multi-modal data (e.g. image-to-text [25–28], video-to-text [29], face-to-voice [30], video-to-audio [31, 32]). In this case, two different embedding networks are trained for each modality. For example, if the case is to learn a cross-modal embedding of image and audio [32], an image and an audio from the same category are sampled, and an audio from the different category are sampled. Then, the embedding feature of the image and the embedding features of the audio are compared. By optimizing the embedding features of the image and audio from the same category to be placed closer than the different one, we can build cross-modal embedding space. This representation learning paradigm is close to our study. However, cross-domain embedding learning is different from the cross-modal embedding learning in that the two embedding networks are from the same modality (e.g., sketch-to-photo [33], sketch-to-3Dshape [34], street-view-to-satellite-view [35], vocal-to-mixed [15]). Because the inherent characteristics in each domain may largely differ, it is required to have separate networks for each domain.

2.3 Representation Learning in MIR

Representation learning in MIR has mainly been explored in semantic level [36–38]. Diverse similarity supervisions

have been employed to train the triplet networks. Tag labels are one representative similarity supervision by regarding the two examples similar if they belongs to the same tag [39, 40]. User’s preference data (similarity judgement [36] or listening history [41]) is another similarity supervision. Artist information has also been explored [37, 42]. In this case, the two examples are treated as similar if they are released from the same artist. The artist based similarity may represent some of the vocal characteristics of the artist. However, artists in some genres do not contain vocal sounds, and vocal-focused representation learning has not been extensively studied [43]. For monophonic singing voice, vocal representation learning was attempted using the DAMP dataset (amateur karaoke vocal recordings) [44]. It was extended to joint embedding between mono and mixed audio by an artificial mash-up of the karaoke recordings and instrumental tracks [15]. However, the outcomes are not directly applicable to commercial popular music.

3. METHODS

We present three training models to learn a vocal embedding space. Each model consists of multiple encoder networks for feature extraction from mixed audio or isolated vocal. We first introduce the backbone network common to all encoders and then describe the three training models.

3.1 Backbone Network

The backbone network for the encoders consists of 8 convolutional layers with 128 3-by-3 filters except the first layer with 64 filters and the last layer with 256 filters. Each convolutional layer is followed by a batch normalization, ReLU, and a 2-by-2 max-pooling layer, while the pooling layer for the last convolutional layer is a global average pooling layer. The network takes mel-spectrogram with 128 mel bins from each audio clip after applying short-time Fourier transform with 1,024 samples of Hann window and 512 samples of hop size. The input size of the CNN encoder is 129 frames, which corresponds to a 3-second-long segment at the sampling rate of 22,050 Hz.

3.2 Training Models

Figure 1 illustrates the training models based on metric learning. The goal of metric learning is to learn an embedding space where inputs from the same class are closer to each other than those from different classes. In our setting, we take either mixed audio or vocal as input, use the output of global average pooling layer in the backbone network as the embedding space, and determine if the embedded inputs belong to the same class or not using artist labels (i.e., singer labels). We build the models upon a triplet network that consists of three encoder networks. They take anchor, positive (same class as the anchor) and negative (different class as the anchor) examples as input. The triplet network is often extended to take multiple negative examples for more effective training. Following the previous works [15, 37], we used four negative examples. The

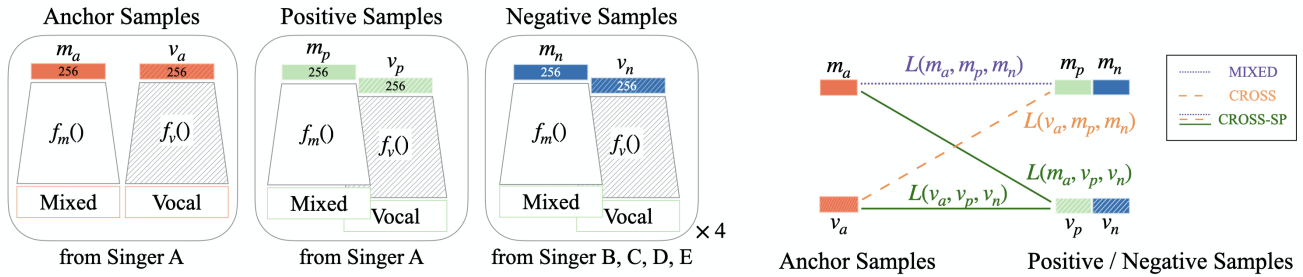


Figure 1. Left: triplet network for metric learning with singer labels. Right: distances used for the loss function in the three models. m and v represent embedding vectors from mixed audio and isolated vocal, respectively. The subscripts (a, p, n) denote for anchor, positive, and negative samples, respectively.

encoders share the weights when they take the same domain of input (vocal or mixed audio). Therefore, we eventually obtain two encoder networks with different weights: one for vocal and the other for mixed audio. In Figure 1, they are denoted as $f_v(\cdot)$ and $f_m(\cdot)$, respectively. Given the common ground, we present three models, differing in the choice of input and the loss function. Each of them is described in detail below.

3.2.1 MIXED

The MIXED model takes only mixed audio as input in the triplet network and thus it uses $f_m(\cdot)$ for feature extraction. This is the baseline model that takes no advantage from vocal source separation. The model was originally proposed as a general music representation learning method using the cost-free artist labels in [37]. The difference in this model is that instrumental music (with no vocals) is excluded in training the model. The triplet loss is formally defined as follows. Let x_a^i , x_p^i and $x_{n_j}^i$ denote the anchor, the positive, and the j -th negative sample of the i -th triplet, respectively, and $m_a^i = f_m(x_a^i)$ denote the embedding vector of input sample x_a^i . Following the previous works [15, 37], we use the hinge rank loss defined as below:

$$L(\text{triplet}^i) = \sum_j [M + d(m_a^i, m_p^i) - d(m_a^i, m_{n_j}^i)] \quad (1)$$

where $d(x, y)$ is given as a cosine distance:

$$d(x, y) = -\frac{x \cdot y}{\|x\|_2 \cdot \|y\|_2} \quad (2)$$

and M is the margin, which was set to 0.4.

3.2.2 CROSS

The CROSS model was proposed to learn the cross-domain embedding between monophonic and mixed music signals [15]. In the triplet network, the anchor takes vocal through $f_v(\cdot)$ and the positive and negative takes mixed audio through $f_m(\cdot)$. Therefore, both vocal and mixed audio from the same class (anchor and positive) are expected to be closed to each other in the embedding spaces. The triplet loss is formally defined as follows. Let y_a^i , y_p^i and $y_{n_j}^i$ denote the vocal counterpart of mixed samples x_a^i , x_p^i and $x_{n_j}^i$ and $v_a^i = f_v(y_a^i)$ denote the embedding vector for vocal. The loss function is defined as follow:

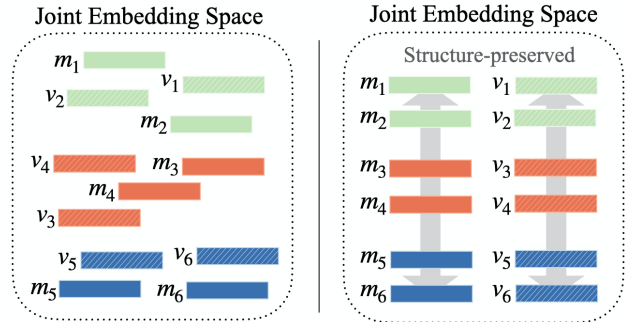


Figure 2. Illustrated examples of cross-domain embedding space when the structure-preserving triplet is not applied (left) and applied (right).

$$L(\text{triplet}^i) = \sum_j [M + d(v_a^i, m_p^i) - d(v_a^i, m_{n_j}^i)] \quad (3)$$

$d(\cdot)$ is the distance defined in Equation 2. Note that the only difference from the MIXED model is the use of embedding vector from vocal source v for anchor samples.

3.2.3 CROSS-SP

The CROSS model effectively learns the cross-domain embedding between mixed audio and its vocal counterpart. However, this does not necessarily learn similarity within the same domain. That is, vocal examples from the same artist or mixed audio examples from the same artist are not enforced to be close to each other in the embedding space. The left column in Figure 2 illustrates a possible distribution of vocal and mixed audio examples. While the pairs of m_i and v_i are closely located by the loss function, the relations among m_i or among v_i can be arbitrary.

This issue has been addressed in the context of cross-modal representation learning where the inputs are image and text [27], or image and audio [32]. They suggested to add constraints to the triplet loss such that the similarity within the same modality is also preserved. Furthermore, they made the loss bi-directional (or symmetric) with regards to two different modalities by having a dual loss where the two triplets take exclusively different modalities of inputs. This setting can be also applied to two different audio domains in our setting: vocal and mixed audio. As a result, we expect that the embedding spaces preserve the structure of similarity in both cross-domains and intra-domain, as illustrated in the right column in Figure

2. We call this model "CROSS-SP" where SP stands for structure-preserving. The structure-preserving triplet loss is defined as a weighted sum of four separate hinge rank loss functions:

$$L(t^i) = \lambda_1 \sum_j [M + d(v_a^i, m_p^i) - d(v_a^i, m_{nj}^i)] \quad (4a)$$

$$+ \lambda_2 \sum_j [M + d(m_a^i, v_p^i) - d(m_a^i, v_{nj}^i)] \quad (4b)$$

$$+ \lambda_3 \sum_j [M + d(m_a^i, m_p^i) - d(m_a^i, m_{nj}^i)] \quad (4c)$$

$$+ \lambda_4 \sum_j [M + d(v_a^i, v_p^i) - d(v_a^i, v_{nj}^i)] \quad (4d)$$

$d(\cdot)$ is the distance defined in Equation 2. Note that all four possible combinations of distances between anchor and positive/negative samples are present: vocal-mix, mix-vocal, mix-mix and vocal-vocal. The first two terms are the bi-directional cross-domain ranking loss [45, 46] and the last two terms are structure-preserving loss [27]. λ_n is a weight for each loss term. The MIXED model can be regarded as a special case where $\lambda_3 = 1$ and others are 0. The CROSS model is also a special case where $\lambda_1 = 1$ and others are 0. In recent studies, the structure-preserving loss terms tend to have a small weight [28] or can be modified to have weak impacts [32]. However, we used 1/4 for all λ_n in the CROSS-SP model, because vocal and mixed audio actually share the same modality and only have different content. An extensive grid search for various combinations of λ_n is left to future research.

4. EXPERIMENTS AND RESULTS

4.1 Dataset and Training Details

We used a filtered version of Million Song Dataset (MSD) [47] for the vocal embedding learning. It contains 4,389 singers with 10 to 20 vocal songs¹. The audio tracks were ensured to include vocal segments using a singing voice detector [48]. For each singer, we used 3 songs for validation, 2 songs for test, and the remaining 5 to 15 songs for training (the test songs were used only for internal evaluation). For vocal source separation, we used the Spleeter vocals/accompaniment separation model from Deezer [8].

To train the vocal embedding models, we used an SGD optimizer with the initial learning rate of 0.01, decay rate of $1e-6$ and the Nesterov momentum. We empirically chose to randomly generate 1200 batches of 25 triplets per epoch. The training is stopped when there is no decrease of validation loss for 20 epochs, and took about 100 epochs.

4.2 Task 1: Singer Identification

We first evaluate the cross-domain vocal embedding for singer identification. The task predicts the correct singer of the query audio among a list of candidate singer models.

¹ We used artist labels in MSD and assumed that artists of songs that contain vocal sounds correspond to "singers" in the experiment.

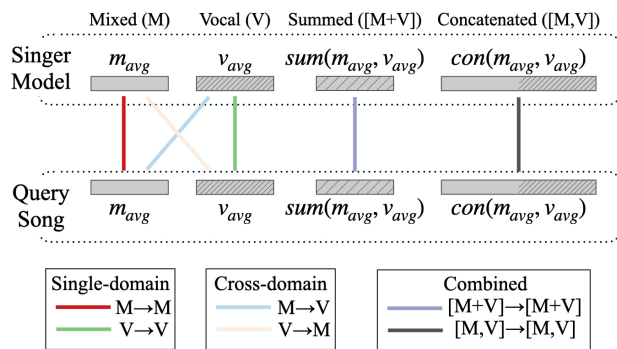


Figure 3. Test scenarios for the singer identification task.

4.2.1 Experiment Settings

We chose 300 singers who have 20 vocal songs from MSD, which are unseen in the training phase. We built the singer models by averaging the embedding vectors from vocal segments of 15 training songs. A query was also computed as an average of the embedding vectors from each of the remaining 5 songs, resulting in a total of 1,500 queries. Since we have two encoders to extract vocal embedding vectors ($f_m(\cdot)$ and $f_v(\cdot)$), singer models and queries can be formed with a different combination. We investigated the following four possibilities for evaluation:

- M: takes mixed audio with $f_m(\cdot)$
- V: takes isolated vocal with $f_v(\cdot)$ but, in the MIXED model, it takes isolated vocal with $f_m(\cdot)$
- [M+V]: takes both isolated vocal and mixed audio with $f_m(\cdot)$ and $f_v(\cdot)$, and computes the sum of the two embedding vectors
- [M, V]: takes both isolated vocal and mixed audio with $f_m(\cdot)$ and $f_v(\cdot)$, and concatenates the two embedding vectors

The entire test scenarios with the different combinations of models and queries are illustrated in Figure 3. They include not only single-domain tests where the singer models and queries are formed from the same encoders (M→M, V→V, [M+V]→[M+V], [M, V]→[M, V]) but also cross-domain tests where the singer models and queries are formed from different encoders (M→V, V→M). We used the cosine distance between the models and queries to identify the singer.

4.2.2 Results

Figure 4 shows the singer identification results given the three training models in the 6 test scenarios. Each of the bar graphs shows top-1 and top-5 accuracy. In general, the CROSS-SP model significantly outperforms the CROSS and MIXED models in most test scenarios. In the single-domain querying tests (M→M, V→V), the CROSS model shows notable improvement over the MIXED model, implying that the encoder for mixed audio, $f_m(\cdot)$, becomes more discriminative when it is jointly trained with the encoder for isolated vocal, $f_v(\cdot)$. This result is not commonly observed in cross-modal embedding research. The difference is presumably attributed to the fact that we use the same modality of audio data although the domains are

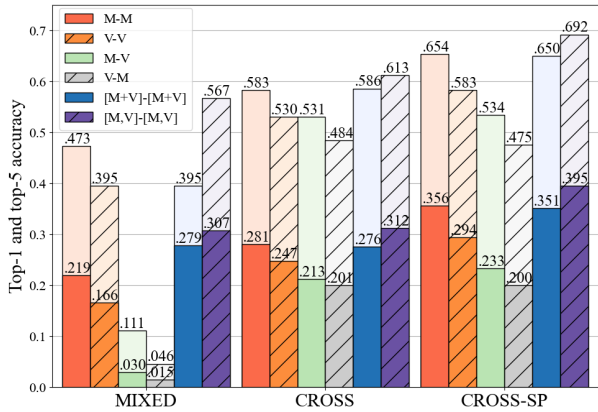


Figure 4. Singer identification result. The filled bars indicate top-1 accuracy while the blank bars indicate top-5 accuracy

Model	Space	R@5	R@10	Pr@5	Pr@10	mAP
MIXED	M	0.1014	0.1519	0.1217	0.0918	0.1176
	V	0.0499	0.0757	0.0598	0.0454	0.0589
	[M+V]	0.1015	0.1515	0.1218	0.0909	0.1200
	[M, V]	0.1139	0.1689	0.1367	0.1013	0.0713
CROSS	M	0.1218	0.1874	0.1462	0.1124	0.1523
	V	0.0971	0.1501	0.1165	0.0901	0.1178
	[M+V]	0.1215	0.1893	0.1458	0.1136	0.1499
	[M, V]	0.1375	0.2010	0.1650	0.1206	0.1672
X-SP	M	0.1617	0.2410	0.1940	0.1446	0.1979
	V	0.1079	0.1689	0.1295	0.1013	0.1350
	[M+V]	0.1625	0.2408	0.1950	0.1445	0.1974
	[M, V]	0.1817	0.2651	0.2180	0.1591	0.2211

Table 1. Results from the query-by-singer task.

different. The single-domain querying tests are improved further in the CROSS-SP model. This validates that the structure-preserving triplet loss improves the arrangement of similar items on both embedding spaces.

In the cross-domain querying tests ($M \rightarrow V$, $V \rightarrow M$), the CROSS and CROSS-SP models show little difference. This indicates that the CROSS-SP model maintains the similarity between the cross-domains despite the additional loss terms. On the other hand, the MIXED model shows poor performance. This is expected because the model was not trained to handle isolated vocals and mixed at the same time.

In the combination querying tests, we can observe an interesting result that the concatenation of the two embedding vectors consistently increases the accuracy in all models. This indicates that musical sounds other than isolated vocals provide additional information to identify singers. This makes sense in that artists are often associated with a particular style or genre of music. In the meantime, the sum of the two embedding vectors did not help improving the accuracy in all models.

4.3 Task 2: Query-by-Singer

A follow-up task using the vocal embeddings is to retrieve songs with the singer information of a query song.

4.3.1 Experiment Settings

We used the same 300 singers from the singer identification task above. For each singer, we chose 6 songs to include in the dataset to be retrieved and 4 songs as queries.

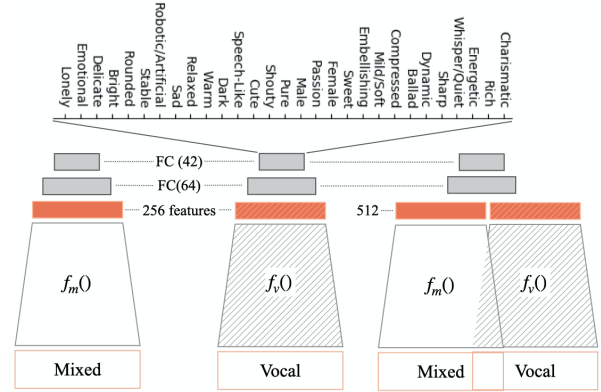


Figure 5. The model structure for the vocal tagging task using transfer learning with the vocal embedding models.

This results in 1800 songs in the search space and 1200 queries. We represented all queries and retrieved songs as an average of the vocal embedding vectors, and calculated the similarity using the cosine distance.

We evaluated the models using Recall-at- k ($R@k$), Precision-at- k ($Pr@k$) and mean average precision (mAP). $R@k$ represents how many songs among the relevant songs are retrieved, which is songs from the same artist in this case. It is the ratio of the number of relevant songs in top- k similar songs over all relevant songs, in our case, 6. $Pr@k$ is a metric which shows how many items are relevant among top- k similar songs. For both metrics, we used 5 and 10 for k . The last metric is mAP, which counts the rank of every relevant song. We tested mixed only embedding space (M), isolated vocal embedding space (V) and concatenated space ([M, V]) for all three strategies. For the MIXED strategy, the same mixed encoder is used for processing isolated vocal too.

4.3.2 Results

Table 1 summarizes the results with the three metrics. The general trend is similar to that in the singer identification task; the vocal embedding space from mixed audio (M) slightly outperforms that from isolated vocals (V) in all three models, and the concatenated embedding space ([M, V]) improves the performance further. Among the three models, CROSS-SP performs the best.

4.4 Task 3: Transfer Learning to Vocal Tagging

In this task, we evaluate the generalization capability of the vocal embedding models trained in the previous section for a downstream task. For this purpose, we used the K-pop Vocal Tag (KVT) dataset designed for music auto-tagging focusing on singing voice [11]. It consists of 6,787 vocal segments from K-pop music tracks. They are annotated with 42 semantic tags which describe various vocal characteristics in the categories of pitch range, timbre, playing techniques, and gender. A subset of the tag labels are shown in Figure 5. Since the vocal tags are also associated with different styles and identities of singers, we hypothesize that the pre-trained vocal embedding space will be useful for the vocal tagging task and thus the transfer learning is effective.

Models	Space	AUC	F1	Prec.	Recall
Baseline	-	0.7116	0.7198	0.6132	0.7661
MIX	M	0.7338	0.7393	0.6495	0.8013
CROSS	M	0.7336	0.7340	0.6459	0.8046
CROSS	V	0.7406	0.7392	0.6511	0.8007
CROSS	[M, V]	0.7449	0.7431	0.6531	0.8052
CROSS-SP	M	0.7376	0.7383	0.6457	0.8054
CROSS-SP	V	0.7401	0.7414	0.6575	0.7751
CROSS-SP	[M, V]	0.7529	0.7469	0.6617	0.8186

Table 2. Result from the vocal tagging task using transfer learning with the vocal embedding models.

4.4.1 Experiment Settings

Figure 5 depicts the transfer learning setting for vocal tagging. We first extract embedding vectors using two pre-trained encoders. The mixed audio encoder, $f_m(\cdot)$, is obtained from all three models (MIX, CROSS and CROSS-SP) whereas the isolated vocal encoder, $f_v(\cdot)$, is from the CROSS and CROSS-SP models only. Either one or the concatenation of the embedding vectors is used as an input feature vector of a classifier that consists of two fully connected layers (64 and 42 units) with the ReLU activation. The output layer takes the sigmoid function for multi-label classification. The classifier head is trained with the binary cross-entropy loss between the sigmoid predictions and the tag labels.

We compared the transfer learning settings with a simple CNN model trained with the KVT dataset from scratch as a baseline. The baseline model is a modified version of the CNN model in the original study of the KVT dataset [11]. The main difference is that the input segment size changes from 129 frames to 107 frames to match the baseline model to the transfer learning settings.

4.4.2 Results

Table 2 summarizes the vocal tagging accuracy measured with AUC, F1 score, precision and recall. Compared to the baseline model, the training learning models show increased performances in all metrics. This indicates that the vocal embedding learned with MSD generalizes to K-pop music vocals even though if the music genre and the target task are different. In terms of input audio domain, isolated vocals is more effective than mixed audio as shown in the CROSS model (M and V) and CROSS-SP models (M and V). This is contrasted to the results in two previous tasks (singer identification and query-by-singer), presumably because the vocal tagging task requires detailed information about timbre and singing techniques which is mainly found in vocal sounds. The results also show that the concatenated embedding vector ([M, V]) further improves the tag predictions. In particular, the CROSS-SP model achieves the best performance. This result confirms that the structure-preserving triplet-loss helps generalization in learning vocal embedding.

4.5 Task 4: Transfer Learning to Artist Classification

Lastly, we conduct artist classification using the artist20 dataset [49]. The transfer learning setting is identical to the task 3 except that the last layer is the softmax unit with 20 outputs that correspond to 20 artists. Similar to [49],

Models	Space	Eval. Level	Accuracy	F1
GMM [49]		Frame	0.590	
GMM [50]*		Frame	0.541	
CRNN [51]		Segment		0.527
CROSS-SP	M	Segment	0.596	0.550
CROSS-SP	V	Segment	0.500	0.496
CROSS-SP	[M+V]	Segment	0.638	0.587
CROSS-SP	[M, V]	Segment	0.638	0.576
SVM [50]*		Song	0.687	
CRNN [51]		Song		0.653
i-Vector [52]		Song	0.8545	0.8459
CROSS-SP	M	Song	0.772	0.753
CROSS-SP	V	Song	0.894	0.891
CROSS-SP	[M+V]	Song	0.815	0.804
CROSS-SP	[M, V]	Song	0.806	0.795

Table 3. Comparison of artist classification using the artist20 dataset (* [50] used 18 artists.)

we report average performance from 6-fold cross validation. Table 3 compares the CROSS-SP model to the baseline [49, 50] and recent works [51, 52]. All of them used album-level train-test split, which tends to be more challenging than song-level train-test split [51]. For comparison, we evaluated the input in two duration levels; One is segment-level (3 seconds) and the other is song-level. The song-level prediction was obtained by averaging the softmax activations from segment-level inputs through each song. The results show that the CROSS-SP model outperforms all previous works. In segment-level evaluation, the summed ([M+V]) and concatenated ([M, V]) embedding spaces are better than individual embedding spaces. In song-level evaluation, on the other hand, using only vocal embedding space (V) is better. This contradictory result can be explained by the consistency of identifiable information in vocal embedding space, which can restrain noisier information from background music.

5. CONCLUSIONS

We presented a method to learn cross-domain embedding spaces between isolated vocals and mixed audio by leveraging the state-of-the-art music source separation algorithm. We show that the structure-preserving triplet-loss used in training the deep neural networks greatly improves the generalization capability when the embedding vectors are used for singer identification, query-by-singer and vocal tagging. Also, we showed the concatenation of the two vocal embedding vectors from isolated vocals and mixed audio are more effective in the tasks. This indicates that unique genres or styles of artists are reflected on their instrumental sounds and thus mixed audio is complementary to vocal sounds. We expect to extend the use of cross-domain vocal embedding to various music applications such as singer-focused music recommendation, vocal-to-music cross retrieval and other vocal-centric MIR tasks. We demonstrate an example at this link as a potential use case².

²We implemented a vocal-to-accompaniment matching system using the cross-domain vocal embedding vector. The demo audio examples are found at <https://khlukekim.github.io/crossdomainembedding/>

6. REFERENCES

- [1] A. Demetriou, A. Jansson, A. Kumar, and R. M. Bitner, "Vocals in music matter: the relevance of vocals in the minds of listeners," in *Proc. ISMIR*, 2018.
- [2] E. Zangerle, M. Vötter, R. Huber, and Y.-H. Yang, "Hit song prediction: Leveraging low-and high-level audio features," in *Proc. ISMIR*, 2019, pp. 319–326.
- [3] Y. Kim and B. Whitman, "Singer identification in popular music recordings using voice coding features," in *Proc. ISMIR*, 2002.
- [4] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.
- [5] R. Nishikimi, E. Nakamura, S. Fukayama, M. Goto, and K. Yoshii, "Automatic singing transcription based on encoder-decoder recurrent neural networks with a weakly-supervised attention mechanism," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 161–165.
- [6] J. Song, S. Y. Bae, and K. Yoon, "Mid-level music melody representation of polyphonic audio for query-by-humming system," in *Proc. ISMIR*, 2002.
- [7] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.
- [8] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.
- [9] S. Kum, J.-H. Lin, L. Su, and J. Nam, "Semi-supervised learning using teacher-student models for vocal melody extraction," in *Proc. ISMIR*, 2020.
- [10] Y. Gao, X. Zhang, and W. Li, "Vocal melody extraction via hrnet-based singing voice separation and encoder-decoder-based f0 estimation," *Electronics*, vol. 10, no. 3, p. 298, 2021.
- [11] K. L. Kim, J. Lee, S. Kum, C. L. Park, and J. Nam, "Semantic tagging of singing voices in popular music recordings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1656–1668, 2020.
- [12] T.-H. Hsieh, K.-H. Cheng, Z.-C. Fan, Y.-C. Yang, and Y.-H. Yang, "Addressing the confounds of accompaniments in singer identification," in *Proc. ICASSP*, 2020, pp. 1–5.
- [13] B. Sharma, R. K. Das, and H. Li, "On the importance of audio-source separation for singer identification in polyphonic music," in *Proc. INTERSPEECH*, 2019, pp. 2020–2024.
- [14] M. Martinez Ramirez, D. Stoller, and D. Moffat, "A deep learning approach to intelligent drum mixing with the wave-u-net." Audio Engineering Society, 2021.
- [15] K. Lee and J. Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice for singing voice," in *Proc. ISMIR*, 2019.
- [16] A. Mesaros, T. Virtanen, and A. Klapuri, "Singer identification in polyphonic music using vocal separation and pattern recognition methods," in *Proc. ISMIR*, 2007, pp. 375–378.
- [17] H. Fujihara, M. Goto, T. Kitahara, and H. G. Okuno, "A modeling of singing voice robust to accompaniment sounds and its application to singer identification and vocal-timbre-similarity-based music information retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 638–648, 2010.
- [18] M. Lagrange, A. Ozerov, and E. Vincent, "Robust singer identification in polyphonic music using melody enhancement and uncertainty-based learning," in *Proc. ISMIR*, 2012.
- [19] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix-a reference implementation for music source separation," *Journal of Open Source Software*, 2019.
- [20] A. Jansson, E. J. Humphrey, N. Montecchio, R. M. Bitner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," in *Proc. ISMIR*, 2017.
- [21] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [22] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proc. ICML workshop on unsupervised and transfer learning*, 2012, pp. 17–36.
- [23] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. NIPS*, vol. 27, 2014.
- [24] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*. Springer, 2015, pp. 84–92.
- [25] N. C. Mithun, R. Panda, E. E. Papalexakis, and A. K. Roy-Chowdhury, "Webly supervised joint embedding for cross-modal image-text retrieval," in *Proc. ACM*

- International Conference on Multimedia*, 2018, pp. 1856–1864.
- [26] A. Salvador, N. Hynes, Y. Aytar, J. Marin, F. Ofli, I. Weber, and A. Torralba, “Learning cross-modal embeddings for cooking recipes and food images,” in *Proc. CVPR*, 2017, pp. 3068–3076.
- [27] L. Wang, Y. Li, and S. Lazebnik, “Learning deep structure-preserving image-text embeddings,” in *Proc. CVPR*, 2016, pp. 5005–5013.
- [28] L. Wang, Y. Li, J. Huang, and S. Lazebnik, “Learning two-branch neural networks for image-text matching tasks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 394–407, 2019.
- [29] N. C. Mithun, J. Li, F. Metzger, and A. K. Roy-Chowdhury, “Learning joint embedding with multimodal cues for cross-modal video-text retrieval,” in *Proc. ACM on International Conference on Multimedia Retrieval*, 2018, p. 19–27.
- [30] A. Nagrani, S. Albanie, and A. Zisserman, “Learnable pins: Cross-modal embeddings for person identity,” in *Proc. ECCV*, 2018.
- [31] D. Surís, A. Duarte, A. Salvador, J. Torres, and X. Giró-i Nieto, “Cross-modal embeddings for video and audio retrieval,” in *Proc. ECCV Workshops*, 2019, pp. 711–716.
- [32] S. Hong, W. Im, and H. S. Yang, “Cbvmr: Content-based video-music retrieval using soft intra-modal structure constraint,” in *Proc. ACM on International Conference on Multimedia Retrieval*, 2018, p. 353–361.
- [33] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse, “Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network,” *Computer Vision and Image Understanding*, vol. 164, pp. 27–37, 2017.
- [34] F. Wang, L. Kang, and Y. Li, “Sketch-based 3d shape retrieval using convolutional neural networks,” in *Proc. CVPR*, 2015, pp. 1875–1883.
- [35] N. Khurshid, T. Hanif, M. Tharani, and M. Taj, “Cross-view image retrieval-ground to aerial image retrieval through deep learning,” in *International Conference on Neural Information Processing*. Springer, 2019, pp. 210–221.
- [36] R. Lu, K. Wu, Z. Duan, and C. Zhang, “Deep ranking: Triplet matchnet for music metric learning,” in *Proc. ICASSP*, 2017, pp. 121–125.
- [37] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, “Representation learning of music using artist labels,” in *Proc. ISMIR*, 2018.
- [38] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, “Metric learning vs classification for disentangled music representation learning,” in *Proc. ISMIR*, 2020.
- [39] J. Choi, J. Lee, J. Park, and J. Nam, “Zero-shot learning for audio-based music classification and tagging,” in *Proc. ISMIR*, 2019.
- [40] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, “Multimodal metric learning for tag-based music retrieval,” in *Proc. ICASSP*, 2020.
- [41] B. McFee, L. Barrington, and G. R. Lanckriet, “Learning similarity from collaborative filters,” in *Proc. ISMIR*, 2010, pp. 345–350.
- [42] J. Lee, J. Park, and J. Nam, “Representation learning of music using artist, album, and track information,” in *Machine Learning for Music Discovery Workshop, ICML*, 2019.
- [43] J. Park, D. Kim, J. Lee, S. Kum, and J. Nam, “A hybrid of deep audio feature and i-vector for artist recognition,” in *Joint Workshop on Machine Learning for Music, ICML*, 2018.
- [44] C.-i. Wang and G. Tzanetakis, “Singing style investigation by residual siamese convolutional neural networks,” in *Proc. ICASSP*, 2018, pp. 116–120.
- [45] R. Kiros, R. Salakhutdinov, and R. Zemel, “Multimodal neural language models,” in *Proc. ICML*, vol. 32, no. 2, 2014, pp. 595–603.
- [46] A. Karpathy, A. Joulin, and L. Fei-Fei, “Deep fragment embeddings for bidirectional image sentence mapping,” in *Proc. NIPS*, 2014, p. 1889–1897.
- [47] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *Proc. ISMIR*, 2011.
- [48] S. Kum and J. Nam, “Joint detection and classification of singing voice melody using convolutional recurrent neural networks,” *Applied Sciences*, vol. 9, no. 7, p. 1324, 2019.
- [49] D. P. Ellis, “Classifying music audio with timbral and chroma features,” 2007.
- [50] M. I. Mandel and D. P. Ellis, “Song-level features and support vector machines for music classification,” 2005.
- [51] Z. Nasrullah and Y. Zhao, “Music artist classification with convolutional recurrent neural networks,” in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.
- [52] H. Eghbal-Zadeh, B. Lehner, M. Schedl, and G. Widmer, “I-vectors for timbre-based music similarity and music artist classification,” in *Proc. ISMIR*, 2015, pp. 554–560.

DECOUPLING MAGNITUDE AND PHASE ESTIMATION WITH DEEP ResUNet FOR MUSIC SOURCE SEPARATION

Qiuqiang Kong¹, Yin Cao², Haohe Liu¹, Keunwoo Choi¹, Yuxuan Wang¹

¹ByteDance ²University of Surrey

¹{kongqiuqiang, liuhaohhe.7, keunwoo.choi, wangyuxuan.11}@bytedance.com

²yin.cao@surrey.ac.uk

ABSTRACT

Deep neural network based methods have been successfully applied to music source separation. They typically learn a mapping from a mixture spectrogram to a set of source spectrograms, all with magnitudes only. This approach has several limitations: 1) its incorrect phase reconstruction degrades the performance, 2) it limits the magnitude of masks between 0 and 1 while we observe that 22% of time-frequency bins have ideal ratio mask values of over 1 in a popular dataset, MUSDB18, 3) its potential on very deep architectures is under-explored. Our proposed system is designed to overcome these. First, we propose to estimate phases by estimating complex ideal ratio masks (cIRMs) where we decouple the estimation of cIRMs into magnitude and phase estimations. Second, we extend the separation method to effectively allow the magnitude of the mask to be larger than 1. Finally, we propose a residual UNet architecture with up to 143 layers. Our proposed system achieves a state-of-the-art MSS result on the MUSDB18 dataset, especially, a SDR of 8.98 dB on vocals, outperforming the previous best performance of 7.24 dB. The source code is available at: https://github.com/bytedance/music_source_separation.

1. INTRODUCTION

Music source separation (MSS) is a task to separate audio mixtures into individual sources such as vocals, drums, accompaniment, etc. MSS is an important topic for music information retrieval (MIR) since it can be used for several downstream MIR tasks including melody extraction [1], pitch estimation [2], music transcription [3], music remixing [4], and so on. MSS also has several direct applications such as Karaoke and music remixing.

MSS methods can be categorized into signal processing based methods and neural network based methods. Several methods have been proposed for source separation such as

non-negative matrix factorizations (NMFs) [5]. NMF decomposes a spectrogram into dictionaries and activations, and separated sources can be obtained by multiplying activations with different dictionaries. Sparse coding was used in [6], where audio signals are transformed into sparse representations for source separation. Independent component analysis (ICA) was used in [7] by assuming that source signals are statistically independent. Other unsupervised source separation methods include modeling average harmonic structures in [8]. Recently, neural network based methods became popular and have achieved state-of-the-art results in the MSS task. Those models include fully connected neural networks [9], recurrent neural networks [10,11], convolutional neural networks [12–18], and time-domain separation models [19–22].

First, several previously introduced MSS systems perform in the time-frequency domain and have achieved the state-of-the-art performance. However, many conventional spectrogram-based systems do not estimate the phases of separated sources [12–16] and it upper bounds performance of MSS systems as we will show in this paper. Recently, several works were proposed to estimate the phases of clean sources. For example, PhaseNet [23] treats the phase estimation as a phase classification problem, and PHASEN [24] estimates the phase of clean sources using a separate neural network. Complex ideal ratio masks (cIRM) [25–27] were also used for MSS. However, directly predicting the real and imaginary parts of cIRMs can be difficult, because the real and imaginary parts are sensitive to signal shifts in the time domain. In this paper, we propose to decouple the magnitude and phase for estimating cIRMs, which increases the performance of the source separation systems. We also elaborately design the magnitude estimation submodule to increase the upper bound of MSS systems.

Second, several magnitude or complex mask-based methods [18] usually limit the magnitude of masks to 1. Based on our analysis, this limits the upper bound of the performance of MSS systems. In this work, we observe that 22% time-frequency bins in the cIRM have magnitudes larger than 1. To predict magnitudes with cIRMs larger than 1, we propose to combine the predictions of mask and spectrogram where the spectrogram term is a residual component to complement the mask prediction term. Therefore, we combine the advantage of mask and linear spectrogram based methods. All of mask magni-



tudes, and spectrograms and phases are learnt by a neural network.

Third, we show that the previous UNets [12, 16, 16, 18] with up to tens of layers have limited separation results in MSS. We show that the depth of neural networks are important for the MSS task. In this work, we propose a deep residual UNet with 143 layers. We propose using residual encoder blocks, residual intermediate layers, and residual decoder blocks to build the 143-layer residual UNet. We show that deep architectures significantly increase the MSS performance.

This paper is organized as follows. Section 2 introduces previous neural network based source separation systems and their limitations. Section 3 introduces our proposed system including the estimation of cIRMs and deep residual UNet. Section 4 shows experimental results and Section 5 concludes this work.

2. BACKGROUNDS

In this paper, we denote the time-domain signal of a mixture and a clean source as $x \in \mathbb{R}^L$ and $s \in \mathbb{R}^L$, respectively, where L is the number of samples of the signal. Their short-time Fourier transforms (STFTs) are denoted as $X \in \mathbb{R}^{T \times F}$ and $S \in \mathbb{R}^{T \times F}$, respectively. T and F correspond to the number of frames and frequency bins. Next, we describe several source separation methods.

2.1 Approach 1: Direct Magnitude Prediction

In direct prediction approaches, a MSS system directly learns a mapping from $|X|$ to $|S|$, i.e., $|\hat{S}| = f(|X|)$. Here, f can be any function approximator, such as a neural network of the fully connected, convolutional or recurrent types.

Typically, a direct prediction method does not estimate the phases of separated sources. Instead, the phases of mixture is used to recover the STFT of separated sources:

$$\hat{S} = |\hat{S}|e^{j\angle X}, \quad (1)$$

where $\angle X \in [-\pi, \pi]^{T \times F}$ is the phase of X . Finally, we apply an inverse STFT \mathcal{F}^{-1} on \hat{S} to obtain the separated waveform $\hat{s} = \mathcal{F}^{-1}(\hat{S})$.

2.2 Approach 2: Magnitude Mask

In magnitude mask-based approaches, in order to perform source separation, a system predicts a mask, $|\hat{M}| \in \mathbb{R}_{\geq 0}^{T \times F}$, that is applied to the input spectrogram element-wisely.

$$|\hat{S}| = |\hat{M}| \odot |X|, \quad (2)$$

The values of \hat{M} can be continuous in the case of using ideal ratio masks (IRMs). The range of IRMs is often bounded between $[0, 1]$, assuming that the magnitudes of individual sources are smaller than the magnitudes of mixture. Furthermore, the magnitude is assumed to be either 0 or 1 in the case of ideal binary mask (IBM).

Similar to direct magnitude prediction, in magnitude mask-based methods, the phase of original mixture is used as an approximation of the phase of the separated sources.

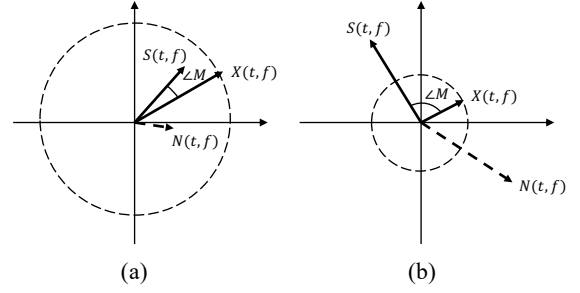


Figure 1. Illustrations of a source signal s , a noise n , and mixture x on a complex plain. (a) is an example when $|M(t, f)|$ smaller than 1 and (b) is an example when $|M(t, f)|$ larger than 1.

2.3 Approach 3: Complex Mask

Accurate phase estimation becomes critical as the performance of the systems in Section 2.1 and 2.2 has improved. Because of that, several works were proposed recently to take the phase estimation into consideration in the model. One ambitious approach is to directly predicting the complex STFT, as an extension of the direct magnitude prediction towards phase [27]. However, accurate prediction of a complex STFT is challenging because the estimation of real and imaginary parts of a complex STFT is more difficult than the estimation of the magnitude. As an alternative, many methods have been introduced to predict complex masks of mixture STFT [25–27]. In PhaseNet [23] and PHASEN [24], the authors proposed to predict the phases of signals independently from the magnitudes.

2.4 Out-of-Phase and Masks

In this work, we adopt cIRM-based methods for source separation due to their superior performance in phase estimation. A complex mask $M \in \mathbb{C}^{T \times F}$ is calculated by:

$$\begin{aligned} M &= S/X \\ &= \frac{S_r + iS_i}{X_r + iX_i} \\ &= \frac{S_r X_r + S_i X_i + i(S_i X_r - S_r X_i)}{X_r^2 + X_i^2}, \end{aligned} \quad (3)$$

where X_r, S_r are real parts of X and S respectively, and X_i, S_i are imaginary parts of X and S respectively. The perfect separation of S from X can be obtained by:

$$\begin{aligned} S &= MX \\ &= |M||X|e^{j(\angle M + \angle X)}. \end{aligned} \quad (4)$$

Equation (4) shows that the separation of S from X includes a magnitude scaling and a phase rotation operation. The magnitude of cIRM ($|M|$) controls how much the magnitude of X should be scaled, and the angle of cIRM ($\angle M$) controls how much the angle of X should be rotated.

We now introduce an additive noise model, i.e., $X = S + N$, which is illustrated in Fig. 1. Here, we focus on each time-frequency bin of STFTs of source, noise,

Table 1. The empirical upper bounds of MSS systems on MUSDB18. ‘acc.’ indicates accompaniment. On the top row, numbers indicate the limit of the magnitude masks.

	Mixture	IBM	IRM (1)	IRM (inf)	cIRM (1)	cIRM (2)	cIRM (5)	cIRM (10)	cIRM (inf)
vocals	-5.69	10.59	10.04	10.42	19.84	31.02	41.04	47.62	54.50
acc.	-5.68	16.10	15.31	15.97	26.54	37.62	47.33	53.51	60.63
bass	-6.36	7.17	6.05	6.07	17.99	27.88	37.86	44.30	54.12
drums	-4.30	8.75	8.03	8.61	19.10	30.38	39.91	46.45	56.08
other	-4.92	8.20	7.28	7.37	18.97	28.91	39.08	45.64	56.00

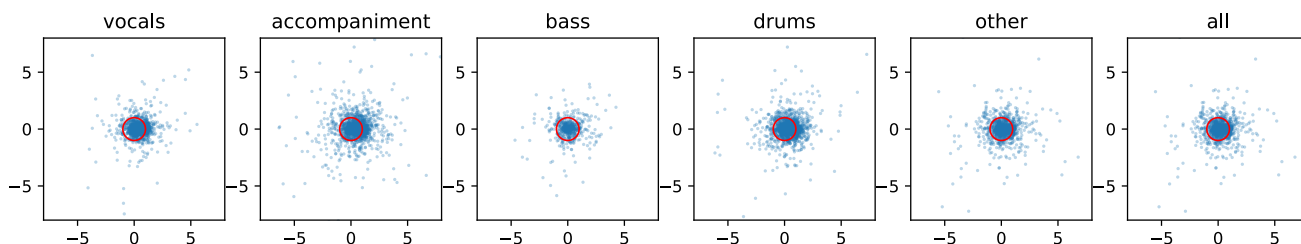


Figure 2. cIRMs of vocals, accompaniment, bass, drums, other, and all sources, on the complex 2D plain. Unit circles are drawn in red.

and mixture ($S(t, f)$, $N(t, f)$, and $X(t, f)$, respectively). Fig. 1 (a) shows an example where the magnitude of cIRM $|M(t, f)|$ is smaller than 1. This is modelled well in the existing methods where the ranges of complex mask is bound to $[0, 1]$. However, as illustrated in Fig. 1 (b), $|M(t, f)|$ can be larger than 1. As in the figure, this may happen when $S(t, f)$ and $N(t, f)$ are out of phase, since that makes the magnitude of mixture to be smaller than that of (individual) signal.

2.5 Empirical analysis of the effect of bounded magnitude mask

In this section, we empirically investigate the upper bound of the performance when the magnitude mask is bounded to be < 1 , the common assumption in many previous methods. We use signal-to-distortion ratio (SDR) [28] as an evaluation metric, which is defined as follow:

$$SDR(s, \hat{s}) = 10 \log_{10} \frac{\|s\|^2}{\|\hat{s} - s\|^2}. \quad (5)$$

A higher SDR indicates better separation results, and vice versa. Ideally, a perfect separation will lead to infinite SDR. We evaluate the upper bound of systems on the vocals, accompaniment, bass, drums and other instruments from the MUSDB18 dataset [29].

The first column of Table 1 shows the SDRs of using the mixture without separation as separated sources. The second column (IBM) shows the upper bound of the performance when IBMs are used. According to the third column (IRM (1)), using IRMs whose magnitudes are bounded in $[0, 1]$ has slightly lower upper bounds compared to those of IBM. IRM uses the phases of mixture but not the phases of clean sources for separating sources so the upper bound SDR is limited. Unbounded IRM (IRM (0, inf), the fourth column) shows a small improvement over bounded IRM, but not significantly.

Compared to IBM, IRM (1) and IRM (inf), the five cIRM columns show that the upper bounds are significantly higher when correct phase information is used. The upper bounds of cIRM (1) is higher than IRM (1) by around 10 dB. The improvement within cIRMs is also dramatic – only by increasing the limit from cIRM (1) to cIRM (2), the upper bounds increase by more than 9.89 dB for all the instruments. When magnitude mask is unbounded, the SDR of cIRM (inf) is infinite in theory. Considering the numerical stability when calculating SDRs, we add a small ϵ to the denominator of (5). We observed the SDRs of cIRM (inf) are higher than 50 dB for all the instruments.

2.6 Distribution of cIRMs

In this section, we visualize the distribution of cIRMs to show that there are much space to improve previous MSS systems. Fig. 2 shows the cIRMs of vocals, accompaniment, bass, drums, other and all sources. The horizontal and vertical axes show the real and imaginary parts of cIRMs calculated by (3), where each point in Fig. 2 corresponds to a $M(t, f)$. The unit circle shown in Fig. 2 corresponds to masks with magnitude values equal to 1. Fig 2 from left to right shows the cIRM distribution of vocals, accompaniment, bass, drums, other instruments, and all sources. It can be seen that there are many cIRMs that having magnitudes larger than 1. The ratio of cIRMs have magnitudes larger than 1 for vocals, accompaniment, bass, drums and others are 20.3%, 34.5%, 6.1%, 26.9% and 13.9% respectively. Along with the analysis in Section 2.5, this observation motivates our work to extend the bounded mask estimation methods to unbound mask estimation methods.

Fig. 2 also shows that, the phases of cIRMs distribute evenly in all directions. However, spectrogram-based methods assume that the phases of cIRMs are all 0.

This observation further justifies to predict the phases in a MSS system.

3. PROPOSED SYSTEM

In this section, we propose a MSS system that incorporates phase estimation that is based on the proposed decoupling of magnitude and phase (Section 3.1). Furthermore, to overcome the limit of bounded magnitude mask as discussed in Section 2, we propose a modification to extend the mask estimation method that allows the magnitude of the resulting mask be larger than 1 (Section 3.2). Finally, we propose a deep Residual UNet with 143 layers, which is the first MSS architectures that is deeper than a hundred layers (Section 3.3). All the proposed systems are trained with a L1-loss that is computed on the waveform domain as illustrated in Figure 3.

3.1 Decoupling Magnitude and Phase for cIRM Estimation

Unlike previous works that directly predict real and imaginary parts of masks [18, 25], we propose to decouple the magnitude and phase estimation for MSS so that we can optimize their designs separately. We denote the complex mask to estimate as $\hat{M} \in \mathbb{C}^{T \times F}$. As a part of the solution, our system outputs a bounded magnitude mask $\hat{M}_{\text{mag}} \in \mathbb{R}^{T \times F}$ whose value is in $[0, 1]$. In practice, it is implemented by applying sigmoid function. Our system also outputs two more tensors, $\hat{P}_r \in \mathbb{R}^{T \times F}$ and $\hat{P}_i \in \mathbb{R}^{T \times F}$. Here, \hat{P}_r and \hat{P}_i are real and imaginary parts of \hat{M} , respectively. Then, instead of calculating the angle $\angle \hat{M}$ directly, we calculate its cosine value $\cos \angle \hat{M}$ and sine value $\sin \angle \hat{M}$ using \hat{P}_r and \hat{P}_i as follows:

$$\begin{aligned} \cos \angle \hat{M} &= \hat{P}_r / \sqrt{\hat{P}_r^2 + \hat{P}_i^2} \\ \sin \angle \hat{M} &= \hat{P}_i / \sqrt{\hat{P}_r^2 + \hat{P}_i^2}. \end{aligned} \quad (6)$$

Then, we estimate the real and imaginary parts of cIRM by:

$$\begin{aligned} \hat{M}_r &= \hat{M}_{\text{mag}} \cos \angle \hat{M} \\ \hat{M}_i &= \hat{M}_{\text{mag}} \sin \angle \hat{M} \end{aligned} \quad (7)$$

The cIRM $\hat{M} = \hat{M}_r + j\hat{M}_i$ is a complex tensor, and is used to separate a target source from X by (4) which involves a magnitude scaling and a phase rotation operation. Finally, we apply an inverse STFT to obtain the separated waveform.

3.2 Combination of Bounded Mask Estimation and Direct Magnitude Prediction

In previous works we show that directly predicting the unbound linear magnitude $|\hat{S}|$ lead to the underperformance of the source separation system. To overcome the limit of the performance discussed in Section 2, we propose to combine a bounded mask and direct magnitude prediction to estimate the magnitude of cIRMs. The motivation is to

use direct magnitude prediction as *residual components*, one that complements the bounded magnitude mask. This is implemented as follow:

$$|\hat{S}| = \text{relu}(\hat{M}_{\text{mag}} \odot |X| + \hat{Q}) \quad (8)$$

where $\hat{Q} \in \mathbb{R}^{T \times F}$ is the direct magnitude prediction. In this way, we take the advantages of both of the methods. The ReLU operation ensures that the predicted magnitude is always larger than 0. The estimation of phase $\angle \hat{M}$ by using \hat{P}_r and \hat{P}_i are the same as the one in Section 3.1. Then, the separated STFT can be obtained by:

$$\hat{S} = |\hat{S}| e^{j(\angle \hat{M} + \angle X)}, \quad (9)$$

where $|\hat{S}|$ is calculated by Eq. (8).

In total, the our proposed MSS system contains four outputs: \hat{M}_{mag} , \hat{Q} , \hat{P}_r and \hat{P}_i . All of those outputs share the same backbone architecture and apply an individual linear layer to obtain their outputs. We use sigmoid non-linearity to predict \hat{M}_{mag} to ensure they have values between 0 and 1. Fig. 3 shows the structure of our proposed method.

3.3 Residual UNet

In this section, we introduce deep residual UNets with hundreds of layers for MSS, which is at least 4 times deeper than previous UNet models [12, 16, 18].

We first introduce a baseline UNet with 33 layers. The 33-layer UNet consists of 6 encoder and 6 decoder layers. Each encoder layer consists of two convolutional layers and a downsampling layer. Each decoder layer consists of one transposed convolutional layer for upsampling and two convolutional layers. Finally, three additional convolutional layers are added after decoder layers. In total, there are 33 convolutional layers.

Next, we introduce a 143-layer residual UNet. In building a residual UNet with hundreds of layers, we use residual encoder blocks (REB) and residual decoder blocks (RDB) to increase its depth. Fig. 3 shows the architecture of our proposed residual UNet where we use 6 REBs and 6 RDBs. Each REB consists of 4 residual convolutional blocks (RCB) as shown in Fig. 4 (a). Each RCB consists of two convolutional layers with kernel sizes 3×3 as shown in Fig. 4 (c). A shortcut connection is added between the input and the output of a RCB. A batch normalization [30] and a leaky ReLU non-linearity [31] with a negative slope of 0.01 is applied before convolutional layers following the pre-act residual network configuration [32]. An 2×2 average pooling layer is applied after each REB to reduce the feature map size. Each REB consists of 8 convolutional layers.

The blocks in the decoder (RDBs) are symmetric to those in the encoder (REB). Each RDB consists of a transposed convolutional layer with a kernel size 3×3 and stride 2×2 to upsample feature maps, followed by four RCBs as shown in Fig. 4 (b). Each RDB consists of 9 convolutional layers, including 8 convolutional layers and 1 transposed convolutional layer. To further increase the representation ability of the residual UNet, we introduce

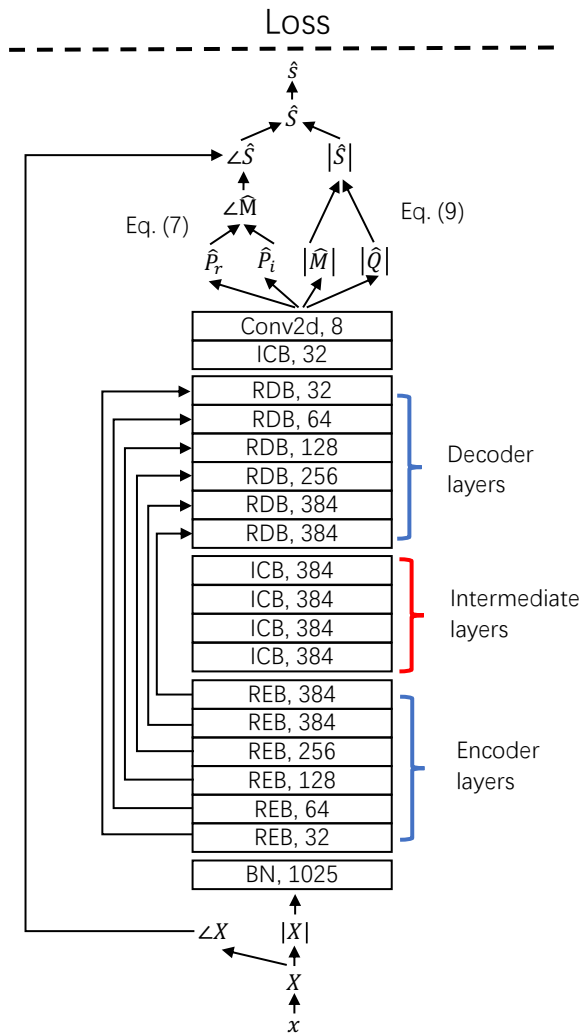


Figure 3. The proposed MSS system with residual blocks. The details of REB, RDB, and RCB are illustrated in Figure 4.

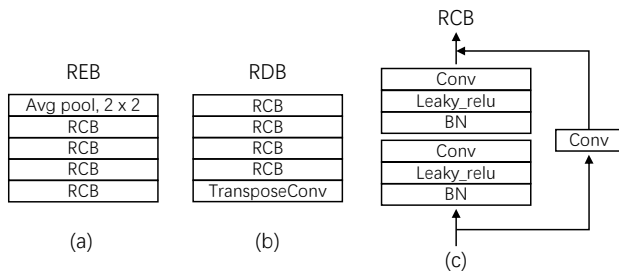


Figure 4. (a) Residual encoder block (REB), (b) residual decoder block (RDB), (3) residual convolutional block (RCB).

intermediate convolutional blocks (ICBs) between REBs and RDBs as shown in Fig. 3. We use 4 ICBs, where each ICB consists of 8 convolutional layers which has the same architecture as the REB except the pooling layer.

After RDBs, an additional ICB with 8 layers and a final convolutional layer with J output channels are applied. For example, for a stereo separation task where only the magnitude of masks $|\hat{M}|$ is used as a baseline, J is set to 2. Similarly, if the decoupling of magnitude and phase are predicted (as in Section 3.1), J is set to 6 (two channels of $|\hat{M}|$, \hat{P}_r and \hat{P}_i). In our complete system in Section 3.2, where the combination of magnitude mask and direct magnitude prediction is used, J is set to 8 (two channels of $|\hat{M}|$, $|\hat{Q}|$, \hat{P}_r and \hat{P}_i). In total, there are 143 convolutional layers in our proposed residual UNet.

4. EXPERIMENTS

4.1 Dataset

We run an experiment to demonstrate the proposed method on the MUSDB18 dataset [29]. The MUSDB18 dataset includes separate vocals, accompaniment, bass, drums, and other instruments. Its training/validation sets contain 100/50 full tracks, respectively. The training set is further decomposed into 86 training songs and 14 songs for development and evaluation. All songs are stereo with a sampling rate of 44.1 kHz. We release the source code of our work online.¹

4.2 Data Processing

We split audio recordings into 3-second segments. Since the proposed system is convolutional layer-based UNet, it does not require previous states to calculate current predictions, making our system to be fully parallelizable. For data augmentation, we apply *mix-audio* data augmentation that is used in [33] to augment vocals, accompaniment, drums, and other instruments which randomly mix two 3-second segments from a *same* source as a new 3-second segment for training. The motivation is that, the addition of two sources also belongs to that source. We do not apply mix-audio data augmentation to bass because bass are usually monophonic in a song. Then, we create mixtures x by summing segments after mix-audio augmentation from different sources. We apply short-time Fourier transform (STFT) on x with a Hann window size of 2048 and a hop size of 441 samples, corresponding to the hop size time of 10 ms.

During training of all the proposed and baseline systems, we set batch size to 16 and apply Adam optimizer [34]. The learning rate is set to 0.001, 0.0005, 0.0001, 0.0002, and 0.0005 for vocals, accompaniment, bass, drums and other instruments. Different learning rates are used because some sources such as drums are easier to be overfitted. Those learning rates are tuned on the validation set of the MUSDB18 dataset. Learning rates are multiplied by a factor of 0.9 after every 15,000 steps. MSS systems are trained for 300,000 steps.

¹ Will be released after acceptance.

Table 2. Comparison of SDRs of previous and our proposed MSS systems.

	vocals	bass	drums	other	acc.
Open-Unmix [14]	6.32	5.23	5.73	4.02	-
Wave-U-Net [22]	3.25	3.21	4.22	2.25	-
Demucs [21]	6.29	5.83	6.08	4.12	-
Conv-TasNet [19]	6.81	5.66	6.08	4.37	-
Spleeter [16]	6.86	5.51	6.71	4.55	-
D3Net [35]	7.24	5.25	7.01	4.53	13.52
ResUNetDecouple+	8.98	6.04	6.62	5.29	16.63

Table 3. SDRs of the proposed systems (2nd – 7th rows) in a comparison to the previous system, UNetPhase.

	vocals	bass	drums	other	acc.
UNetPhase [25]	7.45	5.42	6.51	4.86	15.23
UNet	7.20	4.79	5.94	4.49	14.69
UNetDecouple	7.65	5.00	6.29	4.71	15.21
UNetDecouple+	7.81	5.28	6.47	5.00	15.32
ResUNet	7.79	5.00	6.20	5.13	16.15
ResUNetDecouple	8.72	5.71	6.50	5.20	16.39
ResUNetDecouple+	8.98	6.04	6.62	5.29	16.63

4.3 Result 1: Comparison with Previous Methods

We compare our proposed system with several systems including previous time domain and frequency domain based systems. Signal-to-Distortion Ratio (SDR) [28] is used as evaluation metric. The *museval* toolbox [36] is used to calculate MSS metrics.

Table 2 shows the SDRs of previous MSS systems as well as those of our best performing system. The first row shows the performance of Open-Unmix [14], which consists of three bi-directional long short-term memory layers achieves a vocals SDR of 6.32 dB. The second row shows that the Wave-U-Net [22] system trained in the time-domain achieve slightly lower SDRs than other time-frequency domain systems. The third to the eighth rows show the results of Demucs [21], Conv-TasNet [19], Spleeter [16], and D3Net [35]. Among the compared methods, D3Net achieves the best vocals and drums SDRs of 7.24 dB and 7.01 dB respectively. The Demucs achieves the best bass SDR of 5.83 dB, and the Spleeter achieves the best other SDR of 4.55 dB in previous works. As in the last row of Table 2, our proposed residual UNet with the decoupling and the combination of magnitude masks and direct prediction significantly outperforms previous methods in separating vocals, bass, other, and accompaniments.

4.4 Result 2: Ablation Study

In this section, we show the performances of our proposed systems that partially incorporate our modification. We also compare them with the system from [25], which we call UNetPhase. We implement a UNetPhase with 33 layers.

In Table 3, UNet, UNetDecouple, and UNetDecouple+ are variants of a 33-layer UNet and ResUNet, ResUNet-

Decouple, ResUNetDecouple+ are variants of a 143-layer residual UNet. UNet and ResUNet are models with magnitude masks only, i.e., phase is not considered in the model. ‘Decouple’ indicates that the proposed decoupling of magnitude and phase is applied. ‘+’ indicates the further improvement of combining the magnitude masks and direct prediction as introduced in Section 3.2.

First, UNet, which only predicts the magnitude of masks, performed slightly worse than UNetPhase. Here, we observe the average improvement by predicting phase is 0.57 dB.

Second, we can compare the trend within the row 2-4 or the row 5-7. Both for UNet’s and ResUNet’s, decoupling of the magnitude and phase improves the performance – by 0.35 dB with UNet and 0.45 dB with ResUNet on average. The ‘+’ models shows further average improvements of 0.2 dB and 0.196 dB with UNet and ResUNet, respectively. This result indicates that combining bounded mask estimation and direct magnitude prediction can improve MSS.

Third, when the other conditions are fixed, ResUNet always outperforms UNet for all source instruments. It clearly demonstrates the effectiveness of a very deep architecture in MSS. The average improvement of ResUNet from UNet is 0.7 dB.

The results did not show a clear sign that the upper bound that we discussed in Section 2 is playing a critical role in the current systems. For example, for vocal/bass/drums/other/accompaniments, the upper bounds of cIRM (1), i.e., UNetPhase, are 19.84/17.99/19.10/18.97/26.54 dB, all of which are more than 10 dB higher than the performance of UNetPhase. Compared to UNetPhase, UNetDecouple+, which is a case of cIRM (inf), only slightly outperforms UNetPhase by 0.082 dB on average and did not perform better on bass and drums.

5. CONCLUSION

In this paper, we investigated the music source separation (MSS) task. We showed that previous MSS methods have upper bound of the performance due to a strong assumption on the magnitude of the masks. We also showed that accurate phase estimation and unbound complex ideal ratio masks (cIRMs) are important for MSS. Finally, we analyzed the distribution of cIRMs for MSS and showed that 22% of cIRMs have magnitude larger than one. To overcome the limits, We proposed to decouple the estimation of magnitudes and phases. We also proposed to combine bounded magnitude masks and direct prediction methods for more flexible magnitude estimation. Finally, we proposed a very deep MSS architecture, a residual UNet with 143 layers. In the experiment, we showed that our proposed modifications improve the performance, achieving an SDR of 8.98 dB for vocals in MUSDB18. In the future work, we will explore a more effective approach to design a MSS that solve the issues we analyzed better, especially, the issue of the bounded magnitude masks.

6. REFERENCES

- [1] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, “Melody extraction from polyphonic music signals: Approaches, applications, and challenges,” *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.
- [2] J.-L. Durrieu, B. David, and G. Richard, “A musically motivated mid-level representation for pitch estimation and musical audio source separation,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1180–1191, 2011.
- [3] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, “Automatic music transcription: An overview,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.
- [4] J. Pons, J. Janer, T. Rode, and W. Nogueira, “Remixing music using source separation algorithms to improve the musical experience of cochlear implant users,” *The Journal of the Acoustical Society of America*, vol. 140, no. 6, pp. 4338–4349, 2016.
- [5] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
- [6] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, “Sparse representations in audio and music: from coding to source separation,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, 2009.
- [7] M. E. Davies and C. J. James, “Source separation using single channel ICA,” *Signal Processing*, vol. 87, no. 8, pp. 1819–1832, 2007.
- [8] Z. Duan, Y. Zhang, C. Zhang, and Z. Shi, “Unsupervised single-channel music source separation by average harmonic structure modeling,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 4, pp. 766–778, 2008.
- [9] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, “A regression approach to speech enhancement based on deep neural networks,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, 2014.
- [10] G. Naithani, T. Barker, G. Parascandolo, L. Bramsl, N. H. Pontoppidan, T. Virtanen *et al.*, “Low latency sound source separation using convolutional recurrent neural networks,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 71–75.
- [11] S. Uhlich, M. Porcu, F. Giron, M. Enekl, T. Kemp, N. Takahashi, and Y. Mitsufuji, “Improving music source separation based on deep neural networks through data augmentation and network blending,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 261–265.
- [12] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, “Singing voice separation with deep u-net convolutional networks,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [13] P. Chandna, M. Miron, J. Janer, and E. Gómez, “Monoaural audio source separation using deep convolutional neural networks,” in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017, pp. 258–266.
- [14] F. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, “Open-unmix-a reference implementation for music source separation,” *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, 2019.
- [15] N. Takahashi, N. Goswami, and Y. Mitsufuji, “Mmdenselm: An efficient combination of convolutional and recurrent neural networks for audio source separation,” in *IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 106–110.
- [16] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.
- [17] H. Liu, L. Xie, J. Wu, and G. Yang, “Channel-wise subband input for better voice and accompaniment separation on high resolution music,” in *INTERSPEECH*, 2020.
- [18] Y. Hu, Y. Liu, S. Lv, M. Xing, S. Zhang, Y. Fu, J. Wu, B. Zhang, and L. Xie, “DCCRN: Deep complex convolution recurrent network for phase-aware speech enhancement,” in *INTERSPEECH*, 2020.
- [19] Y. Luo and N. Mesgarani, “Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation,” *IEEE/ACM transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.
- [20] F. Lluís, J. Pons, and X. Serra, “End-to-end music source separation: is it possible in the waveform domain?” *arXiv preprint arXiv:1810.12187*, 2018.
- [21] A. Défossez, N. Usunier, L. Bottou, and F. Bach, “Music source separation in the waveform domain,” *arXiv preprint arXiv:1911.13254*, 2019.
- [22] D. Stoller, S. Ewert, and S. Dixon, “Wave-u-net: A multi-scale neural network for end-to-end audio source separation,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [23] N. Takahashi, P. Agrawal, N. Goswami, and Y. Mitsufuji, “PhaseNet: Discretized Phase Modeling with Deep Neural Networks for Audio Source Separation,” in *INTERSPEECH*, 2018, pp. 2713–2717.

- [24] D. Yin, C. Luo, Z. Xiong, and W. Zeng, “Phasen: A phase-and-harmonics-aware speech enhancement network,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 9458–9465.
- [25] H. Choi, J.-H. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, “Phase-aware speech enhancement with deep complex u-net,” in *International Conference on Learning Representations (ICLR)*, 2019.
- [26] D. Wang and J. Chen, “Supervised speech separation based on deep learning: An overview,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, 2018.
- [27] K. Tan and D. Wang, “Complex spectral mapping with a convolutional recurrent network for monaural speech enhancement,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6865–6869.
- [28] E. Vincent, R. Gribonval, and C. Févotte, “Performance measurement in blind audio source separation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.
- [29] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, “MUSDB18 - a corpus for music separation,” 2017.
- [30] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.
- [31] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *International Conference on Machine Learning (ICML)*, vol. 30, no. 1, 2013.
- [32] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European conference on computer vision (ECCV)*, 2016, pp. 630–645.
- [33] X. Song, Q. Kong, X. Du, and Y. Wang, “CatNet: music source separation system with mix-audio augmentation,” *arXiv preprint arXiv:2102.09966*, 2021.
- [34] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.
- [35] N. Takahashi and Y. Mitsufuji, “D3Net: Densely connected multidilated DenseNet for music source separation,” *arXiv preprint arXiv:2010.01733*, 2020.
- [36] F. Stöter, A. Liutkus, and N. Ito, “The 2018 signal separation evaluation campaign,” in *International Conference on Latent Variable Analysis and Signal Separation*, 2018, pp. 293–305.

ARTIST SIMILARITY WITH GRAPH NEURAL NETWORKS

Filip Korzeniowski Sergio Oramas Fabien Gouyon

Pandora Media LLC., Oakland, California, USA

fkorzeniowski@pandora.com

ABSTRACT

Artist similarity plays an important role in organizing, understanding, and subsequently, facilitating discovery in large collections of music. In this paper, we present a hybrid approach to computing similarity between artists using graph neural networks trained with triplet loss. The novelty of using a graph neural network architecture is to combine the topology of a graph of artist connections with content features to embed artists into a vector space that encodes similarity. To evaluate the proposed method, we compile the new OLGA dataset, which contains artist similarities from AllMusic, together with content features from AcousticBrainz. With 17,673 artists, this is the largest academic artist similarity dataset that includes content-based features to date. Moreover, we also showcase the scalability of our approach by experimenting with a much larger proprietary dataset. Results show the superiority of the proposed approach over current state-of-the-art methods for music similarity. Finally, we hope that the OLGA dataset will facilitate research on data-driven models for artist similarity.

1. INTRODUCTION

Music similarity has sparked interest early in the Music Information Retrieval community [1, 2], and has since then become a central concept for music discovery and recommendation in commercial music streaming services.

There is however no consensual notion of *ground-truth* for music similarity, as several viewpoints are relevant [2]. For instance, music similarity can be considered at several levels of granularity; musical items of interest can be musical phrases, tracks, artists, genres, to name a few. Furthermore, the perception of similarity between two musical items can focus either on (1) comparing *descriptive* (or *content-based*) aspects, such as the melody, harmony, timbre (in acoustic or symbolic form), or (2) *relational* (sometimes called *cultural*) aspects, such as listening patterns in user-item data, frequent co-occurrences of items in playlists, web pages, et cetera.

In this paper, we focus on *artist-level* similarity, and formulate the problem as a *retrieval* task: given an artist, we want to retrieve the most similar artists, where the ground-truth for similarity is *cultural*. More specifically, artist similarity is defined by music experts in some experiments, and by the “wisdom of the crowd” in other experiments.

A variety of methods have been devised for computing artist similarity, from the use of audio descriptors to measure similarity [3], to leveraging text sources by measuring artist similarity as a document similarity task [4]. A significant effort has been dedicated to the study of graphs that interconnect musical entities with semantic relations as a proxy to compute artist similarity. For instance, in [5], user profiles, music descriptions and audio features are combined in a domain specific ontology to compute artist similarity, whereas in [6], semantic graphs of artists are extracted from artist biographies.

Other approaches use deep neural networks to learn artist embeddings from heterogeneous data sources and then compute similarity in the resulting embedding space [7]. More recently, metric learning approaches trained with triplet loss have been applied to learn the embedding space where similarity is computed [8–13].

In this work, we propose a novel artist similarity model that combines graph approaches and embedding approaches using graph neural networks. Our proposed model, described in details in Sec. 2, uses content-based features (audio descriptors, or musicological attributes) together with explicit similarity relations between artists made by human experts (or extracted from listener feedback). These relations are represented in a graph of artists; the topology of this graph thus reflects the contextual aspects of artist similarity.

Our graph neural network is trained using triplet loss to learn a function that embeds artists using content features and graph connections. In this embedding space, similar artists are close to each other, while dissimilar ones are further apart.

To evaluate our approach (see Sec. 4), we compile a new dataset from publicly available sources, with similarity information and audio-based features for 17,673 artists, which we describe in Sec. 3. In addition, we evaluate the scalability of our method using a larger, proprietary dataset with more than 136,731 artists.

2. MODELLING

The goal of an artist similarity model is to define a function $s(a, b)$ that estimates the similarity of two artists—i.e.,



yields a large number if artist a is considered similar to artist b , and small number if not.

Many content-based methods for similarity estimation have been developed in the last decades of MIR research. The field has closely followed the state-of-the-art in machine learning research, with general improvements coming from the latter translating well into improvements in the former. Acknowledging this fact, we select our baselines based on the most recent developments: Siamese neural networks trained with variants of the triplet loss [9–13]. Building and training this type of models falls under the umbrella of *metric learning*.

2.1 Metric Learning

The fundamental idea of metric learning is to learn a projection $\mathbf{y}_v = f(\mathbf{x}_v)$ of the input features \mathbf{x}_v of an item v into a new vector space; this vector space should be structured in a way such that the distances between points reflect the task at hand. In our case, we want similar artists to be close together in this space, and dissimilar artists far away.

There is an abundance of methods that embed items into a vector space, many rooted in statistics, that have been applied to music similarity [14]. In this paper, we use a neural network for this purpose. The idea of using neural networks to embed similar items close to each other in an embedding space was pioneered by [15], with several improvements developed in the following decades. Most notably, the contrastive learning objective—where two items are compared to each other as a training signal—was replaced by the *triplet loss* [16, 17]. Here, we observe three items simultaneously: the *anchor* item \mathbf{x}_a is compared to a *positive* sample \mathbf{x}_p and a *negative* sample \mathbf{x}_n . With the following loss formulation, the network is trained to pull the positive close to the anchor, while pushing the negative further away from it:

$$\mathcal{L}(t) = \left[d(\mathbf{y}_a, \mathbf{y}_n) - d(\mathbf{y}_a, \mathbf{y}_p) + \Delta \right]^+,$$

where t denotes the triplet $(\mathbf{y}_a, \mathbf{y}_p, \mathbf{y}_n)$, $d(\cdot)$ is a distance function (usually Euclidean or cosine), Δ is the maximum margin enforced by the loss, and $[\cdot]^+$ is the ramp function.

As mentioned before, state-of-the-art music similarity models are almost exclusively based on learning deep neural networks using the triplet loss. We thus adopt this method as our baseline model, which will serve as a comparison point to the graph neural network we propose in the following sections.

2.2 Graph Neural Networks

A set of artists and their known similarity relations can be seen as a graph, where the artists represent the nodes, and the similarity relations their (undirected) connections. Graph methods thus naturally lend themselves to model the artist similarity problem [6]. A particular set of graph-based models that has been gaining traction recently are *graph neural networks* (GNNs), specifically *convolutional GNNs*. Pioneered by [18], convolutional GNNs have become increasingly popular for modelling different tasks

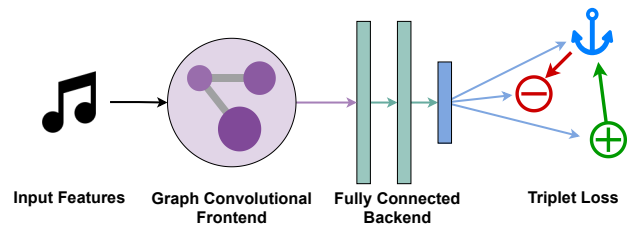


Figure 1: Overview of the graph neural network we use in this paper. First, the input features \mathbf{x}_v are first passed through a *front-end* of graph convolution layers (see Sec. 2.2.2 for details); then, the output of the front-end is passed through a traditional deep neural network *back-end* to compute the final embeddings \mathbf{y}_v of artist nodes. Based on these embeddings, we use the triplet loss to train the network to project similar artists (positive, green) closer to the anchor, and dissimilar ones (negative, red) further away.

that can be interpreted as graphs. We refer the interested reader to [19] for a comprehensive and historical overview of GNNs. For brevity, we will focus on the one specific model our work is based on—the GraphSAGE model introduced by [20] and refined by [21]—and use the term GNNs for convolutional GNNs.

2.2.1 Model Overview

The GNN we use in this paper comprises two parts: first, a block of *graph convolutions* (GC) processes each node’s features and combines them with the features of adjacent nodes; then, another block of fully connected layers project the resulting feature representation into the target embedding space. See Fig. 1 for an overview.

We train the model using the triplet loss, in an identical setup as the baseline model. Viewing the proposed GNN from this angle, the only difference of the GNN from a standard embedding network is the additional *Graph Convolutional Frontend*. In other words, if we remove all graph convolutional layers, we arrive at our baseline model, a fully connected Deep Neural Network (DNN).

2.2.2 Graph Convolutions

The graph convolution algorithm, as defined in [20, 21], features two operations which are not found in classic neural networks: a *neighborhood function* $\mathcal{N}(\cdot)$, which yields the set of neighbors of a given node; and an *aggregation function*, which computes a vector-valued aggregation of a set of input vectors.

As a neighborhood function, most models use guided or uniform sub-sampling of the graph structure [20–22]. This limits the number of neighbors to be processed for each node, and is often necessary to adhere to computational limits. As aggregation functions, models commonly apply pooling operators, LSTM networks, or (weighted) point-wise averages [20].

In this work, we take a simple approach, and use point-wise weighted averaging to aggregate neighbor representations, and select the strongest 25 connections as neighbors.

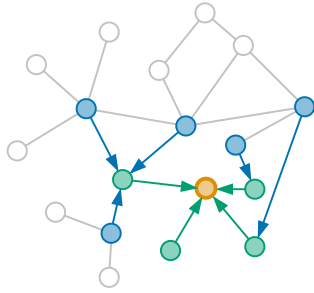


Figure 2: Tracing the graph to find the necessary input nodes for embedding the target node (orange). Each graph convolution layer requires tracing one step in the graph. Here, we show the trace for a stack of two such layers. To compute the embedding of the target node in the last layer, we need the representations from the previous layer of itself and its neighbors (green). In turn, to compute these representations, we need to expand the neighborhood by one additional step in the preceding GC layer (blue). Thus, the features of all colored nodes must be fed to the first graph convolution layer.

If weights are not available, we use the simple average of random 25 connections. This enables us to use a single sparse dot-product with an adjacency matrix to select and aggregate neighborhood embeddings. Note that this is not the full adjacency matrix of the complete graph, as we select only the parts of the graph which are necessary for computing embeddings for the nodes in a mini-batch.

Algorithm 1 describes the inner workings of the graph convolution block of our model. Here, the matrix $\mathbf{X} \in \mathbb{R}^{D \times V}$ stores the D -dimensional features of all V nodes, the symmetric sparse matrix $\mathbf{A} \in \mathbb{R}^{V \times V}$ defines the connectivity of the graph, and $\mathcal{N}(v)$ is a neighborhood function which returns all connected nodes of a given node v (here, all non-zero elements in the v^{th} row of \mathbf{A}).

To compute the output of a graph convolution layer for a node, we need to know its neighbors. Therefore, to compute the embeddings for a mini-batch of nodes \mathcal{V} , we need to know which nodes are in their joint neighborhood. Thus, before the actual processing, we first need to trace the graph to find the node features necessary to compute the embeddings of the nodes in the mini-batch. This is shown in Fig. 2, and formalized in lines 1–4 of Alg. 1.

At the core of each graph convolution layer $k \in [1 \dots K]$ there are two non-linear projections, parameterized by projection matrices $\mathbf{Q}_k \in \mathbb{R}^{H_{Q_k} \times D}$ and $\mathbf{W}_k \in \mathbb{R}^{H_{W_k} \times (H_{Q_k} + D)}$, and a point-wise non-linear activation function σ , in our case, the Exponential Linear Unit function (ELU). Here, H_{Q_k} and H_{W_k} are the output dimensions of the respective projections. The last output, $\mathbf{X}_K \in \mathbb{R}^{H_{W_K} \times V}$, holds the l_2 -normalized representations of each node in the mini-batch in its columns. It is fed into the following fully connected layers, which then compute the output embedding \mathbf{y}_v of a node. Finally, these embeddings are used to compute the triplet loss and back-propagate it through the GNN.

Algorithm 1: GRAPH CONVOLUTION BLOCK

Input : Node input features \mathbf{X} .
 Sparse connectivity matrix \mathbf{A} .
 Nodes in mini-batch $\mathcal{V} \subset [1 \dots V]$.

Output: Node output representation \mathbf{X}_K

▷ Trace back input nodes for each layer.

```

1  $\mathcal{V}_K \leftarrow \mathcal{V}$ ;
2 for  $k = K - 1 \dots 0$  do
3   |  $\mathcal{V}_k \leftarrow \bigcup_{v \in \mathcal{V}_{k+1}} \mathcal{N}(v)$ ;
4 end
   ▷ Select input features for first layer. We use  $\mathbf{M}[r, c]$  to
   denote selecting  $r$  rows and  $c$  columns from a matrix
    $\mathbf{M}$ .
5  $\mathbf{X}_0 = \mathbf{X}[:, \mathcal{V}_0]$ ;
6 for  $k = 1 \dots K$  do
7   |  $\mathbf{A}_k = \mathbf{A}[\mathcal{V}_{k-1}, \mathcal{V}_k]$ ;
8   |  $\mathbf{N}_k = \sigma(\mathbf{Q}_k \cdot \mathbf{X}_{k-1}) \cdot \mathbf{A}_k$ ;
9   |  $\mathbf{X}_k \leftarrow \sigma\left(\mathbf{W}_k \cdot \begin{bmatrix} \mathbf{N}_k \\ \mathbf{X}_{k-1}[:, \mathcal{V}_k] \end{bmatrix}\right)$ ;
   ▷  $l_2$ -normalize embeddings of each output node.
10  |  $\mathbf{X}_k \leftarrow \left[ \frac{\mathbf{x}_v}{\|\mathbf{x}_v\|_2} \mid v \in \mathcal{V}_k \right]$ ;
11 end
12 return  $\mathbf{X}_K$ 
    
```

3. DATASETS

Many published studies on the topic of artist similarity are limited by data: datasets including artists, their similarity relations, and their features comprise at most hundreds to a few thousand artists. In addition, the quality of the ground truth provided is often based on 3rd party APIs with obscure similarity methods like the last.fm API, rather than based on data curated by human experts.

For instance, in [6], two datasets are provided, one with ~2k artists and similarity based on last.fm relations, and another with only 268 artists, but based on relations curated by human experts. In [4], a dataset of 1,677 artists based on last.fm similarity relations is used for evaluation. Also, the dataset used in the Audio Music Similarity and Retrieval (AMS) MIREX task, which was manually curated, contains data about only 602 artists. Other works, like [8], use tag data shared among tracks or artists as a proxy for similarity estimation—which can be considered as a weak signal of similarity—and use a small set of 879 human-labeled triplets for evaluation.

For all these issues regarding existing datasets, we compiled a new dataset, the OLGA Dataset, which we describe in the following.

3.1 The OLGA Dataset

For the OLGA (“Oh, what a Large Graph of Artists”) dataset, we bring together content-based low-level features from AcousticBrainz [23], and similarity relations from AllMusic. Assembling the data works as follows:

1. Select a common pool of artists based on the unique artists in the Million Song Dataset [24].

2. Map the available MusicBrainz IDs of the artists to AllMusic IDs using mapping available from MusicBrainz.
3. For each artist, obtain the list of “related” artists from AllMusic; this data can be licensed and accessed on their website. Use only related artists which can be mapped back to MusicBrainz.
4. Using MusicBrainz, select up to 25 tracks for each artist using their API, and collect the low-level features of the tracks from AcousticBrainz.
5. Compute the track feature centroid of each artist.

In total, the dataset comprises 17,673 artists connected by 101,029 similarity relations. On average, each artist is connected to 11.43 other artists. The quartiles are at 3, 7, and 16 connections per artist. The lower 10% of artists have only one connection, the top 10% have at least 27.

While the dataset size is still small compared to industrial catalog sizes, it is significantly bigger than other datasets available for this task. Its size and available features will allow us to apply more data-driven machine learning methods to the problem of artist similarity.¹

For our experiments, we partition the artists following an 80/10/10 split into 14,139 training, 1767 validation, and 1767 test artists.

3.2 Proprietary Dataset

We also use a larger proprietary dataset to demonstrate the scalability of our approach. Here, explicit feedback from listeners of a music streaming service is used to define whether two artists are similar or not.

For artist features, we use the centroid of an artist’s track features. These track features are *musicological* attributes annotated by experts, and comprise hundreds of content-based characteristics such as “amount of electric guitar”, or “prevalence of groove”.

In total, this dataset consists of 136,731 artists connected by 3,277,677 similarity relations. The number of connections per artists is a top-heavy distribution with few artists sharing most of the connections: the top 10% are each connected to more than 134 others, while the bottom 10% to only one. The quartiles are at 2, 5, and 48 connections per artist.

We follow the same partition strategy as for the OLGA dataset, which results in 109,383 training, 13,674 validation, and 13,674 test artists.

4. EXPERIMENTS

Our experiments aim to evaluate how well the embeddings produced by our model capture artist similarity. To this end, we set up a ranking scenario: given an artist, we collect its K nearest neighbors sorted by ascending distance, and evaluate the quality of this ranking. To quantify this, we use normalized discounted cumulative gain [25] with

a high cut-off at $K = 200$ (“ndcg@200”). We prefer this metric over others, because it was shown that at high cut-off values, it provides better discriminative power, as well as robustness to sparsity bias (and, to moderate degree, popularity bias) [26]. Formally, given an artist a with an ideal list of similar artists \mathbf{s} (sorted by relevance), the $nDCG_K$ of a predicted list of similar artists $\hat{\mathbf{s}}$ is defined as:

$$nDCG_K(a, \hat{\mathbf{s}}, \mathbf{s}) = \frac{\sum_{k=1}^K g(\hat{s}_k, a) d(k)}{\sum_{k=1}^K g(s_k, a) d(k)},$$

where $g(\cdot, a)$, the *gain*, is 1 if an artist is indeed similar to a , and 0 otherwise, and $d(k) = \log_2^{-1}(k + 1)$ the *discounting* factor, weights top rankings higher than the tail of the list.

In the following, we first explain the models, their training details, the features, and the evaluation data used in our experiments. Then, we show, compare and analyze the results.

4.1 Models

As explained in Sec. 2.2.1, a GNN with no graph convolutional layers is identical to our baseline model (i.e. a DNN trained using triplet loss). This allows us to fixate hyper-parameters between baseline and the proposed GNN, and isolate the effect of adding graph convolutions to the model. For each dataset, we thus train and evaluate four models with 0 to 3 graph convolutional layers.

The other hyper-parameters remain fixed: each layers in the graph convolutional front-end consists of 256 ELUs [27]; the back-end comprises two layers of 256 ELUs each, and one linear output layer with a 100 dimensions; we train the networks using the ADAM optimizer [28] with a linear learning-rate warm-up [29] for the first epoch, and following a cosine learning rate decay [30] for the remaining 49 epochs (in contrast to [30], we do not use warm-restarts); for selecting triplets, we apply distance-weighted sampling [31], and use a margin of $\Delta = 0.2$ in the loss; finally, as distance measure, we use Euclidean distance between l_2 -normalized embeddings.

We are able to train the largest model with 3 graph convolutional layers within 2 hours on the proprietary dataset, and under 5 minutes on OLGA, using a Tesla P100 GPU and 8 CPU threads for data loading.

4.2 Features

We build artist-level features by averaging track-level features of the artist’s tracks. Depending on the dataset, we have different types of features at hand.

In the OLGA dataset, we have low-level audio features as extracted by the Essentia library.² These features represent track-level statistics about the loudness, dynamics and spectral shape of the signal, but they also include more abstract descriptors of rhythm and tonal information, such as bpm and the average pitch class profile. We select all numeric features and pre-process them as follows: we apply

¹ The procedure to assemble the dataset, including relevant metadata, is available on <https://gitlab.com/fdlm/olga/>.

² See https://essentia.upf.edu/streaming_extractor_music.html#music-descriptors

element-wise standardization, discard features with missing values, and flatten all numbers into a single vector of 2613 elements.

In the proprietary dataset, we use numeric musicological descriptors annotated by experts (for example, “the nasality of the singing voice”). We apply the same pre-processing for these, resulting in a total of 170 values.

Using two different types of content features gives us the opportunity to evaluate the utility of our graph model under different circumstances, or more precisely, features of different quality and signal-to-noise ratio. The low-level audio-based features available in the OLGA dataset are undoubtedly noisier and less specific than the high-level musical descriptors manually annotated by experts, which are available in the proprietary dataset. Experimenting with both permits us to gauge the effect of using the graph topology for different data representations.

In addition, we also train models with *random vectors* as features. For each artist, we uniformly sample a random vector of the same dimension as the real features, and keep it constant throughout training and testing. This way, we can differentiate between the performance of the real features and the performance of using the graph topology in the model: the results of a model with no graph convolutions is only due to the features, while the results of a model with graph convolutions but random features is only due to the usage of the graph topology.

4.3 Evaluation Data

As described in Section 3, we partition artists into a training, validation and test set. When evaluating on the validation or test sets, we only consider artists from these sets as candidates and potential true positives. Specifically, let \mathcal{V}_{eval} be the set of evaluation artists, we only compute embeddings for those, and retrieve nearest neighbors from this set, and only consider ground truth similarity connections within \mathcal{V}_{eval} .

This notion is more nuanced in the case of GNNs. Here, we want to exploit the *known artist graph topology* (i.e., which artists are connected to each other) when computing the embeddings. To this end, we use all connections between artists in \mathcal{V}_{train} (the training set) and connections between artists in \mathcal{V}_{train} and \mathcal{V}_{eval} . This process is outlined in Fig. 3.

Note that this does not leak information between train and evaluation sets; the features of evaluation artists have not been seen during training, and connections within the evaluation set—these are the ones we want to predict—remain hidden.

4.4 Results

Table 1 compares the baseline model with the proposed GNN. We can see that the GNN easily out-performs the DNN. It achieves an NDCG@200 of 0.55 vs. 0.24 on the OLGA dataset, and 0.57 vs. 0.44 on the proprietary dataset. The table also demonstrates that the graph topology is more predictive of artist similarity than content-based features: the GNN, using random features, achieves

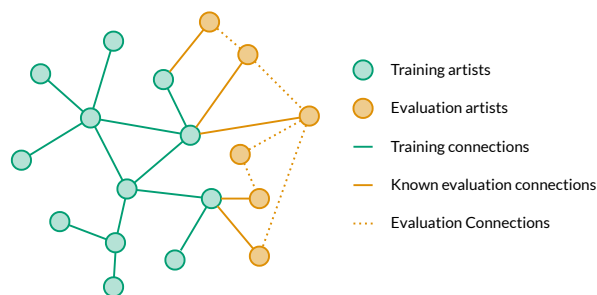


Figure 3: Artist nodes and their connections used for training (green) and evaluation (orange). During training, only green nodes and connections are used. When evaluating, we extend the graph with the orange nodes, but only add connections between validation and training artists. Connections among evaluation artists (dotted orange) remain hidden. We then compute the embeddings of all evaluation artists, and evaluate based on the hidden evaluation connections.

Dataset	Features	DNN	GNN
OLGA	Random	0.02	0.45
	AcousticBrainz	0.24	0.55
Proprietary	Random	0.00	0.52
	Musicological	0.44	0.57

Table 1: NDCG@200 for the baseline (DNN) and the proposed model with 3 graph convolution layers (GNN), using features or random vectors as input. The GNN with real features as input gives the best results. Most strikingly, the GNN with random features—using only the known graph topology—out-performs the baseline DNN with informative features.

better results than a DNN using informative features for both datasets (0.45 vs. 0.24 on OLGA, and 0.52 vs 0.44 on the proprietary dataset).

Additionally, the results indicate—perhaps to little surprise—that low-level audio features in the OLGA dataset are less informative than manually annotated high-level features in the proprietary dataset. Although the proprietary dataset poses a more difficult challenge due to the much larger number of candidates (14k vs. 1.8k), the DNN—which can only use the features—improves more over the random baseline in the proprietary dataset (+0.44), compared to the improvement (+0.22) on OLGA. These are only indications; for a definitive analysis, we would need to use the exact same features in both datasets.

Similarly, we could argue that the topology in the proprietary dataset seems more coherent than in the OLGA dataset. We can judge this by observing the performance gain obtained by a GNN with random feature—which can only leverage the graph topology to find similar artists—compared to a completely random baseline (random features without GC layers). In the proprietary dataset, this performance gain is +0.52, while in the OLGA dataset,

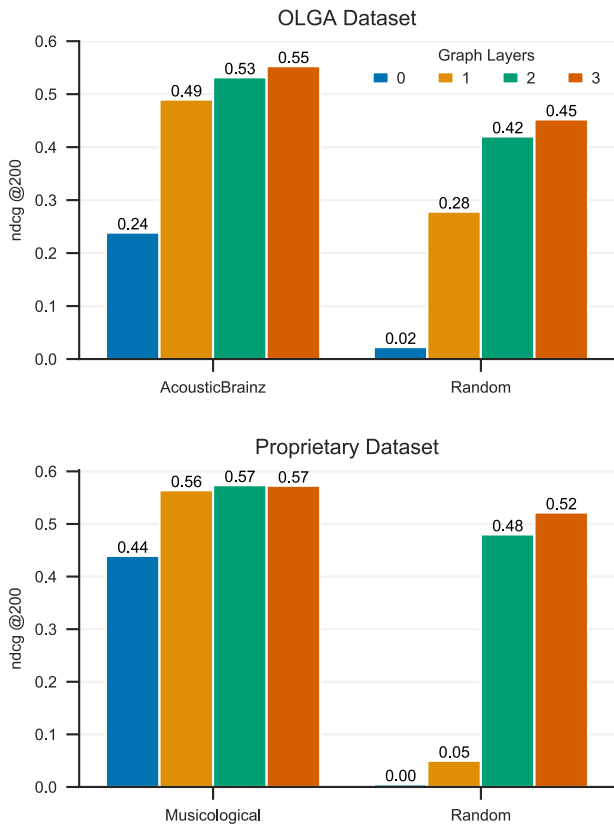


Figure 4: Results on the OLGA (top) and the proprietary dataset (bottom) with different numbers of graph convolution layers, using either the given features (left) or random vectors as features (right).

only +0.43. Again, while this is not a definitive analysis (other factors may play a role), it indicates that the large amounts of user feedback used to generate ground truth in the proprietary dataset give stable and high-quality similarity connections.

Figure 4 depicts the results for each model and feature set depending on the number of graph convolutional layers used. (Recall that a GNN with 0 graph convolutions corresponds to the baseline DNN.) In the OLGA dataset, we see the scores increase with every added layer. This effect is less pronounced in the proprietary dataset, where adding graph convolutions does help significantly, but results plateau after the first graph convolutional layer. We believe this is due to the quality and informativeness of the features: the low-level features in the OLGA dataset provide less information about artist similarity than high-level expertly annotated musicological attributes in the proprietary dataset. Therefore, exploiting contextual information through graph convolutions results in more uplift in the OLGA dataset than in the proprietary one.

Looking at the scores obtained using random features (where the model depends solely on exploiting the graph topology), we observe two remarkable results. First, whereas one graph convolutional layer suffices to outperform the feature-based baseline in the OLGA dataset (0.28 vs. 0.24), using only one GC layer does not produce meaningful results (0.05) in the proprietary dataset. We

believe this is due to the different sizes of the respective test sets: 14k in the proprietary dataset, while only 1.8k in OLGA. Using only a very local context seems to be enough to meaningfully organize the artists in a smaller dataset.

Second, most performance gains are obtained with two GC layers, while adding the third GC layer pushes the results to a much lesser degree. Our explanation for this effect is that most similar artists are connected through at least one other, common artist. In other words, most artists form similarity cliques with at least two other artists. Within these cliques, in which every artist is connected to all others, missing connections are easily retrieved by no more than 2 graph convolutions.

In fact, in the OLGA dataset, ~71% of all cliques fulfill this requirement. This means that, for any hidden similarity link in the data, in 71% of cases, the true similar artist is within 2 steps in the graph—which corresponds to using two GC layers.

5. SUMMARY AND FUTURE WORK

In this paper, we described a hybrid approach to computing artist similarity, which uses graph neural networks to combine content-based features with explicit relations between artists. To evaluate our approach, we assembled a novel academic dataset with 17,673 artists, their features, and their similarity relations. Additionally, we used a much larger proprietary dataset to show the scalability of our method. The results showed that leveraging known similarity relations between artists can be more effective for understanding their similarity than high-quality features, and that combining both gives the best results.

Our work is a first step towards models that directly use known relations between musical entities—like tracks, artists, or even genres—or even across these modalities. Multi-modal connections could also help predicting artist similarity; we could add collaborations, or band membership connections to the graph. Finally, it would be interesting to analyze the effect of our approach on long-tail recommendations and/or the cold-start problem.

6. REFERENCES

- [1] J.-J. Aucouturier and F. Pachet, “Music Similarity Measures: What’s The Use?” in *Proc. of the 3rd International Conference on Music Information Retrieval (ISMIR)*, Paris, France, Oct. 2002.
- [2] D. P. W. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence, “The Quest for Ground Truth in Musical Artist Similarity,” in *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, Paris, France, Oct. 2002.
- [3] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, “On Rhythm and General Music Similarity,” in *Proc. of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, Oct. 2009.

- [4] M. Schedl, D. Hauger, and J. Urbano, "Harvesting microblogs for contextual music similarity estimation: A co-occurrence-based framework," *Multimedia Systems*, vol. 20, no. 6, pp. 693–705, Nov. 2014.
- [5] Ò. Celma and X. Serra, "FOAFing the music: Bridging the semantic gap in music recommendation," *Journal of Web Semantics*, vol. 6, no. 4, pp. 250–256, Nov. 2008.
- [6] S. Oramas, M. Sordo, L. Espinosa-Anke, and X. Serra, "A Semantic-Based Approach for Artist Similarity," in *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, Oct. 2015.
- [7] B. McFee and G. R. G. Lanckriet, "Heterogeneous Embedding For Subjective Artist Similarity," in *Proc. of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, Kobe, Japan, Oct. 2009.
- [8] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled Multidimensional Metric Learning for Music Similarity," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.
- [9] G. Doras, F. Yesiler, J. Serrà, E. Gómez, and G. Peeters, "Combining Musical Features for Cover Detection," in *Proc. of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, Oct. 2020.
- [10] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Metric Learning vs Classification for Disentangled Music Representation Learning," in *Proc. of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, Aug. 2020.
- [11] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation Learning of Music Using Artist Labels," in *Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, Jun. 2018.
- [12] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and Scalable Version Identification Using Musically-Motivated Embeddings," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.
- [13] M. Dorfer, A. Arzt, and G. Widmer, "Learning Audio-Sheet Music Correspondences for Score Identification and Offline Alignment," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, Jul. 2017.
- [14] M. Slaney, K. Q. Weinberger, and W. White, "Learning a Metric for Music Similarity," in *Proc. of the 9th International Conference on Music Information Retrieval (ISMIR)*, Philadelphia, USA, Sep. 2008.
- [15] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "Siamese" time delay neural network," in *Proc. of the 6th International Conference on Neural Information Processing Systems (NIPS)*, San Francisco, USA, Nov. 1993.
- [16] E. Hoffer and N. Ailon, "Deep Metric Learning Using Triplet Network," in *Similarity-Based Pattern Recognition (SIMBAD)*, A. Feragen, M. Pelillo, and M. Loog, Eds., Copenhagen, Denmark, Oct. 2015.
- [17] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning Fine-Grained Image Similarity with Deep Ranking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, USA, Jun. 2014.
- [18] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral Networks and Locally Connected Networks on Graphs," in *Proc. of the International Conference on Learning Representations (ICLR)*, Banff, Canada, Apr. 2014.
- [19] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
- [20] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, Long Beach, USA, Dec. 2017.
- [21] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *Proc. of the 24th International Conference on Knowledge Discovery & Data Mining (SIGKDD)*, Jul. 2018.
- [22] J. Oh, K. Cho, and J. Bruna, "Advancing GraphSAGE with A Data-Driven Node Sampling," in *Proc. of the ICLR Workshop on Representation Learning on Graphs and Manifolds*, New Orleans, USA, May 2019.
- [23] A. Porter, D. Bogdanov, R. Kaye, R. Tsukanov, and X. Serrà, "AcousticBrainz: A Community Platform for Gathering Music Information Obtained from Audio," in *Proc. of the 16th Conference of the International Society for Music Information Retrieval (ISMIR)*, Málaga, Spain, Oct. 2015.
- [24] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The Million Song Dataset," in *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, A. Klapuri and C. Leder, Eds., Miami, USA, Oct. 2011.
- [25] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, Oct. 2002.

- [26] D. Valcarce, A. Bellogín, J. Parapar, and P. Castells, “On the robustness and discriminative power of information retrieval metrics for top-N recommendation,” in *Proc. of the 12th ACM Conference on Recommender Systems (RECSYS)*, Vancouver, Canada, Oct. 2018.
- [27] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs),” in *Proc. of the International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, May 2016.
- [28] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. of the International Conference on Learning Representations (ICLR)*, San Diego, USA, May 2015.
- [29] J. Ma and D. Yarats, “On the adequacy of untuned warmup for adaptive optimization,” in *Proc. of the 35th Conference on Artificial Intelligence (AAAI)*, Virtual conference, Feb. 2021.
- [30] I. Loshchilov and F. Hutter, “SGDR: Stochastic Gradient Descent with Warm Restarts,” in *Proc. of the International Conference on Learning Representations (ICLR)*, Toulon, France, May 2017.
- [31] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, “Sampling Matters in Deep Embedding Learning,” in *Proc. of the International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017.

“FINDING HOME”: UNDERSTANDING HOW MUSIC SUPPORTS LISTENERS’ MENTAL HEALTH THROUGH A CASE STUDY OF BTS

Jin Ha Lee¹, Arpita Bhattacharya², Ria Antony³, Nicole Santero⁴, Anh Le⁵

University of Washington^{1,3,5}; University of California, Irvine²; University of Nevada, Las Vegas⁴
jinhalee@uw.edu; arpitab@uci.edu; rantony@uw.edu;
santeron@unlv.nevada.edu; anhle@uw.edu

ABSTRACT

The positive impact of music on people’s mental health and wellbeing has been well researched in music psychology, but there is a dearth of research exploring the implications of these benefits for the design of commercial music services (CMS). In this paper, we investigate how popular music can support the listener’s mental health through a case study of fans of the music group BTS, with a goal of understanding how they perceive and describe the way music is influencing their mental health. We aim to derive specific design implications for CMS to facilitate such support for fans’ mental health and wellbeing. Through an online survey of 1190 responses, we identify and discuss the patterns of seven different mood regulations along with major themes on fans’ lived experiences of how BTS’s music (1) provides comfort, (2) catalyzes self-growth, and (3) facilitates coping. We conclude the study with a discussion of four specific suggestions for CMS features incorporating (1) visual elements, (2) non-music media, (3) user-generated content for collective sense-making, and (4) metadata related to mood and lyrical content that can facilitate the mental health support provided by popular music.

1. INTRODUCTION

The positive physiological and psychological effects of music on health and wellbeing are well researched in the field of music psychology [1-4]. Music can positively impact people’s mental health by supporting mood regulation [5-6], social relationships [7-8], and increasing positive emotions and self-esteem [9]. However, in the field of music information retrieval (MIR), there is limited research exploring the connections between music and the mental health of the listener. While there is substantial research on music mood in the MIR field [10], the primary focus is on identifying the mood of the music with the goal of supporting better organization, browsing, or recommendation rather than influencing the emotions of the listeners to support their health and wellbeing (e.g., [11-16]). Researchers have only recently started to investigate more deeply into the impact of music on mental health in the MIR field, especially on how such insights inform the design of music services and systems (e.g., [9]). Multiple recent MIR user studies also point out the importance of the social aspect in

listening to music [8, 17]. In particular, the complex context of being a music fan involves online communities and social media, as well as the fan’s perceived relationship with the artists [18] which will inevitably influence the impact of music on listeners’ mental health. Yet, there is a dearth of research exploring the holistic context of the listener, including external factors related to music such as the social relationship of listeners to artists and other fans.

We aim to address this gap by investigating how popular music can support music listeners’ mental health and wellbeing through a case study of the ARMY fandom. The ARMY fandom supports BTS, a seven-person music group from South Korea, and is one of the biggest music fandoms in 2021 [18]. There are several reasons for focusing on music fans as a user group in this study, and specifically the fans of BTS. First, we wanted to explore the impact of not only the music itself but also the context in which users engage with music on their mental health and wellbeing, including the context of music creation (e.g., information about the artist) and other social context (e.g., relationship with other fans). Of the possible fandoms to study, we chose the ARMY fandom due to (1) its size and diversity demonstrated by the recent census data of over 400K responses from ARMYs [19], (2) fan’s active engagement with each other demonstrated by the social media activities [20-21], and (3) prior literature indicating the music of BTS being specifically helpful in supporting their fan’s mental health as they relate to the messages in the songs [22-23].

We conducted an empirical study employing an online survey to better understand the impact of BTS’s music on their fans’ mental health. We aimed to answer the following research questions:

RQ1: How is BTS’s music helping to support fans’ mental health and well-being?

RQ2: What are the implications for designing commercial music services to facilitate such support for fans’ mental health and wellbeing?

2. RELATED WORK

2.1 Music and Mental Health

A significant body of literature in the field of music psychology examines different positive influences music can have on people’s mood, memory, learning, and identity as well as the use of music in various forms of therapy [24]. Regout conducted a literature review to understand why music has a positive influence on mental wellbeing and how music could be properly used in therapy, and showed that music can have an effect on people’s emotion,



© J. H. Lee, A. Bhattacharya, R. Antony, N., Santero, and A. Le. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. H. Lee, A. Bhattacharya, R. Antony, N., Santero, and A. Le, “Finding Home”: Understanding how music supports listeners’ mental health through a case study of BTS, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

memory, and the experience of pleasure [24]. For instance, Knight and Rickard [25] suggested that music can lessen anxiety by measuring physiological signals such as heart-beat, cortisol level, blood glucose, etc. Thayer et al. [26] also identified listening to music as one of the most successful strategies for changing a bad mood, raising energy and reduce tension. Using music in therapy helps reduce cortisol levels which can help decrease the perception of anxiety and depression [28-29]. Nilsson [27] conducted a systematic review on 42 randomized controlled trials of the effects of music interventions and found that music intervention can reduce pain and anxiety in patients. Woelfer and Lee [7] found that young people experiencing homelessness used music for mood regulation and coping as it allowed them to “calm down or relieve tension”, “help get through difficult times”, or “relieve boredom”. They also discussed how participants listened to songs for a cathartic experience and for reducing the sense of isolation.

Several studies specifically researched how emotion in music plays a role in mood regulation, “a process of satisfying personal mood-related needs by musical activities” [5]. To better understand and measure the emotional functions of music, Saarikallio and Erkkilä [5] presented a model of mood regulation which provides a useful framework for studying how this regulatory process can be applied in various contexts of music listeners. In this model, seven types of regulatory strategies are used for different changes in mood: entertainment, revival, strong sensation, diversion, discharge, mental work, and solace [5]. We used this model to inform a part of our codebook under the category of “Mood regulation”, and the mapping can be seen in the Findings section.

Some studies looked into the influence of popular music in particular. For instance, Kresovich [30] investigated how college students’ exposure to pop songs referencing mental health difficulties affect them. The study found that participants’ increased perceived personal connection with the songs and parasocial relationships with their performing artists were associated with increased mental health empathy [30]. A few researchers examined how BTS’s music influences people’s mental health. Blady [22] analyzed the discography of BTS, identifying multiple themes that support mental health by challenging stereotypes and social norms destructive to youth, encouraging self-love and empathy and showing the value of perseverance, vulnerability, and compassion. Rubin [23], in her qualitative study of 47 participants, studied how Strong Experiences with Music (SEMs) experienced by ARMYs contributed to personal growth, resulting in perspective changes and the ARMYs’ use of music for different purposes like improving mood, soothing, and motivating themselves.

2.2 Support for Mental Health in Fandoms

While some have previously believed members of fandom communities are more prone to wellbeing issues, recent research has shown no significant difference between fans and non-fans in terms of their psychological distress and wellbeing [31]. Furthermore, some research highlighted the positive impact of fandoms on mental health. For instance, fans have reported that being a part of such communities has helped them find a sense of purpose and

meaning in life [32]. Previous analyses on sports fans have shown that identification with sports teams have resulted in fans developing greater self-esteem and feeling less lonely [33] and many of these fans have noted that they identify strongly with their teams [34]. Though the positive benefits of fan comradery are widely accepted in sports, the value of fans’ social and emotional experience is often stigmatized and diminished for fans of boy bands, particularly female fans [18, 20-21].

A sense of belonging has been strongly associated with the in-group identification among different fan communities [35]. In a study examining whether this type of social connection uniquely mediates the relationship between fandom identification and wellbeing, Reysen et al. [36] noted that some fandoms provide additional aid for its members in the form of social support, advice, and resources for coping, and showed how these social connections can result in adequate wellbeing of both stigmatized and non-stigmatized fan groups. All fans have the potential to benefit from social connection as these communities, regardless of their particular interests, are similar with respect to such psychological processes [36]. Laffan [32] reports similar findings regarding K-pop fans showing increased happiness, self-esteem, and social connectedness.

With the rise of social media and fandoms growing in a digital environment, there is an opportunity for more research on how the wellbeing of fans are impacted online. There are both positive and negative aspects of social media, and research has shown that the effect of fandom activities and mental health can be different depending on the user characteristics, such as gender [37], warranting more research on different types of user groups. McInroy [38] studied how online fan communities serve as a source of social support for sexual and gender minority youth and found that participation in these communities can support the sense of connection, empowerment, and temporary escape from stress, decrease isolation, and offer opportunities for mentorship. Our study will contribute to gaining additional insights on how being part of music fandoms affects fans’ mental health.

3. STUDY DESIGN AND METHOD

We conducted an online survey asking 38 questions about fans’ engagement with BTS’s musical and non-musical content, BTS-related activities, overall experience in the fandom and their mental health. We asked six questions about how they became fans and seven demographic questions. The survey was released in early February 2021 and was open for two weeks. Participants were recruited through a prominent research account in the fandom, resulting in 10,300 engagements on Twitter. The study was approved by the institutional review board at the University of Washington. We obtained a total of 1,190 responses. The demographic information of participants is summarized in Table 1.

Age	Mean = 26, Min =13, Max = 71, StdDev = 10.02
Race	Asian = 567, White = 317, Hispanic = 210, Black = 43, Native Hawaiian/Pacific Islander = 7, American Indian/Alaska Native = 3, Described in own words = 106

Gender	Female = 1105, Non-binary = 29, Male = 15, Described in own words = 26
Country of Residence (20 or more)	USA = 268, Philippines = 155, India = 123, Indonesia = 54, Malaysia = 47, Mexico = 45, South Africa = 38, UK = 32, Germany = 29, Canada = 26

Table 1. Demographic summary of survey respondents.

The scope of the survey was broader, but in this paper we focus on reporting the findings related to the research questions stated above. This primarily includes the analysis of responses to the open-ended question asking, “Can you describe how BTS’s music is helping to support your mental health and wellbeing?” along with other demographic information about the participants.

The free-text responses were coded using a combination of inductive and deductive approaches [39]. To create the draft codebook, the research team distributed a portion of the responses to each member (approximately 200 responses per person) to review. Using a Mural board (a collaborative online tool), each team member captured and recorded emerging ideas and themes on virtual sticky notes as they reviewed their assigned responses. Afterwards, the team collectively engaged in a sorting activity to categorize the main themes which resulted in the initial codebook [40]. The codebook has three main categories of codes (1) Mood regulation, (2) Appeal, and (3) Outcome. The codes related to Mood regulation were well-aligned with the regulatory strategies from Saarikallio and Erkkilä [5], and the descriptions shown in Table 2 are adopted from their model. The rest of the thematic codes were derived inductively from the data.

Using this initial codebook, we had four coders code the open-ended survey responses. The codebook was refined through an iterative process. After coding part of the data, the team met to discuss any issues that emerged during the coding process. A series of discussions led to a few changes - Motivation was subsumed under Coping, and Social Justice and Open Discussion were merged into Difficult Topics as the coded results showed conceptual overlapping. Some definitions were rewritten for clarification. After the codebook was finalized, all four coders reviewed and revised their initial coding using the codebook.

Each survey response was coded by two independent coders, to allow for checking the intercoder reliability. Using a formula in a spreadsheet, all the discrepancies in the coded results were highlighted. We followed a consensus model [41] where two coders reviewed and discussed all the discrepancies in code application, aiming to reach a consensus. When a consensus could not be reached, a third researcher acted as a tie-breaker.

4. FINDINGS

Table 2 shows how frequently each code was applied when coding responses to the question asking how BTS’s music supports listeners’ mental health and wellbeing. In the following sections, we discuss the major themes that emerged from analyzing these coded responses, summarizing the connections among the appeal of the music and how the music changed the listeners’ moods resulting in specific mental health related outcomes.

	Code	Description	Freq
Mood Regulation	Comfort	Feeling understood, comforted, and reassured	505
	Uplift	Lifting up spirit and keeping a positive mood	356
	Release	Being able to express the feelings and release the negative emotions	201
	Reflect	Music promotes imagery, insights, introspection	153
	Relax	Feeling revived, relaxed, and energized	112
	Distract	Forgetting about current negative mood and challenging situations	63
	Excite	Intense feeling of arousal, thrills	14
Appeal	Lyrics (Message)	Resonating with the message of the lyrics and/or concept of the song	544
	Diversity	A wide variety of music	107
	Authenticity	Feeling that the artists realistically present themselves	78
	Difficult Topics	Ability to discuss difficult topics like social injustice, mental health, challenges of youth, existential crisis, etc.	60
	Sound (Melody)	Comments on liking how the song sounds and/or the melody of the song	49
Outcome	Coping	Listeners being able to deal with challenges, music helping to “ground” them, helping them to get by every day	492
	Connection	Feeling connections to the artists, music, and other fans; comments on shared thoughts and experiences	293
	Acceptance	Feeling accepted by others/themselves in terms of their identity and experience	197
	Self-growth	Learning something about oneself or the world; being inspired to take on tasks leading to self-improvement	146
	Empowerment	Feeling strength, courage, and confidence	54

Table 2. Frequency of codes.

4.1 Feeling Comforted, Understood, and Not Alone

Finding comfort through BTS’s music was the most frequently mentioned way of regulating mood. Participants mentioned “feeling safe” and “understood” during situations that were challenging, frustrating or unfulfilling. When discussing comfort, participants often imagined the songs giving them a hug (e.g., “The lyrics are like a warm hug” (P475), “a pat on the back” (P91), or holding their hands (e.g., “I feel like they hold my hand and say “hey it’s okay we can get through this together” rather than “oh your [sic] are sad here’s a sad song.” (P313)).

Their songs just made me feel safe. 2! 3! hugs me and holds my hand to hope for better days with me [...] The first time I listen to mono, I looked for a pen and wrote a poem - I haven’t written a poem in 2 years. Their songs make me feel alive. Their music is always here to tell me that I’m not alone. Their voices soothes [sic] my worries. (P55)

Many participants noted that the position the artists take when they convey a positive message does make a difference in how relatable they are, especially when participants can see the artists are “also experiencing such emotions and pains” (P750). Thus, this feeling of comfort was strengthened through the authenticity of the artists and their message. P764 explains how BTS’s music does not try to “sugarcoat” the situation or give unrealistic hope.

I think it's especially important to note that while they give hope through this, it's not baseless hope or toxic positivity [...] they don't give false hope that things will magically get better - instead the boys say that while things probably won't change, tomorrow is still a new day to start over with. This realistic view can be found in many of BTS' songs and it's this that really supports my mental health [...]" (P764)

This sense of being understood and comforted led to many participants feeling intimately connected to the songs, artists, and others:

It feels like I just shared my deepest fears with someone and that person's telling me that it's okay, that they understand me, that they got through it and therefore so can I. (P673)

These imagined connections can be powerful, especially when used as mechanisms for processing difficult situations, releasing negative emotions, and grounding oneself to overcome those situations. P47 shares:

[...] listening to the songs makes me feel like if I and they keep a very nice secret together and understand each other completely. As if they were my dear and closest friends who understand the bad things that happen in my life and always have the perfect words to tell me [...] One month ago, my brother passed away due covid-19. You can imagine how I felt. The moment I received the news he was being incinerated, the same exact minute, the preview for the winter package 2021 was released. With the images of them lighting matches, I felt as if they were honouring my brother's memory. I felt so supported and shocked at the same time. It was incredible. (P47)

4.2 Self-growth through Understanding and Accepting Oneself

Many participants reported feeling cathartic after listening to BTS's music, commenting on how it helped them understand and articulate their own emotions. Participants who were considering suicide or had attempted suicide in the past said BTS's song *Magic Shop* supported them when they "found" it.

I was in a dark place and they were and still are my light. i was about to commit suicide, but then i came across their song magic shop and never looked back. their lyrics is my therapy and my healing. At the end of a bad day, i can always look for them. Their songs are so comforting and it's like they know exactly what to say to you. i'm not someone that show my emotions or feelings easily but when i'm listening to their music, it feels like i can let it all out. finding bts wasn't finding just a boy band. it was finding a home, they are like a warm hug, a hot chocolate on a winter day, the sun after rain. (P111)

BTS' music is very therapeutic for me. Their sound and lyrics speak right to my heart and have awoken feelings that were long asleep. (P874)

Numerous similar responses suggest that listeners perhaps compartmentalize and hide from their feelings leading to "feeling numb" or being unable to verbalize their emotions, and the music serves as a catalyst to unlock those feelings. Participants shared how they found the release of such thoughts and feelings helpful for their mental health, often accompanied by crying. P608 shared:

I am the kind of person who generally doesn't cry unless I am really angry. Thus, I am not able to express well when I am sad or hopeless and keep my feelings bottled up inside me. But when I listened to Zero o'clock, it was the first time that I cried just

because I was sad. At that point of time I realised how important was BTS for my life. After I actually cried and let out my feelings, I feel a lot better now. (P608)

Participants also stated how the messages in the songs help them think about and discuss difficult issues they have sometimes been avoiding or with social stigma.

Their music has the power to invite me to reflect about subjects that I often avoid, like fears of failure, imperfectness, hopelessness, and loss. But those are things that I need to face to learn about myself. Listening to their music feels like learning the journey of life together with them...sometimes painful, sometimes playful, another time full of anger, and there are joys and comfort too, but never alone. (P779)

[...] Their overall message of normalizing mental and the conversation of mental health in society is so important for me. I have OCD, Anxiety, and Depression, and BTS talking about mental health, writing about it, sharing their personal experiences with mental health issues helped create an environment free of judgment and doubts. (P260)

This kind of reflection also led to opportunities for introspection, self-acceptance, and self-growth. Participants shared how listening to certain songs led them to challenge their way of thinking, change their behavior, or take concrete actions to better their mental health:

When I was afraid to go to counselling, yoongi's interlude shadow motivated me because if he can share his fears with millions, I can share it with at least one person. (P559)

Their music can make me feel emotions i didn't even know they existed...when i listen to their songs i feel like a [sic] learn something new about myself, them, the world, society etc. Through their music I've learnt to love and appreciate myself and others and they make me feel safe, comfortable like when i listen to them i feel like I'm home. (P561)

P912 noted this change to positive moods sometimes led to a long-term impact beyond when they were actively listening to music.

I didn't noticed [sic] it first but after becoming army, I have become cheerful and confident (even though I'm [sic] typical introvert). Now it's very hard to upset me for so long I quickly get over it because of their music. (P912)

4.3 Intentional Coping through Various Music

The diversity of BTS's music emerged as an appeal. Participants talked about accessing specific songs when they wanted to attain a certain mood such as energy for work, relaxing after a tough day, or calming anxiety.

On my daily life, BTS music gives me comfort and companionship. I curate my BTS music/MV playlists depending on my mood or activities, for example I choose cheerful and/or energetic songs to make boring chores more palatable, comforting and beautiful songs when I'm feeling anxious or unsettled, soft and chill songs to help me focus when I'm reading or working. (P328)

In addition to feeling comforted or cathartic which was common for participants, the music also helped them sometimes forget about their troubles or worries.

Putting on my earphones and listening to BTS helps me escape, disconnect from my real life, stop hearing the screams and negative words around me. Also the lyrics of songs like Paradise and 00 o'clock give me peace with myself, reduce my anxiety and self

blaming. I have a special playlist I use when I need to sleep without hearing my parents arguing or cry one [sic] in my room [...] (P21)

However, most participants did not simply dwell in escapism; listening to specific songs could also help them feel more focused, empowered, and persistent, helping them deal with real-life challenges. P886 explains:

The messages in the music keep me positively motivated to not give up day to day. My husband died of cancer in December 2019 after two years of treatments. [...] Certain songs like Not Today kept me focused on pushing through during my husband's treatment and Spring Day is still my song for grief. Even Life Goes On has made me feel seen and understood. I stay focused on my personal value and persistence and on forgiving myself for failures through the music [...] (P886)

Participants also talked about having a rush of positive feelings from certain songs and they specifically sought them out when they needed a “serotonin boost” (P679). Some expressed their feelings by singing and dancing along or studying and memorizing the lyrics. Through making strategic choices of music from the diverse discography of BTS and having go-to songs for achieving different moods, fans were able to better cope with difficult situations and have something to look forward to and enjoy. Some expressed how essential listening to BTS has become in their daily lives, such as P561:

[...] i don't think that i will ever stop listening to them because their music has helped me get through a lot so i can't just stop they're a part of my lifestyle at this point...there isn't even a day without listening to AT LEAST one song of theirs...i can't live without music and above all without BTS music. (P561)

5. DISCUSSION

5.1 Connection of Listeners and Songs through Support from Textual and Visual Elements

Many participants personified BTS’s songs and lyrics as a “friend” or “companion”, often experiencing physical sensations of comfort while listening to the songs. These imagined connections provided real comfort to the participants and helped them develop emotional resilience for self-coping. Visualizing a place or person of comfort through associated imagery and externalizing thoughts that have been repressed by relating to lyrics are therapeutic strategies which participants learned and experienced through these songs, akin to tapping into positive imagination for recovering from trauma [42].

Researchers have found that situations that are more difficult to tackle than one has emotional, social, and physical resources for, can lead to avoidant coping mechanisms such as denial or escapism [43]. BTS’s lyrical message helped participants find comfort and cope by balancing escapism, distraction, and/or the permission to feel the reality of their experience. For a few fans, BTS’s music helped them escape or forget their difficult reality momentarily which they could not control (e.g., the pandemic or abusive household). For others, it helped them connect with reality, and provided space for articulating and normalizing negative experiences, feelings of loss, and emotions that are a part of one’s lived experience but are hidden, stigmatized, and/or feared in the society. The therapeutic value of such

lyrics occurs when repressed emotions and thoughts which are otherwise alienated and hated upon by the individual and society are now heard, acknowledged, and given a healthy outlet (e.g., crying, writing, or singing). This reduces the conflict within self and helps participants accept and welcome a range of emotions (e.g., [44]). BTS’s lyricism addressing diverse life circumstances acknowledges two kinds of hope (1) pursuing an ideal dream and (2) reducing the conflict in accepting the authentic, imperfect reality of change. The two types of hope helped participants to relate to and cope with the reality of mundane and negative experiences.

Developing routine coping practices such as reading, writing, or journaling about the lyrics demonstrated reflection and self-growth. Participants relied on subtitles in some music videos and fan-translated lyrics to learn the message of the song. The importance of visual musical experiences has also been noted in [45] as they discuss the dominance of YouTube videos containing music/lyrics.

Design implication: Providing visual cues for songs can enhance the user experience by providing different metaphors for personification and connections. Real-time translation of lyrics can support coping by reducing the burden of navigating multiple tools. Additionally, lyric videos allow listeners to not only process the text more effectively, but also visualize it as they hear the songs again.

5.2 Connections with Artists beyond Music

Most participants experienced connections with artists beyond music through the artists’ live conversations with fans, personal stories, struggles, behind the scenes stories of the song/album, art, and imagery. These participants valued the authenticity of the artists in various types of work, explaining how easy it was to relate to them on a personal level.

These connections with the fandom often break the boundaries of how the artists portray themselves in the public eye and their private lives. As fans perceived BTS’s music, performance, and actions to be congruent with their real-life personalities, they felt a deeper connection with the authenticity of the artists and their message. Often, fans cited being better versions of themselves like BTS thus, not only idolizing the artists but also taking actions in the real world inspired by the artists.

Design implication: There is value in integrating music services with streaming of other media and stories to support fans in making and sustaining these connections and fostering open discussions on lifestyles and issues relevant to their real life. A good example is the *Weverse* app which integrates the performance video, vlogs and other non-music video content, magazine articles, etc. This helps to collectively provide a better picture of the artist, allows fans to feel closer to and further relate to the artist, and reduces the burden of switching between multiple platforms.

5.3 Connections with Other Fans through Discussion of Music

Some participants spoke about feeling an abstract connection with “others out there” who might be listening and relating to the same music. This reduced their sense of

loneliness. Some discussed songs and lyrics with other ARMYs and bonded with them, even learning to share openly (e.g., *“It helped me open up to the friends I met through their music and ultimately helped me to go get professional help.”*(P171)).

As interpretation of art is abstract and variable, fandoms are known to theorize and make meaning of music videos and songs together, especially in the context of transmedia storytelling [18, 20]. Online platforms such as “songmeanings.com” is an example of such fan engagement. There is also appeal in the comments left for songs on platforms like Soundcloud which show comments with the timestamp of the song, potentially providing an opportunity for ‘singing along’ in comments, externalizing emotions felt, and/or reflecting on the music or lyrics while listening in real time [46]. Live streaming and live chats on music streaming platforms also allow for the experience of real time connections [47]. User interactions are often observed in the comments section for live BTS performances on YouTube. Stronger connections between social media and CMS may be beneficial with more social features in music streaming services such as a post discussing lyrics, theories, or reflections connected to the art and message [48]. As communities centered around music continue to cross boundaries of language and culture, it is important to celebrate such diversity on CMS by supporting translation and interpretation of lyrics in different cultural contexts.

Design implication: Fans often curate social media posts to add lyrics and streaming links, sometimes going above and beyond to edit video clips and/or add lyrical translations of parts of the song they want to share or discuss. Adding a feature in streaming services which allows users to take snippets of the song, add a translation or their own interpretation of the lyrics, and share and discuss them in-context, would make such experiences more accessible to those without sophisticated video editing skills.

5.4 Playlists and Metadata Supporting Different Mood Regulations

In our data, we saw that BTS’s music helped participants both maintain and achieve diverse emotional states such as positive moods, relaxation, release of negative moods, and comfort. Digital mental health interventions can be a system-driven “push” wherein the system recognizes appropriate time to intervene (just-in-time adaptive interventions (e.g., [49])) or a user-driven “pull” wherein the participant actively seeks out the intervention and personalizes it to their own context [50]. In the context of music and mental health, we draw parallels to these actions as recommendations of songs that were pushed to participants allowed them to discover helpful songs serendipitously. Some participants also spoke about actively searching or browsing helpful songs and curating playlists for when they wanted to be in a certain mental state.

Explaining a serendipitous discovery, many participants talked about finding the right song at the moment when they needed it. As discussed above, in some cases, the song and its message led to the participant realizing how they were feeling, leading to catharsis of intense emotions they were bottling up inside such as grief or loss during the COVID-19 pandemic [51]. The songs and lyrics

provided an external voice for their thoughts. Additionally, coping with repeated lyrics of the song as a “mantra” can help reduce intrusive thoughts and calm anxiety [52].

BTS’s diverse discography was cited by many participants as an aspect that helped them relate to every mood that they experienced. Catharsis without comfort can be difficult to process alone if an individual is unable to get out of their negative state of mind. Therefore, supporting diversity in general when recommending music to facilitate mood regulation (e.g., mixing up comforting songs with songs that may talk about potentially triggering emotions and experiences such as suicide) may be important.

Design implication: This strong association of mood with music implies that CMS should expand their taxonomy to include mood and lyrics for recommendations. Platforms should recommend broader mood-based playlists that support a range of emotions using metadata from the lyrical content of the song and the community’s responses instead of just basing playlists on the current mood of the listener or the dominant mood of the songs. The outcome or change in mood that fans experience should also be considered and tagged.

6. CONCLUSION AND FUTURE WORK

Our study shows how BTS’s music supports fans’ mental health and wellbeing through a variety of mood regulations, allowing them to cope, reflect, and grow by listening to their music, and even fundamentally impacting their behavior or outlook in some cases.

Studying one particular fandom from an emic approach allows us to gain a deeper understanding of the context and culture of the community and the discography of the artist to better understand the user data, resulting in rich discussion. However, such a study design also has limitations. First, the study focuses specifically on the fans of BTS and thus the findings and design implications are more relevant to popular music with lyrics. Future research with fans of artists with vastly different kinds of music will help expand our knowledge on how music can support the listeners’ mental health. Second, despite the large number of respondents, the sample does not represent the whole fandom given its size and global nature, especially due to the fact that the survey was administered in English. Further research is needed to better understand fans in different languages and cultural contexts. An investigation of male-dominated music fandoms could also offer additional insights. In our future work, we plan to explore the impact of individual songs on listeners to understand whether the impact can be long-lasting, resulting in more substantial changes to listener’s behavior or outlook.

While the findings discussed in this paper are specific to BTS’s music, the ways in which fans interact with, look up to, and imagine and feel connections with the songs and artists are not unique to just BTS or K-pop. Numerous Western artists such as Halsey, Lady Gaga, Coldplay, Demi Lovato, and Logic, to name a few, openly discuss and make songs about mental health issues, and their fans use music to manage and discuss their own mental health. The design implications derived from our findings can also support fans of other artists and in general, help people use music for supporting their mental wellbeing.

7. ACKNOWLEDGEMENTS

We sincerely thank all the BTS ARMYs for their participation in this study, and their thoughtful and detailed responses. We also thank BTS for inspiring this research, and their positive influences on ARMYs' mental health and wellbeing, especially during this challenging time of the global pandemic.

8. REFERENCES

- [1] A. C. North and D. J. Hargreaves, "The power of music," *The Psychologist*, vol. 22, pp. 1012-1015, 2009.
- [2] S. B. Hanser, "Music, health, and well-being," in *Handbook of Music and Emotion: Theory, Research, Applications*, USA: Oxford University Press, 2010.
- [3] R. A. R. MacDonald, G. Kreutz, and L. A. Mitchell, *Music, Health, and Wellbeing*, USA: Oxford University Press, 2012.
- [4] D. Knox and R. MacDonald, "The role of technology in music listening for health and wellbeing," *Journal of Biomusical Engineering*, vol. 3, no. 1, 2015.
- [5] S. Saarikallio and J. Erkkilä, "The role of music in adolescents' mood regulation," *Psychology of Music*, vol. 35, no. 1, pp. 88-109, 2007.
- [6] Z. E. Papinczak, G. A. Dingle, S. R. Stoyanov, L. Hides, and O. Zelenko, "Young people's uses of music for well-being," *Journal of Youth Studies*, vol. 18, no. 9, pp. 1119-1134, 2015.
- [7] J. Woelfer and J. H. Lee, (2012). "The role of music in the lives of homeless young people: a preliminary report," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Porto, Portugal, 2012, pp. 367-372.
- [8] J. H. Lee, L. Pritchard, and C. Hubbles, "Can we listen to it together?: factors influencing reception of music recommendations and post-recommendation behavior," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Delft, The Netherlands, 2019.
- [9] X. Hu, J. Chen, and Y. Wang, "University students' use of music for learning and well-being: a qualitative study and design implications," *Information Processing and Management*, vol. 58, 2021.
- [10] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: a state of the art review," in *Proc. of the Intl. Conf. on Music Information Retrieval*, 2010, pp. 255-266.
- [11] T. Li and M. Ogihara, "Detecting emotion in music," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Baltimore, MD, USA, Oct. 2003.
- [12] X. Hu and J. S. Downie, "Exploring mood metadata: Relationships with genre, artist and usage metadata," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Vienna, Austria, 2007, pp. 462-467, 2007.
- [13] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. Vlahavas, "Multilabel classification of music into emotion," in *Proc. of the Intl. Conf. on Music Information Retrieval*, Philadelphia, PA, USA, 2008.
- [14] K. Bischoff, C. S. Firan, R. Paiu, W. Nejd, C. Laurier, and M. Sordo, "Music mood and theme classification-a hybrid approach," in *Proc. of the Intl. Society for Music Information Retrieval Conf.*, Kobe, Japan, 2009.
- [15] Y. Hu, X. Chen, and D. Yang, "Lyric-based song emotion detection with affective lexicon and fuzzy clustering method," in *Proc. of the Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.
- [16] C. Laurier, M. Sordo, J. Serra, and P. Herrera, "Music mood representation from social tags," in *Proc. of the Intl. Society for Music Information Conf.*, Kobe, Japan, 2009.
- [17] S. Y. Park, A. Laplante, J. H. Lee, and B. Kaneshiro, "Tunes together: Perception and experience of collaborative playlists," *Proc. of the Intl. Society for Music Information Conf.*, Delft, The Netherlands, 2019.
- [18] J. H. Lee and A. T. Nguyen, "How music fans shape commercial music services: A case study of BTS and ARMY," in *Proc. of the Intl. Society for Music Information Conf.*, Montréal, Canada, 2020.
- [19] C. Grover, R. Ciocirlea, and N. Santero. *2020 Results: BTS ARMY Census*. Available: <https://www.btsarmy.census.com/results>.
- [20] J. Lee, *BTS, Art Revolution: BTS Meets Deleuze*, Seoul, South Korea: Parrhesia Publishers, 2018.
- [21] J. Lee, *BTS and ARMY Culture*. Seoul, South Korea: CommunicationBooks, 2019.
- [22] S. Blady. *Bulletproof to Stigma*. Speak Up. 2019. Available: <https://www.speak-up.co/bulletproof-to-stigma>
- [23] S. Rubin, "Strong experiences with music (SEMs) as experienced by ARMY," presented at BTS: A Global Online Interdisciplinary Conference, 2021.
- [24] G. T. Regout, "Music and (narrative) psychology: A literature review," *Psychotherapy*, Sec. 60, 2020.
- [25] W. E. J. Knight and N. S. Rickard, "Relaxing music prevents stress-induced increases in subjective anxiety, systolic blood pressure, and heart rate in healthy males and females," *Journal of Music Therapy*, vol. 38, no. 4, pp. 254-272, 2008.
- [26] R. E. Thayer, R. Newman, and T. M. McClain, "Self-regulation of mood: strategies for changing a bad mood, raising energy, and reducing tension," *Journal of Personality and Social Psychology*, vol. 67, no. 5, pp. 910-925, 1994.
- [27] U. Nilsson, "The anxiety- and pain-reducing effects of music interventions: A systematic review," *Aorn Journal*, vol. 87, no. 4, pp. 780-807, 2008.
- [28] M. L. Chanda and D. J. Levitin, "The neurochemistry of music," *Trends in Cognitive Sciences*, vol. 17, no. 4, pp. 179-193, 2013.

- [29] J. E. de la Rubia Ortí, M. P. García-Pardo, C. C. Iranzo, J. J. C. Madrigal, S. S. Castillo, M. J. Rochina, and V. J. P. Gascó, "Does music therapy improve anxiety and depression in alzheimer's patients?," *The Journal of Alternative and Complementary Medicine*, vol. 24, no. 1, pp. 33-36, 2018.
- [30] A. Kresovich, "The influence of pop songs referencing anxiety, depression, and suicidal ideation on college students' mental health empathy, stigma, and behavioral intentions," *Health Communication*, 2020.
- [31] M. E. S. Reyes, A. G. F. Santiago, A. J. A. Domingo, E. N. Lichingyao, M. N. M. Onglengco, and L. E. McCutcheon, "Fandom: Exploring the relationship between mental health and celebrity worship among Filipinos," *North American Journal of Psychology*, vol. 18, no. 2, pp. 307–316, 2016.
- [32] D. A. Laffan, "Positive psychosocial outcomes and fanship in k-pop fans: A social identity theory perspective," *Psychological Reports*, 2020.
- [33] D. L. Wann, E. Brame, M. Clarkson, D. Brooks, and P. J. Waddill, "College student attendance at sporting events and the relationship between sport team identification and social psychological health," *Journal of Intercollegiate Sports*, vol. 1, pp. 242- 254, 2008.
- [34] S. E. Smith, F. G. Grieve, R. K. Zapalac, W. P. Derbyberry, and J. Pope, "How does sport team identification compare to identification with other social institutions?," *Journal of Contemporary Athletics*, vol. 6, pp. 69-82, 2012.
- [35] C. Schroy, C. N. Plante, S. Reysen, S. E. Roberts, and K. C. Gerbasi, "Different motivations as predictors of psychological connection to fan interest and fan groups in anime, furry, and fantasy sport fandoms," *The Phoenix Papers*, vol. 2, no. 2, pp. 148-167, 2016.
- [36] S. Reysen, C. Plante, and D. Chadborn, "Better together: Social connections mediate the relationship between fandom and well-being," *AASCIT Journal of Health*, vol. 4, no. 6, pp. 68-73, 2017.
- [37] M. Choi, H. Cho, and Y. Kim, Young-Ah, "The effect of internet usage, fandom activities and sense of community on adolescents' mental health in the digital era," *Journal of Digital Convergence*, vol. 14, no. 9, pp. 349–358, 2016.
- [38] L. B. McInroy, "Building connections and slaying basilisks: Fostering support, resilience, and positive adjustment for sexual and gender minority youth in online fandom communities," *Information, Communication & Society*, vol. 23, no. 13, pp. 1874-1891, 2020.
- [39] J. Corbin and A. Strauss, *Basics of Qualitative Research (4th ed.)*, SAGE Publications, Inc., 2015.
- [40] K. Holtzblatt, J. B. Wendell, and S. Wood, *Rapid Contextual Design: A How-to Guide to Key Techniques for User-centered Design*, Elsevier, 2004.
- [41] C. Hill, B. Thompson, and E. Williams, "A guide to conducting consensual qualitative research," *The Counseling Psychologist*, vol. 25, no. 4, pp. 517–572, 1997.
- [42] L. Parnell, and D. J. Siegel, *Attachment-focused EMDR: Healing relational trauma (First ed.)*, New York ; London, W.W. Norton & Company, 2013.
- [43] Glanz, K. and Schwartz, M. D., "Stress, coping, and health behavior", in *Health behavior and health education: theory, research, and practice*, Glanz, K., Rimer, B. K., & Viswanath, K., Ed. John Wiley & Sons, 2008, pp. 211-236.
- [44] H. J. Alberts, F. Schneider, and C. Martijn, "Dealing efficiently with emotions: Acceptance-based coping with negative emotions requires fewer resources than suppression," *Cognition & Emotion*, vol. 26, no. 5, pp. 863-870, 2012.
- [45] J. H. Lee, H. Cho, and Y. S. Kim, "Users' music information needs and behaviors: Design implications for music information retrieval systems," *Journal of the Association for Information Science and Technology*, vol. 67, no. 6, pp. 1301-1330, 2016.
- [46] C. Hubbles, D. W. McDonald, and J. H. Lee, "F#@ that noise: SoundCloud as (a-)social media?," in *Proc. of ASIS&T*, 2017.
- [47] K. Pires and G. Simon, " YouTube live and Twitch: A tour of user-generated live streaming systems," in *MMSys '15*, New York, NY, USA, 2015, 225–230.
- [48] A. N. Hagen, and M. Lüders, "Social streaming? Navigating music as personal and social," *Convergence*, vol. 23, no. 6, pp. 643-659, 2017.
- [49] J. M. Smyth and K. E. Heron, "Is providing mobile interventions" just-in-time" helpful? An experimental proof of concept study of just-in-time intervention for stress management," in *IEEE Wireless Health (WH)*, Oct. 2016, pp. 1-7.
- [50] A. Bhattacharya, R. Vilardaga, J. A. Kientz, & S. A. Munson, "Lessons from practice: Designing tools to facilitate individualized support for quitting smoking," *ACM Transactions on Computer-human Interaction : A Publication of the Association for Computing Machinery*, pp. 3057–3070, 2017.
- [51] M. Oh, *How BTS Sings of Healing the Mind*. Weverse Magazine. 2021. Available: <https://magazine.weverse.io/article/view?lang=en&num=104>
- [52] J. R. Doty, *Into the Magic Shop: A Neurosurgeon's Quest to Discover the Mysteries of the Brain and the Secrets of the Heart*, Penguin, 2017.

CROSS-CULTURAL MOOD PERCEPTION IN POP SONGS AND ITS ALIGNMENT WITH MOOD DETECTION ALGORITHMS

Harin Lee^{1,2} Frank Höger² Marc Schönwiesner^{1,3} Minsu Park⁴ Nori Jacoby²

¹Max Planck Institute for Human Cognitive and Brain Sciences

hlee@cbs.mpg.de

²Max Planck Institute for Empirical Aesthetics

{frank.hoeger, nori.jacoby}@ae.mpg.de

³Leipzig University

marcs@uni-leipzig.de

⁴New York University Abu Dhabi

minsu.park@nyu.edu

ABSTRACT

Do people from different cultural backgrounds perceive the mood in music the same way? How closely do human ratings across different cultures approximate automatic mood detection algorithms that are often trained on corpora of predominantly Western popular music? Analyzing 166 participants' responses from Brazil, South Korea, and the US, we examined the similarity between the ratings of nine categories of perceived moods in music and estimated their alignment with four popular mood detection algorithms. We created a dataset of 360 recent pop songs drawn from major music charts of the countries and constructed semantically identical mood descriptors across English, Korean, and Portuguese languages. Multiple participants from the three countries rated their familiarity, preference, and perceived moods for a given song. Ratings were highly similar within and across cultures for basic mood attributes such as *sad*, *cheerful*, and *energetic*. However, we found significant cross-cultural differences for more complex characteristics such as *dreamy* and *love*. To our surprise, the results of mood detection algorithms were uniformly correlated across human ratings from all three countries and did not show a detectable bias towards any particular culture. Our study thus suggests that the mood detection algorithms can be considered as an objective measure at least within the popular music context.

1. INTRODUCTION

Music can express a range of emotions, from melancholy and sadness to love and joy. Since music has the powerful ability to reflect and modify one's emotional state [1], researchers have explored how people perceive emotion in music from diverse angles [2]. However, while the universality and variability of musical properties across cultures have been reported in many aspects [3,4], cross-cultural congruence of perceived moods in music has yet to reach a clear agreement [5]. In this paper, we present an empiri-

cal analysis of cross-cultural experiments that characterizes agreements and discrepancies on perceived moods in music by comparing between three cultures and also between human judgments and four algorithmic estimates.

1.1 Background

Many cognitive models have been proposed to account for emotional experience, and these can also be applied to account for the case of music. Basic emotion theory relies on discrete categories of emotions, whereas a competing explanation relies on dimensional models to argue that all affective states arise from a few independent affective dimensions [6]. In the context of music, previous studies have suggested that basic emotions such as happiness, sadness, and anger can be universally recognized in music by demonstrating that listeners can correctly deduce an intended emotion from unfamiliar musical traditions [7,8]. Nevertheless, critics have argued that the application of basic emotion theory to music is too simplistic and it fails to capture the full emotional richness expressed through music (see Eerola & Vuoskoski for a review [2]). Consequently, other scholars have proposed music-specific emotion models, including nine [9] or thirteen [10] dimensions.

Many researchers used such high-dimensional models and provided extensive empirical support. Yet, since such studies were based mainly on homogeneous Western participants (e.g., college students at a large research university [11]), it still remains unclear how these mood dimensions can be applied cross-culturally. Although significant cross-cultural differences were recently reported in multinational comparisons (e.g., [12,13]) or through non-US contexts (e.g., [14,15]), these studies are also limited in testing within a single musical style or a single population, respectively.

There is also a prevalent issue in the choice of lexical semantics when comparing between languages. That is, the subtle differences in the nuance of the mood terms translated into other languages can result in the varied interpretation of the meanings. Recent research has shown that the semantic alignment of emotion terms is highly variable across languages [16,17]. Thus, the cross-cultural differences observed in the aforementioned studies could have partly been driven by varied interpretations of word meanings rather than by the perception of music itself.



© H. Lee, F. Höger, M. Schönwiesner, M. Park, and N. Jacoby. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** H. Lee, F. Höger, M. Schönwiesner, M. Park, and N. Jacoby, "Cross-cultural mood perception in pop songs and its alignment with mood detection algorithms", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

In the domain of music information retrieval (MIR), multiple automatic mood detection algorithms have been introduced within the last decade. These algorithms attempt to predict a listener’s perception of high-level mood attributes in songs such as *danceability* and emotional *valence*. Such technology has shown to have diverse applications, including music recommendation [18], hit song prediction [19,20], and investigation of musical trends over time [21], all of which rely on quantifying large corpora of music. However, the generalizability of these tools is concerning because these mood detection algorithms are often trained on databases predominantly consisting of English-language songs judged by Western annotators (e.g., Billboard HOT100 [22] and Million Song Dataset [23]). It therefore raises important questions on how robustly these measures would perform on non-Western songs and reflect the perception of non-Western listeners.

1.2 Research Question

The challenges and opportunities in studying cross-cultural applicability of perceived moods in music lead us to introduce three research questions that guide the remainder of this paper:

RQ1. Do people with different cultural backgrounds perceive mood in music differently?

RQ2. Is there a more robust agreement within and across cultures for certain moods compared to others?

RQ3. How well do mood detection algorithms in MIR approximate human judgments, and is there a cultural bias in the algorithms?

We address the first two questions with a cross-cultural comparison between Brazilian, South Korean, and American raters, examining how they perceive nine types of moods in music. We chose these populations since they are geographically and culturally distant and speak different languages. Moreover, as recent evidence has shown that shared music interests around the world generate unique clusters in the West, Asia, and Latin America [1,24], we wanted to include one country from each cultural region in our study.

Many studies of musical emotions rely on film music excerpts or hand-picked musical stimuli that have been shown to evoke certain emotions [2]. In our design, we built a novel dataset of pop songs, randomly drawn from major music charts in those three countries, in order to test in a setting that is closer to real-world experience (see section 2.1). To begin, we used visual stimuli to validate whether the translated mood terms convey the exact intended meanings in the three languages (see section 3.3). We then ran identical online experiments in each country, asking participants to rate songs from all three cultures according to preference, familiarity, and nine semantic lexical mood descriptors (see section 2.3). We compared the similarity of mood ratings both within and across countries

and identified a set of mood attributes that were more highly agreed upon than others.

To address RQ3, we used Spotify’s API to retrieve four high-level mood features (*danceability*, *energy*, *valence*, and *acousticness*) and assessed how well these algorithms approximate human raters. Next, we examined whether there is a Western bias in the algorithms by testing if MIR values align better with raters from the US.

2. MATERIALS

2.1 Song Selection

We established a novel and balanced dataset of 360 pop songs originating from Brazil, S. Korea, and the US. We retrieved the major music charts for songs that made the charts between 2010 and 2019 (US: Billboard HOT 100¹, Brazil: Crowley Broadcast Analysis² and Top-40 Charts³, Korea: Gaon Music Chart⁴).

We ran search queries for all unique songs using Spotify’s Web API, retrieving a maximum of 50 results. For every search, a fuzzy string match ratio between 0 and 1 was computed based on the Levenshtein distance between the query and result strings, normalized by the maximum possible distance. We selected the entry that had the highest value and excluded songs with no results above a ratio of 0.7. We also manually inspected a subsample of 100 songs to validate the accuracy of the matching process (98% in Brazilian and US songs; 97% in Korean songs). Songs with the word *live* or *remix* in the title or without preview audio (to use as stimuli in the experiment) were also excluded.

The list was further reduced by retaining only the songs that were performed by artists matching the nationality of the chart (e.g., songs by Korean artists in the Korean charts) to balance the musical styles from the three countries. A song’s origin was determined according to the International Standard Recording Code (ISRC), which begins with two alphabet letters that correspond to the standard ISO-2 country code to indicate the place of registration, and we manually inspected the final set.

We randomly sampled 12 songs per year in each country’s list while controlling for duplicate artists. The final dataset consisted of 360 songs, that is, 120 songs from each country with songs distributed evenly across the 10-year window (the full list of songs and the experimental data are available at <https://osf.io/3uw9d/>).

2.2 MIR Mood Features

High-level audio features for all songs were retrieved using Spotify’s Web API (see their reference manual for detailed descriptions of all available acoustic features⁵). Although other MIR libraries such as Essentia [25] also offer a similar set of features, it has been shown that audio codec and level of audio compression can influence the extraction outcome [26]. In fact, comparing Spotify’s feature *danceability* with the same feature extracted using Essen-

¹ Billboard Chart: <https://www.billboard.com>

² Crowley Broadcast Analysis: <https://charts.crowley.com.br>

³ Top40 Charts: <https://www.top40-charts.com>

⁴ Gaon Chart: <http://www.gaonchart.co.kr>

⁵ Spotify Web API: <https://developer.spotify.com/documentation/web-api>

tia with low-quality preview audio resulted in small correlation ($r = .28$). Due to this limitation, we decided to only include Spotify’s features in this study. Unfortunately, Spotify’s algorithm is not open-source and we leave open the task of validating and comparing between different MIR algorithms to future researchers.

2.3 Selection of Mood Attributes

We used the nine-factor Geneva Emotion Music Scale (GEMS) [9] and the 13 emotion dimensions proposed by Cowen and colleagues [10] as a reference and chose the seven co-occurring dimensions in the two models as our target mood dimensions for investigation. These seven mood attributes were: *energetic/pump-up*, *relaxing/calm*, *dreamy*, *in love*, *joyful/cheerful*, *anxious/tense*, and *sad/depressing*. In addition, we added two extra mood attributes (*danceable* and *electronic*) that are directly comparable with the mood features retrieved through Spotify,⁶ giving rise to nine mood attributes in total.

3. EXPERIMENT

3.1 Participants

We created independent participant pools in Brazil, S. Korea, and the US. The US participants were recruited from Amazon Mechanical Turk (MTurk) and had to pass a series of pre-screening tasks to qualify for the pool. The pre-screening tasks were: a LexTale task [27] to check for English fluency, a headphone screening task [28] to ensure they follow the instructions when asked to wear headphones, and an attention check item to screen for fraudulent behavior.

Since MTurk is not very popular in Brazil and is unavailable in S. Korea, participants from these countries were recruited independently by hiring research assistants in the local area. We required all participants to be native speakers and to reside in their respective countries.

We advertised identical experiments to all three participant pools with the content of the experiment having been translated into their respective languages (section 3.3 describes how we chose the most appropriate words in Korean and Portuguese for the nine mood attributes). In total, 166 participants participated in the study and 11,500 ratings were collected across the 360 songs (see Table 1).

	Brazil	S. Korea	US
Participants (N)	58 (21 F)	54 (27 F)	54 (21 F)
Ratings per mood	3,865	3,859	3,776
Average ratings per participant	66.6 ($SD=33.1$)	71.5 ($SD=33.1$)	69.9 ($SD=33.1$)
Average ratings per song	10.7 ($SD=2.74$)	10.7 ($SD=2.31$)	10.5 ($SD=3.02$)

Table 1. Collected ratings across 360 songs among the three countries. “F” denotes female participants.

⁶ Spotify’s feature *acousticness* was paired with human judgment on *electronic* due to ambiguities in translations. Similarly, while *valence* from Spotify arises from a dimensional emotion model, it was paired with human judgment on *sad* as the closest matching discrete term.

3.2 Experiment Procedure

The stimuli set of 360 songs were divided into 12 blocks, with each block containing 10 songs from each country, evenly distributed across the years. This allowed participants to perform a varied number of blocks while always being presented with a counter-balanced set of stimuli across song origins and years.

After providing their informed consent, participants were randomly assigned to one of the 12 blocks, with each block containing 30 songs (see Figure 1). In each trial, they heard a 20 seconds snippet of a song randomly drawn from the block. They were then asked to identify the gender of the singer (“male”, “female”, or “don’t know”), their familiarity of the song (4 choices ranging from “never heard of it” to “I know it very well”), preference (5-point scale from “dislike a lot” to “like a lot”), and their perception of nine moods presented in a random order (4-point scale from “not at all ...” to “very ...”).⁷ Full questionnaire text is available in supplementary S2.

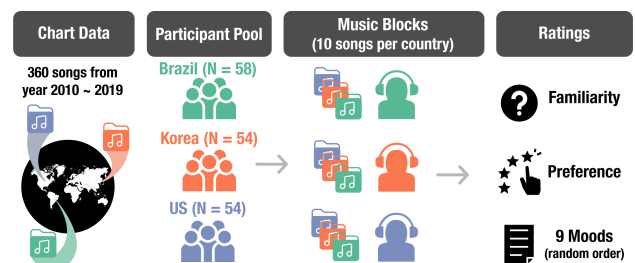


Figure 1. Experimental setup.

The experiment took around 40 minutes to complete and participants were compensated at an hourly rate of \$9. Each participant could participate up to four times, with our system ensuring that they were never assigned to the same block more than once. The number of participants, gender ratio, and the average number of ratings per participant were well balanced across the three countries (see Table 1). All experiments were conducted using *PsyNet* [29], an automated recruitment framework for online experiments. Ethical approval was obtained by the Max Planck Society.

3.3 Validation of Mood Terms

Since the nine mood attributes were based in English, we had to translate those descriptors into Korean and Portuguese. However, word meanings can vary across cultures and this variability can lead to unintended biases [30]. Thus, to ensure that the intended meanings are interpreted in the same way across cultures with different languages, we validated the word meanings in a different modality (i.e., perception of images).

First, we preselected 17 Korean words and 20 Portuguese words that potentially align with the English reference words that we selected from the literature [9,10]. Second, we searched an online stock footage library⁸ with the

⁷ We also collected ratings on the most suitable color to the song. However, this is not discussed as it is beyond the scope of this paper.

⁸ <https://www.shutterstock.com>

target mood terms and compiled a set of images (2 images per target mood term). We then recruited participants (38, 31, and 20 from the US, Brazil, and S. Korea, respectively) from the same participant pool recruited for the main experiment. We asked the participants to rate the relevance of a word to a displayed image on a 4-point scale. Finally, to find the best matching combinations of words across the languages, we computed Pearson correlations across the image items for all possible pairs of every mood dimension. We then selected the triplet word combinations with the highest overall correlations (see supplementary S1 for all word comparisons).

Figure 2B illustrates an example of how we found the best matching translations for mood *dreamy*. Using dictionary definitions for the word “dreamy” in Portuguese (“sonhador”) and Korean (“몽환적인”) yielded a mean correlation of $r = 0.73$, whereas the expression “like in a dream” in Korean (“꿈꾸는 듯한”) and Portuguese (“como um sonho”) resulted in considerably higher semantic alignment ($r = 0.88$). Thus, the latter combination was a more appropriate translation, and we used these words as the guiding passage in the main experiment for the mood dimension *dreamy*. The final combination of words chosen for the nine moods ranged in correlations between 0.88 and 0.98.

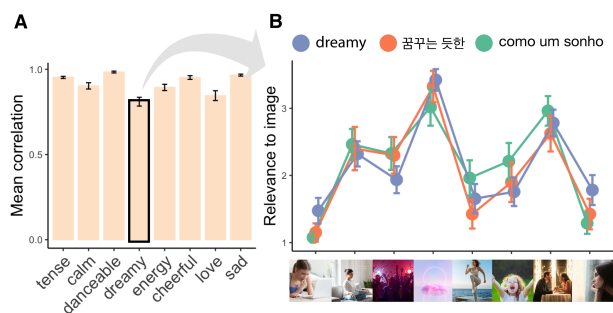


Figure 2. Matching mood semantics across English, Portuguese, and Korean with error bars representing 95% CI. (A) Correlations between word combinations across languages. (B) An example of best-matching triplet word combinations for *dreamy*.

4. RESULTS

4.1 Familiarity & Preference

Familiarity ratings provided by the three countries showed that the raters were most familiar with songs from their own country (see Figure 3A; all $ps < .001$ with Tukey’s HSD), ensuring that our recruited participants and the songs in the dataset were adequate representative samples. While Brazilians and Koreans hardly knew each other’s music (with a mean close to 1 = “never heard of it”), they were relatively familiar with American songs, reflecting the global presence of American pop songs. By contrast, raters from the US knew very little about both Korean and Brazilian songs.

Koreans and Americans also preferred their local music over foreign music (see Figure 3B; $ps < .05$ with Tukey’s HSD), but the difference was small. Moreover, Brazilians did not prefer their local music over American

music. This demonstrates that, while the raters may have been most familiar with their local music, they do not particularly prefer local pop songs over foreign ones.

However, when computing the correlations between familiarity and preference at individual participant level, there were strong correlations among all countries (Brazil: $r = .67$, Korea: $r = .66$, US: $r = .49$, all $ps < .001$; see Figure 3C). These results are consistent with numerous studies that have observed a strong relationship between a listener’s familiarity and preference in music [31].

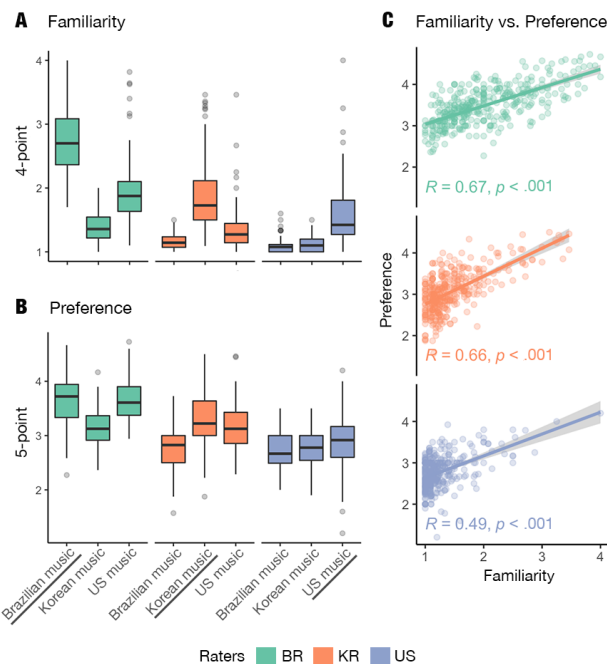


Figure 3. (A) Familiarity and (B) preference of songs among the three countries shown with box and whisker plot separated by song’s origin. The underlined texts represent music that is local to the rater. (C) Scatter plot with general linear model fitting showing the relationship between familiarity and preference at the participant level.

4.2 Within Country Agreement

How are the ratings for the nine moods agreed upon among the raters within each country (RQ1)? Does agreement converge more for some moods than others (RQ2)?

To answer these questions, we examined rater agreement within each country by computing the intraclass correlation (ICC) using two-way random effects with the absolute agreement and multiple raters model [32]. Among the total of 12 blocks that divided the 360 song items, each block consisted on average 9.5 ($SD = 2.62$) unique raters from the same country who completed the same block. The ICC was computed for (i) each block separately and averaged across the 12 blocks for every mood variable (hereafter denoted as ICC_{mean}) and (ii) the perceived *gender of the singer* (used as baseline measure). As an alternative statistical measure, we also used split-half correlation to estimate the reliability. This showed nearly identical values and patterns to the ICC (for details, see supplementary S3).

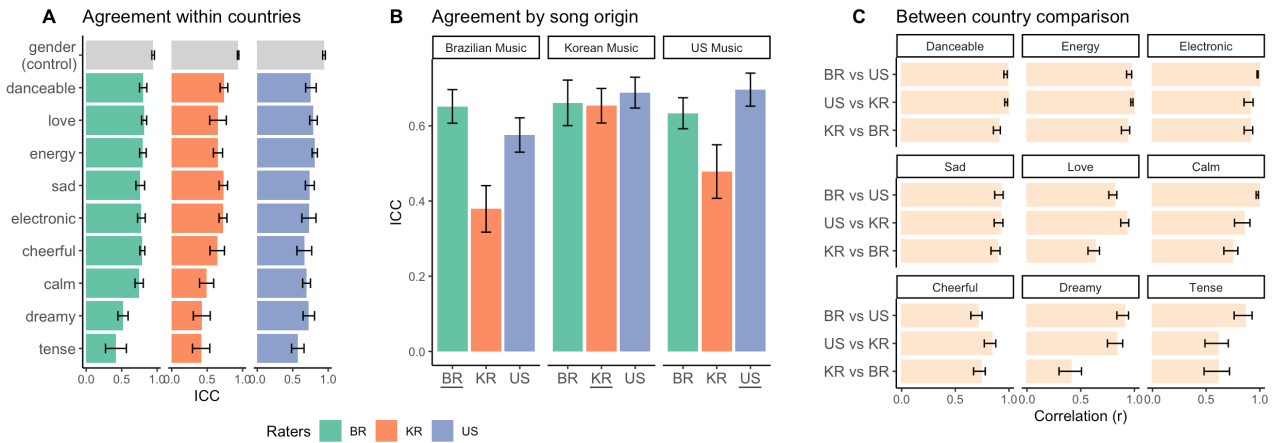


Figure 4. (A) Agreement across the moods within each country. The grey bars represent the baseline measure. (B) Agreement by song origin. Underlined labels in the x-axis representing local music to the rater. (C) Between country pairwise comparison using Pearson correlation corrected for attenuation. All error bars represent 95% CI.

4.2.1 Agreement on Moods

All three countries showed high within-country agreement for moods such as *energy*, *danceable*, *electronic*, *sad*, *cheerful*, and *love* (all $ICC_{mean} > .64$; see Figure 4A), some of which were almost comparable to the baseline measure - *singer's gender* ($ICC_{mean} = .94$). Certain mood variables had a stronger agreement in some countries than in others. For instance, Koreans had a lower agreement for *calm* (.49) compared to the other two countries ($> .70$).

There was a strong agreement about *dreamy* among the Americans (.73) but not within Koreans (.38) and Brazilians (.52). The agreement about *tense* was generally weakest across all three countries (US: .57, S. Korea: .37; Brazil: .38). Across all mood variables, ratings were more consistent among the raters in Brazil ($M = .73, SD = .16$) and the US ($M = .74, SD = .10$) than in S. Korea ($M = .63, SD = .17$).

4.2.2 Agreement on Local vs. Foreign Music

We also examined the raters' agreement for their local music versus foreign music by separating songs according to their origin and averaging across the mood variables (see Figure 4B). Koreans (orange bars) generally showed only a moderate amount of agreement for Brazilian and American music, but a substantially higher agreement for their local music ($F(2,321) = 20.7, p < .001, ges = .114$). Meanwhile, Brazilians and Americans did not particularly converge more for their local music over foreign music.

We found that Korean music received strong agreements from all three countries, with no observable group differences ($F(2, 321) = 0.53, p > .05, ges = .003$). Comparably, Brazilian music and American music received the highest agreement from their local raters but with no significant differences (Tukey's HSD with adjusted $ps > .05$).

The observed pattern is interesting, as one might expect that the intended emotional cues are most easily recognized within the same culture and therefore better converge on their agreement. Previous works have supported this [12,13], contrary to the results we observed. Moreover, while Korean songs were unfamiliar to Brazilians and Americans, they showed high convergence, suggesting

that there may be distinct or *cliché* acoustic properties in Korean music that are easily recognized and agreed upon across different cultures.

4.3 Between Country Agreement

We next investigated how the mood perception in different dimensions aligns between the countries (i.e., BR vs. US, US vs. KR, KR vs. BR) by aggregating rater responses in each country across the song items. When computing correlations between the rater groups, the correlation is attenuated due to the measurement errors in each group. Thus, we corrected for this attenuation [33] by accounting for the internal reliability, which we estimate by the mean ICC values we computed in section 4.2. All correlation hereafter is reported using this corrected version, but the raw correlation is also reported in supplementary S3. In addition, confidence intervals were obtained by sampling 1,000 bootstrapped values with replacement.

When comparing the alignments between countries, we found that four variables, *danceable*, *energy*, *electronic*, and *sad*, are very strongly correlated among all the between-country pairs (all $rs > .93$; see Figure 4C). These four variables were also features that are directly comparable with Spotify's mood detection algorithms. In contrast, considerably lower correlations were observed between Brazilians and Koreans for *love* ($r = .63$) and *dreamy* ($r = .41$). Interestingly, these two countries exhibited much stronger correlations when paired with the US raters (all $rs > .83$). The ratings were highly similar between Brazilian and American raters for *tense*, but showed considerably weaker agreement between the Brazilian vs. Korean and American vs. Korean pairs, possibly due to the shortage of excerpts in our stimuli set manifesting this description.

Together, these results suggest that more basic mood attributes may be perceived similarly across the cultures, while more complex attributes may be perceived differently depending on the listener's cultural background.

4.4 Human vs. MIR

How well does MIR approximate human judgment? Is there a cultural bias in the algorithm? To examine these

set of questions (RQ3), we computed the correlation between all human raters (regardless of nationality) with MIR estimations. We found that on average those correlations range between 0.49 and 0.63 across the variables *energy*, *danceable*, *sad*, and *electronic* (see Figure 5A; all $ps < .001$). These values were substantially lower than the mean correlations observed between the human raters for the same four features (all $rs > .93$).

Although the alignment of perceived mood ratings with Spotify’s features is not weak, such a clear congruence across different cultures suggests that mood detection algorithms can be improved further to better capture these high-level perceptions that are cross-culturally consistent.

4.4.1 Is there a Cultural Bias?

Next, we investigated whether MIR is biased to a particular cultural group. Figure 5B shows the average correlations between human ratings and MIR values across individual song-rate pairs in each country. We found no significant differences between any of the country-country pairs using bootstrapping ($ps > .20$) and also by using a method [34] for significance testing between correlations ($ps > .18$). Thus, we conclude that there is no clearly observable bias in MIR in favoring a certain culture, contrary to our initial prediction that it may align better with the US raters.

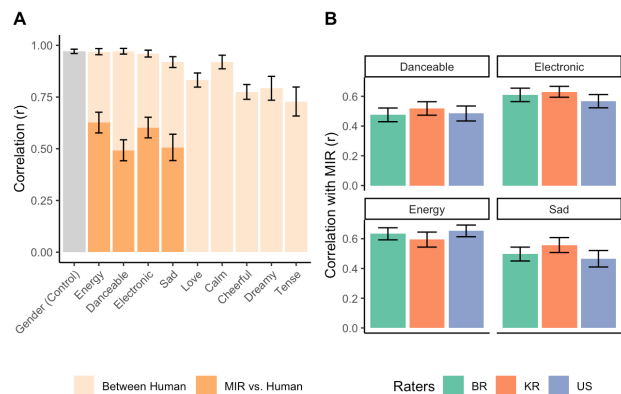


Figure 5. (A) Correlations among people are presented in light orange while correlations between people and MIR are in dark orange. The grey bar represents the baseline measure. (B) Correlations between MIR and raters from three countries. Error bars represent 1SD and all correlations are corrected for the attenuation.

4.4.2 Song Familiarity and MIR

Considering that our participants had a wide range of degrees of familiarity with songs, we also tested whether prior experience with a song influenced the alignment with MIR. We divided the songs into familiar and unfamiliar categories and found that all four MIR features were more strongly correlated with unfamiliar songs than familiar songs: *danceable* ($M_{diff} = 0.13, p = .16$), *electronic* ($M_{diff} = 0.16, p = .02$), *energy* ($M_{diff} = 0.27, p < .001$), and *sad* ($M_{diff} = 0.10, p = .38$), with p-values corrected using Bonferroni correction. This result is consistent with the idea that the listener’s personal relationships to music may result in

higher variability in mood perception (e.g., due to autobiographical memories [35]), which may have been reflected as reduced correlations with the objective algorithms.

5. DISCUSSION

We compared participant ratings from Brazil, South Korea, and the US and examined within and between-country differences in listeners’ perception of nine mood attributes in music. We also assessed the robustness and generalizability of automatic mood detection algorithms in MIR by comparing it with human raters from those three countries.

Our findings reveal that relatively simple mood attributes such as *danceable*, *energy*, *sad*, *cheerful*, and *electronic* are highly agreed upon among the listeners both within and across cultures. This result suggests that pop songs may have intrinsically similar acoustic properties that are reliably recognized regardless of the listener’s cultural background, even when they are unfamiliar with the musical style [7,8]. Some of these properties are low-level features (e.g., tempo, loudness) that can also be picked up by a mood detection algorithm as we found the algorithms to be quite good at approximating human judgments. Nonetheless, the substantial gap between the correlations among humans and correlations of human vs. MIR reveals the current limitation of algorithms failing to capture the full extent of human perception. This implies that people’s mood perception in music may not be explained solely by the acoustic properties. However, recent advances in mood classification incorporating mid-level features [36] (e.g., melodiousness) and multi-modalities [37,38] (e.g., fusing audio and lyrics) show promising paths for improvement.

Our results also show that complex mood attributes may not have concretely shared musical characteristics and are thus perceived relatively differently depending on the listener’s cultural background [12-15]. For instance, the ratings for mood *dreamy* converged well only among the Americans but not within and between the Korean and Brazilian raters. The mood *love* was strongly agreed upon within all three countries, but there was little agreement between Koreans and Brazilians. These cross-cultural differences are striking considering both South Korea and Brazil are highly globalized and thus largely exposed to the mainstream media. We would expect significantly larger cross-cultural variation in small-scale societies [39,40] and future research can look to incorporate more diverse musical styles and participant groups.

Automatic mood detection algorithms in MIR are generally trained on English songs, with the ratings obtained from Western annotators. Given this circumstance, we predicted that the algorithm would be culturally biased and align better with the US raters. However, counter to our intuition, we found no cultural bias in the algorithms. All four mood features in our comparison (*danceable*, *energy*, *sad*, and *electronic*) aligned similarly well with raters from the three countries. Given these outcomes, we conclude that current mood detection algorithms are good objective proxies for human judgments and are not culturally biased, at least within the popular music context in industrialized societies.

6. DATA AVAILABILITY

Raw participant ratings, musical stimuli used in the experiment, and additional statistical results are available at <https://osf.io/3uw9d/>

7. ACKNOWLEDGEMENT

We greatly appreciate the support we received from Fernanda Fernandes and Seunghyun Jeong for the recruitment of participants in Brazil and South Korea. We would also like to thank the four anonymous reviewers who provided constructive feedback. This work was funded by the MPI for Human Cognitive and Brain Science (studentship awarded to HL) and MPI for Empirical Aesthetics (NJ).

8. REFERENCES

- [1] M. Park, J. Thom, S. Mennicken, H. Cramer, and M. Macy, “Global music streaming data reveal diurnal and seasonal patterns of affective preference”, *Nature Human Behaviour*, vol. 3, no. 3, Art. no. 3, Mar. 2019.
- [2] T. Eerola and J. Vuoskoski, “A comparison of the discrete and dimensional models of emotion in music”, *Psychology of Music*, Jan. 2011.
- [3] P. E. Savage, S. Brown, E. Sakai, and T. E. Currie, “Statistical universals reveal the structures and functions of human music”, *PNAS*, vol. 112, no. 29, pp. 8987–8992, Jul. 2015.
- [4] S. A. Mehr *et al.*, “Universality and diversity in human song”, *Science*, vol. 366, no. 6468, Nov. 2019.
- [5] T. Eerola, “Modeling emotions in music: Advances in conceptual, contextual and validity issues”, presented at the *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, Jan. 2014.
- [6] J. A. Russell, “A circumplex model of affect”, *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.
- [7] L. L. Balkwill and W. F. Thompson, “A cross-cultural investigation of the perception of emotion in music: psychophysical and cultural cues”, *Music Perception: An Interdisciplinary Journal*, vol. 17, no. 1, pp. 43–64, 1999.
- [8] T. Fritz *et al.*, “Universal recognition of three basic emotions in music”, *Current Biology*, vol. 19, no. 7, pp. 573–576, Apr. 2009.
- [9] M. Zentner, D. Grandjean, and K. R. Scherer, “Emotions evoked by the sound of music: Characterization, classification, and measurement”, *Emotion*, vol. 8, no. 4, pp. 494–521, 2008.
- [10] A. S. Cowen, X. Fang, D. Sauter, and D. Keltner, “What music makes us feel: At least 13 dimensions organize subjective experiences associated with music across different cultures”, *PNAS*, vol. 117, no. 4, pp. 1924–1934, Jan. 2020.
- [11] J. Henrich, S. J. Heine, and A. Norenzayan, “The weirdest people in the world?”, *Behavioral and Brain Sciences*, vol. 33, no. 2–3, pp. 61–83, Jun. 2010.
- [12] X. Hu and J. H. Lee, “A cross-cultural study of music mood perception between American and Chinese listeners.”, in *Proc. of the 13th Int. Society for Music Information Retrieval Conf.*, Porto, Portugal, Oct. 2012, pp. 535–540.
- [13] X. Hu, J. H. Lee, K. Choi, and J. S. Downie, “A cross-cultural study on the mood of K-POP songs”, in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, Taipei, Taiwan, Oct. 2014, pp. 385–390.
- [14] K. Kosta, Y. Song, G. Fazekas, and M. B. Sandler, “A study of cultural dependence of perceived mood in Greek music.”, in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, Curitiba, Brazil, Nov. 2013, pp. 317–322.
- [15] A. Singhi and D. G. Brown, “On cultural, textual and experiential aspects of music mood.”, in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, Taipei, Taiwan, Oct. 2014, pp. 3–8.
- [16] B. Thompson, S. G. Roberts, and G. Lupyan, “Cultural influences on word meanings revealed through large-scale semantic alignment”, *Nature Human Behaviour*, vol. 4, no. 10, Art. no. 10, Oct. 2020.
- [17] J. C. Jackson *et al.*, “Emotion semantics show both cultural variation and universal structure”, *Science*, vol. 366, no. 6472, pp. 1517–1522, Dec. 2019.
- [18] D. S. Cheng, T. Joachims, and D. Turnbull, “Exploring acoustic similarity for novel music recommendation”, in *Proc. of the 21st Int. Society for Music Information Retrieval Conf. Virtual Conference*, Oct. 2020, pp. 583–589.
- [19] M. Interiano, K. Kazemi, L. Wang, J. Yang, Z. Yu, and N. L. Komarova, “Musical trends and predictability of success in contemporary songs in and out of the top charts”, *Royal Society Open Science*, vol. 5, no. 5, p. 171274, 2018.
- [20] E. Zangerle, M. Vötter, R. Huber, and Y.-H. Yang, “Hit song prediction: Leveraging low- and high-level audio features”, in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, Delft, The Netherlands, Nov. 2019, pp. 319–326.
- [21] M. Mauch, R. M. MacCallum, M. Levy, and A. M. Leroi, “The evolution of popular music: USA 1960–2010”, *Royal Society Open Science*, vol. 2, no. 5, p. 150081, 2015.
- [22] J. A. Burgoyne, J. Wild, and I. Fujinaga, “An expert ground truth set for audio chord recognition and music analysis.”, in *Proc. of the 12th Int. Society for Music Information Retrieval Conf.*, Miami, United States, Oct. 2011, pp. 633–638.

- [23] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, “The Million Song dataset”, in *Proc. of the 12th Int. Society for Music Information Retrieval Conf.*, Oct. 2011, pp. 591-596.
- [24] S. F. Way, J. Garcia-Gathright, and H. Cramer, “Local trends in global music streaming”, in *Proc. of the Int. AAAI Conf. on Web and Social Media*, May 2020, pp. 705–714.
- [25] D. Bogdanov *et al.*, “Essentia: An audio analysis library for music information retrieval.”, in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, Curitiba, Brazil, Nov. 2013, pp. 493–498.
- [26] C. C. S. Liem and C. Mostert, “Can’t trust the feeling? How open data reveals unexpected behaviour of high-level music descriptors” in *Proc. of the 21st Int. Society for Music Information Retrieval Conf. Virtual Conference*, Oct. 2020, pp. 240-247.
- [27] K. Lemhöfer and M. Broersma, “Introducing LexTALE: A quick and valid lexical test for advanced learners of English”, *Behavioural Research Methods*, vol. 44, no. 2, pp. 325–343, 2012.
- [28] K. J. P. Woods, M. H. Siegel, J. Traer, and J. H. McDermott, “Headphone screening to facilitate web-based auditory experiments”, *Attention, Perception, & Psychophysics*, vol. 79, no. 7, pp. 2064–2072, Oct. 2017.
- [29] P. Harrison *et al.*, “Gibbs Sampling with People”, *Advances in Neural Information Processing Systems*, vol. 33, pp. 10659–10671, 2020.
- [30] F. van de Vijver and N. K. Tanzer, “Bias and equivalence in cross-cultural assessment: An overview”, *European Review of Applied Psychology*, vol. 54, no. 2, pp. 119–135, 2004.
- [31] E. Schubert, “The influence of emotion, locus of emotion and familiarity upon preference in music”, *Psychology of Music*, vol. 35, no. 3, pp. 499–515, Jul. 2007.
- [32] K. O. McGraw and S. P. Wong, “Forming inferences about some intraclass correlation coefficients”, *Psychological Methods*, vol. 1, no. 1, pp. 30–46, 1996.
- [33] C. Spearman, “The proof and measurement of association between two things”, *The American Journal of Psychology*, vol. 15, no. 1, pp. 72–101, 1904.
- [34] B. Diedenhofen and J. Musch, “cocor: A comprehensive solution for the statistical comparison of correlations”, *PLOS ONE*, vol. 10, no. 4, p. e0121945, Apr. 2015.
- [35] K. Jakubowski and A. Ghosh, “Music-evoked autobiographical memories in everyday life”, *Psychology of Music*, Dec. 2019.
- [36] S. Chowdhury, A. V. Portabella, V. Haunschmid, and G. Widmer, “Towards explainable music emotion recognition: The route via mid-level features”, in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, Delft, The Netherlands, Nov. 2019, pp. 237–243.
- [37] H. Xue, L. Xue, and F. Su, “Multimodal music mood classification by fusion of audio and lyrics”, in *MultiMedia Modeling*, Cham, 2015, pp. 26–37.
- [38] J. S. Gómez-Cañón, E. Cano, P. Herrera, and E. Gómez, “Joyful for you and tender for us: the influence of individual characteristics and language on emotion labelling and classification” in *Proc. of the 21st Int. Society for Music Information Retrieval Conf. Virtual Conference*, Oct. 2020, pp. 853-860.
- [39] J. H. McDermott, A. F. Schultz, E. A. Undurraga, and R. A. Godoy, “Indifference to dissonance in native Amazonians reveals cultural variation in music perception”, *Nature*, vol. 535, no. 7613, pp. 547–550, Jul. 2016.
- [40] N. Jacoby, E. A. Undurraga, M. J. McPherson, J. Valdés, T. Ossandón, and J. H. McDermott, “Universal and non-universal features of musical pitch perception revealed by singing”, *Current Biology*, vol. 29, no. 19, pp. 3229-3243.e12, Oct. 2019.

RECONSIDERING QUANTIZATION IN MIR

Jordan Lenchitz

College of Music, Florida State University
jlenchitz@fsu.edu

ABSTRACT

This paper presents a critique of the ubiquity of boilerplate quantizations in MIR research relative to the paucity of engagement with their methodological implications. The wide-ranging consequences of reflexivity on the future of scholarly inquiry combined with the near-universal contemporary recognition of the need to broaden the scope of MIR research invite and merit critical attention. To that end, focusing primarily on twelve-tone equal-tempered pitch and dyadic rhythm models, we explore the practical, cultural, perceptual, historical, and epistemological consequences of these pervasive quantizations. We analyze several case studies of meaningful and successful past research that balanced practicality with methodological validity in order to posit several best practices for both future intercultural studies and research centered on more narrowly constructed corpora. We conclude with a discussion of the dangers of solutionism on the one hand and the self-fulfilling prophecies of status quoism on the other as well as an emphasis on the need for intellectual honesty in metatheoretical discourse.

1. INTRODUCTION

Fostering cultural diversity in MIR research is not merely a question of “adapting” existing methodologies developed on the basis of certain *a priori* assumptions to repertoires or tasks that perhaps challenge said assumptions. In particular, it should also entail critical reflection on the benefits and drawbacks of those assumptions in studies of all repertoires, even (if not especially) those on the basis of which such assumptions were made in the first place. Twelve-tone equal-tempered quantizations of pitch and dyadic quantizations of rhythm represent arguably the most ubiquitous such assumptions in contemporary MIR research, and yet despite their prevalence their foundational role underlying many diverse methodological approaches all too often passes unannounced. This paper therefore presents an analysis not of any musical information in particular but rather of the apparatuses we use to retrieve it from musics in the hopes of fostering productive future methodological conversations.

Quantization may be defined as the organization of dimensional information into discrete sets of values (otherwise known as categorization in perception science), and it is cognitively essential for creating music. That being said, the centrality of the process of quantization to music

does not justify the perpetuation of reliance on any “default” quantizations. Indeed, certain quantizations are ubiquitous, all-too-often unstated *a priori* assumptions underlying a substantial majority of MIR methodologies. Are they compromises? Almost invariably yes they are. Do they represent pragmatic choices? Quite possibly they do, depending on the context. But pragmatic compromises or otherwise, the foundational position of these quantizations endows them with consequences that merit consideration in the context of any methodological decision-making and especially if meaningful progress is to be made in expanding the purview of MIR and its applications.

This paper is structured as follows: after this introduction providing the rationale for a reconsideration of quantization in MIR, its consequences practical, cultural, perceptual, historical, and epistemological are each explored in turn with an emphasis on pitch and rhythm in the second section. The third section provides illustrative case studies that demonstrate strategies for balancing pragmatism with methodological validity and posits best practices for both intercultural and intracultural research. The fourth and final section argues for the necessity of avoiding both solutionism on the one hand and status quoism on the other and emphasizes the importance of intellectual honesty in metatheoretical discourse.

2. CONSEQUENCES OF QUANTIZATION

2.1 Practical Consequences

The most obvious practical consequence of the ubiquity of certain quantizations in MIR is the widespread availability of platforms built around them and the corresponding noticeable absence of alternatives. In and of itself this is unsurprising and not necessarily a drawback; and yet, it can bring about the existence of unfortunate self-fulfilling prophecies. Consider the case of the justifiably popular Python audio analysis package *librosa* [1], in which one may specify the number of notes per octave but the assumption that they are equally spaced is not so easily changed. As maintainer Brian McFee put it with respect to pitch on the Music Information Retrievers Slack in July 2020, equal temperament “is a compromise, but one I’m willing to live with for the time being; extending to support just intonation in a fully consistent way would be a huge undertaking, much bigger than just adding notation support.” We are inclined to agree with McFee in his assessment that implementing meaningful support for tuning systems other than equal temperaments would be a very nontrivial task. At the same time, however, such tasks tend to be welcomed by MIR researchers as motivation for innovation. Why, then, does this remain unaddressed?



© J. Lenchitz. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Lenchitz, “Reconsidering Quantization in MIR”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

Imagine a hypothetical MIR researcher who desires to do work on musics in non-equal-tempered pitch systems. Methodological convenience is seldom acknowledged as a motivating factor in the planning and implementation of research plans and yet psychologically it has an effect on what individuals decide to do and not to do. The existence vs. nonexistence of tools structured so tightly around certain pitch quantizations, in turn, has implications for such convenience or lack thereof. Adding the need to produce a novel notation and/or data format on top of the already major task of encoding a new corpus in a machine-readable format as prerequisites for exploring avenues for actual information retrieval can make the scale of a prospective project quickly balloon in size and therefore be discouraging to its actual execution. And even for projects that do make it to completion, there remain the potential for obstacles in the peer-review process. Excessive specialization brings about the possibility for manuscripts to be reviewed by homogeneous niches whose shared assumptions may prevent work from being disseminated to the broader community and from being built upon by subsequent endeavors, and such dynamics of review admit no simple fixes.

Beyond non-symbolic tools such as *librosa*, it additionally bears mentioning that symbolic tools have also embraced the compromise of equal-tempered quantization of pitch at just about every level of their functionality. To name just a few, *Humdrum* [2] and *music21* [3] are equally “not fully compatible” [4] with musics which do not consist of twelve pitches logarithmically equally spaced over 2:1 octaves in terms of pitch structure. These tools are all of course excellent in countless use cases and this is not by any means to suggest they be abandoned. Rather, this acknowledgement of their shared assumption with respect to the modeling of pitch leads to a fuller and more useful understanding of their limitations in considering what one might choose in terms of platform for a study if this exact assumption is not desired to be made for any given research plan. In particular, the pitch quantizations inherent in the notation encodings upon which such symbolic tools rely could be taken to motivate non-symbolic approaches in cases where one’s target musics are not served as well by these encodings’ inherent assumptions.

2.2 Cultural Consequences

Anthropologists, ethnomusicologists, linguists and other scholars distinguish between emic and etic understandings of culturally-specific phenomenon, which can be approximately understood as insider and outsider perspectives [5]. Quantizations in MIR are overwhelmingly based on etic views of culturally-specific musical phenomenon which harm the ecological validity of methodologies. Twelve-tone equal-tempered pitch quantization is regrettable in this regard because of the number of musical cultures in the world that use or have used more or fewer than twelve notes per octave (e.g. Indian *rāga*, Turkish *makam*, or Indonesian Gamelan musics), twelve non-equally-spaced notes per octave (e.g. much of the history of Western musics, as discussed further in subsection 2.4) or non-octave based pitch structures (e.g. tritave-based works by Wendy Carlos et al.). Tools that are ill-suited to handle the musical

realities of these and similar culturally-specific phenomenon lead either to their exclusion from consideration or (and arguably worse) facilitate their inclusion but in ways that do them a disservice.

Dyadic rhythm quantization has similarly negative consequences, especially in light of the differing approaches to meaning and teleology across musical cultures. Micro-timings pose a practical challenge in any case but depending on the culture in question, such minute expressive discrepancies from most quantizations could serve as the primary determinants of culturally-specific meaning. And with respect to teleology, cyclic vs. linear conceptions of rhythm and time ought to factor in to how and why we decide to treat rhythm in both symbolic and signal processing-based approaches (e.g. because structural repetitions might rely on subtle changes near the beginnings or ends of units of repetition part and parcel with large-scale formal processes). Yet by and large these implications are either swept under the rug or taken to mean that work will instead focus elsewhere.

At the beginning of this paper we alluded to the fact that fostering cultural diversity in MIR is not merely a matter of “adapting” existing methodologies to a broader variety of musics. The cultural consequences of quantization highlight this fact in that it would be quite simple if not trivial to apply say a twelve-tone chroma feature to Turkish *makam* or a dyadic rhythm quantization to Burundi Whisped Inanga but any results so obtained could be meaningless to practitioners of those musical cultures. It is not enough, furthermore, to involve emic perspectives in a research plan if its computational approaches do not also take them into account. Consideration of not only the primary audiences of MIR research but also all of the stakeholders in its endeavors leads to the realization that the deeper issues of cultural diversity thus lie in methodology as well as in repertoire.

2.3 Perceptual Consequences

Many if not all MIR methodologies attempt to relate to humans’ perceptions of musics in some way in order to yield results relevant to the experiences of listeners. And yet, the quantizations of pitch and rhythm that underlie many of these methodologies exhibit to a nontrivial extent an arbitrariness divorced from perceptual realities. In the domain of pitch, for instance, the number of notes in the twelve-tone equal-tempered scale from 16 Hz to 16 kHz is only 120 whereas the number of perceptible pitch steps in the same range is approximately 1400 [6]. If a methodology sought to explore perceptual quantization among individuals with absolute pitch and socialized in musical cultures featuring twelve notes per octave, the decision to limit its pitch quantization accordingly would follow naturally. Most methodologies, however, do not intend to ask and attempt to answer such questions and yet they employ this quantization of pitch all the same.

With respect to rhythm quantization, even SOTA or near-SOTA results similarly bring caveats in terms of their perceptual implications. Automatic transcription does not claim to be a representation of listeners’ experiences of the music in question, and yet its use in analyzing performed divergences from symbolic rhythmic notation attests to a

modeling of performers' conscious or subconscious decision-making in some sense. Automatic transcriptions of performances by the legendary Canadian musician Glenn Gould, for instance, produce rational approximations of his keyboard ornamentation necessary to fit into the prevailing dyadic rhythmic grid [7]. Such results are valid in the sense that they more or less closely match the raw interonset interval (IOI) data up to the capability of the algorithms in question. Still, from a perceptual perspective, the idea that either listeners or performers are themselves quantizing the dizzying diversity of IOIs present into some similarly messy rational dyadic mental representation is rather unfortunate.

This is not to say that MIR models that do not claim to be perceptual models should be rejected on perceptual grounds. On the contrary, such models should be considered in terms of their claims and their intentions. At the same time, however, recognition of the perceptual consequences of quantization in the modeling of musics could easily lead to more reflexivity in scholarly discourse, the absence of which seems to reflect an unspoken consensus. Consensus is one straightforward means of making tangible progress on solving difficult problems but when it is merely implicit and unspoken it is less likely to serve as a solid foundation for such progress. The perceptual implications of quantizations in MIR therefore would do well to figure more in scholarly discourse both for the sake of increasing methodological validity and also to promote engagement with the state of the art in the cognitive sciences—aims which have indeed been documented for as long as ISMIR has existed [8].

2.4 Historical Consequences

If we narrow our scope temporarily to studies of Western European classical musics, we find that common quantizations of pitch and rhythm have striking consequences with respect to historical ecological validity. Revisionist histories of tuning do not change the historical reality that twelve-tone equal temperament is a relatively recent phenomenon in practice. In terms of repertoire, one example par excellence is J. S. Bach's *The Well-Tempered Clavier* (WTC), a collection of keyboard works whose very title explicitly testifies to the composer's intention to explore all 24 unequal major and minor keys which were usable compared to meantone and yet had different affective profiles due to minor but nontrivial intervallic size differences. Any information retrieved from a symbolic representation of the WTC imposing the enharmonic equivalence and logarithmically equal note spacing of twelve-tone equal temperament on its pitch content would therefore do a disservice to the historical circumstance surrounding the creation of the work and divorce results obtained from its meaning at the time of its creation.

If we look more recently in music history, there do exist plenty of corpora for which a twelve-tone equal-tempered pitch model is quite appropriate. Composers working since the advent of twelve-tone equal-tempered tuning who rely on this pitch logic in the structure of their musics are a natural fit for this particular quantization, as are the generations of popular musicians who inherited this structure more or less wholesale. In a similar vein to what Cella [9]

has observed in the case of contemporary classical music, one possible explanation for the application of this quantization to musics earlier than for which it is best-suited may lie in the 'follow the money' reality of many MIR projects. That is, given that the largest audiences today are served by tools centered around the quantization of pitch most common in modern popular musics, one would not be surprised for there to be less investment in tools which would best handle earlier pitch quantizations. If dynamics of pitch quantization were fully considered before planning and executing research on historical repertoires, we might study some understudied repertoires more and overstudied repertoires less.

Rhythm quantization, in turn, is analogously problematic from the perspective of impacting historical ecological validity. The case of French Baroque *notes inégales* may serve as a representative example. In this case the widespread divergence of performance practice from symbolic representations of the music leads to situations where MIR studies based on symbolic data can make valid claims about the symbolic data that are nonetheless not reflective of the music as performed by historically-informed practitioners and would therefore not align with non-symbolic studies of the same music. Moreover, rhythm quantization in cases of performance practice poses a much less daunting (though still nontrivial) challenge for well-intentioned researchers than pitch. Whether or not something is challenging is seldom the sole determining factor in the processes of methodological development, and yet the relative ease of addressing *notes inégales* suggests that the broader trend at play here is a minimization of concern for historical realities in the implementation of such studies rather than practicality.

2.5 Epistemological Consequences

At once the most concerning and least-addressed drawback of the omnipresence of certain quantizations in MIR is their tendency to foster confirmation bias. If our hypotheses are to withstand critical scrutiny, they must be not only falsifiable but also tested in such a way that the results are not preordained by our methodological decision-making. A useful lesson can be drawn from the example of studies of cognitive and academic benefits of music training in children, where a recent meta-analysis suggests that confirmation bias might well have influenced the validity of nearly four decades of results [10]. The lesson to learn from these experimental studies is that when research is designed with the expectation to find a certain result, it should not be surprising nor necessarily meaningful to produce that result.

For example, the use of twelve-tone equal-tempered chroma features in signal-based analysis can quite possibly produce statistically significant results with respect to those bins. This does not, however, necessarily tell us anything meaningful about the underlying signal because such bins are merely rounding the chroma information actually present to the nearest 100 cents. The same principle applies to dyadic rhythm quantization of IOI information, where the results obtained can be elegant and convincing with respect to this quantization but again do not neces-

sarily reveal a deeper truth about the signal in question depending on the nature of its unquantized IOIs. Multidimensional attempts to quantize timbre, in turn, have had success in synthesis applications [11] that should not however be interpreted as a guarantee that the same approaches will lead to ecologically valid results in analysis.

Another epistemological drawback of such quantizations is their tendency to foster selection bias. As discussed earlier in the context of practical consequences, when mainstream tools exist that are suited for certain musics and not others, it should not come as a surprise that a significant number of researchers tend to apply those tools to those musics at the expense of others. Given the present recognition of the need to counter sampling bias in MIR research, all factors that contribute to it accordingly deserve acknowledgment and discussion. Methodologically careful intercultural studies as well as intracultural research have been performed and continue to be performed by members of the MIR community, and this is not to neglect their contributions (which will be addressed in the following section). Rather, to understand the role of these quantizations in contributing to selection bias is to admit that despite their ubiquity, they are far from “neutral” positions one might assume them to be.

3. CASE STUDIES AND BEST PRACTICES

If the preceding critiques of boilerplate quantizations have perhaps presented a morose picture of the state of contemporary MIR, the following case studies and recommended best practices should serve as a more optimistic change of pace. There are undoubtedly other case studies we have omitted that would serve just as well as examples and other best practices we do not recommend here. This section is therefore best understood as an attempt to highlight directions for future work to complement the aforementioned discussions of the consequences of “default” quantizations.

3.1 Case Studies

Although it is rather uncommon, theoretical work does exist that has explored pitch with an eye toward information retrieval through continuous rather than quantized perspectives. Callender [12] has investigated continuous harmonic spaces using a Fourier-based approach to symbolic data that enables the application of “harmonic intuitions to all possible chords of pitches and pitch classes in all possible tuning systems.” Wakefield [13] has examined the mathematical and computational implications of “joint time-chroma distributions” that could be used to produce unquantized chroma features on the signal-processing side. Neither of these theoretical contributions has been the subject of much follow-up work which is regrettable considering that they suggest possibilities for methodologies approaching the extraction of pitch information without assuming any particular quantization of pitch in advance. At the same time, that continuous alternatives to the received wisdom of certain quantizations have largely passed unnoticed in MIR is to an extent to be expected since those most committed to working with specific sets

of discrete values are not incentivized to explore options beyond them.

Approaches to quantization starting either from a continuous perspective or closer to it also merit mention here. Moelants, Cornelis, and Leman [14] implemented a methodology in which “pitch is first analyzed on a continuous scale” and “peak analysis is then applied on these data to extract the actual scale used.” Among the advantages of this approach are the fact that it is applicable to many signal-based MIR methods and that it is easily generalizable across repertoires. Six and Cornelis [15] used a granular “resolution of 1200 cents” to cover more than the pitches of twelve-tone equal temperament in order to “form musically meaningful representations” of non-Western musical traditions. In both of these cases the methodology was planned and implemented with maximal ecological validity as one desired outcome. Whether starting with continuous data or relying on a granular quantization for the sake of more meaningful analysis, both of these examples demonstrate that it is not only possible but quite doable to make such decisions in one’s own research agenda when the musical situations at hand call for it.

Two differing approaches to working with histogram bins can serve as worthwhile examples of quantizations not starting from continuous perspectives but still centering culturally-specific knowledge in their computational implementations. Panteli [16] in a comparative study of Cypriot pitch patterns increased their histogram resolution by a factor of three compared to the octave partitions specified in Byzantine and Turkish theoretical sources “for better precisions and tuning robustness” while remaining tied to the traditions’ emic perspectives. Bozkurt [17] found in the analysis of traditional *makam* music in Turkey performed by a living master that the histogram matched neither twelve-tone equal temperament nor the official standard tuning system of the music as codified in print sources and accordingly designed a tuning application based on such recordings rather than any frequency presets. Working within some discrete universe to start but modifying it to suit the investments of multiple stakeholders can balance tensions inherent in intercultural work.

One research project that serves as the umbrella for many approaches to quantization worthy of emulation is “CompMusic: Computational Models for the discovery of the world’s music” [18] coordinated by Serra. Recognizing the drawbacks of the hegemony of Western-centered paradigms in MIR, this project not only includes members from each of the cultures being studied but also targets practitioners of these specific traditions in its development of interactive systems. Another example of an organizational umbrella fostering potential departures from common quantization norms was the Music Encoding Initiative (MEI) and their MEI Incubator [19], which gave practitioners a “common space to ‘grow’ their customizations and share them with other members of the community.” Both of these case studies are also helpful reminders of the fact that much of the work needing to be done to overcome limitations of certain quantizations lies in the domain of encoding and formatting corpora for processing.

Lastly, two more recent examples involving the study of Indian art music (IAM) are further demonstrative of other possibilities for sensibly handling quantization in a

way most germane to the repertoire in question. The methodology of Ranjani et al. [20] involved “non-uniform quantization intervals...selected from pitch and time scales prevalent in IAM [and] accommodating pitch and inter-note-interval variations on [a] pitch-time grid,” accounting for the repertoire in the methodology rather than the other way around. Viraraghavan et al. [21], in turn, proposed a transcription methodology involving a Viterbi algorithm that outperformed uniform quantization (which would be the default in MIR of Western musics) in terms of adherence to *rāga*. The commonalities between these two contributions accordingly exceed their corpus and their aims in that both made conscious methodological decisions that decentered assumptions made in the study of Western musics in accordance with the questions they sought to ask and answer.

3.2 Best Practices

There are several potential best practices to be drawn from the brief survey of case studies presented above. Chiefly among these is the recognition of the continuous reality of pitch as a prerequisite for an implementation of its quantization. Our position is not that MIR researchers believe as Vincenzo Galilei did that pitch is discrete nor that all quantizations are unconscious reflexes but rather that intentionality is key in these methodological underpinnings. Continuous models used to obtain discrete ones (as in [14]) are applicable to any musics and increased granularity (as in [15]) need not be limited to non-Western repertoires. Discrete models informed by the theoretical and performance output of practitioners within the cultures in question [16-17] can foster increased usability of results beyond scholarly applications. Especially for the study of expressive intonation, experimental and microtonal musics, and signal-processing based on performance rather than symbolic analysis, these approaches with or without the orientation of explicitly continuous thinking at all stages of execution [12-13] are likely to increase ecological validity and offer pathways into understudied repertoires.

The institutional case studies [18-19], in turn, highlight the utility of large-scale cooperation as well as the potential impact of targeted strategic planning. By aiming big and serving both as a proof of concept and a timely contribution to the cutting-edge, CompMusic demonstrates the potential for reverse-engineering research structure from concrete culturally-specific goals. And though more open-ended than CompMusic and focusing on symbolic rather than signal representations, the MEI Incubator’s acknowledgement of the importance of explicitly carving space for digging into underexplored territory while minimizing duplication of labor can be taken as another replicable best practice. The deliberate emphasis of both of these initiatives on maximal dissemination of their output reinforces the importance of democratization of knowledge for empowering practitioners who may not yet realize they have options beyond those they might consider by default.

Finally, the case studies involving IAM [20-21] suggest that repertoire driving methodology and not the converse is a successful means for optimizing the ecological validity of research. Another best practice they suggest is the incorporation of emic knowledge of diverse musical cultures

in the research process. One need not be a practitioner of any particular musical repertoire in order to be able to access pertinent ethnomusicological research on it. On the contrary, the majority of such scholarship is written by experts who understand that they have firsthand experience with a given repertoire or culture but their audience does not. MIR as a whole has been generally good about interfacing [22] with the cognitive sciences, the library sciences, and the computational disciplines pertinent to what it does but has a spottier track record historically with respect to engaging with relevant ethnomusicology beyond studies of recordings and transcriptions [23]. In our opinion, remedying this would represent an additional best practice for future work.

4. RECOMMENDATIONS

4.1 Against Solutionism

Our first recommendation is for MIR to avoid the solutionist impulse it might feel in response to critiques of certain quantizations. Indeed, attempting to quickly address perceived drawbacks of methodologies without fully probing the nature of the issues at hand “is likely to have unexpected consequences that could eventually cause more damage than the problems they seek to address” [24]. We have deliberately refrained from positing any alleged panacea to the prevalence of these quantizations firstly because none exist but more importantly because that would be antithetical to our broader aim of fostering a conversation involving as many stakeholders with differing perspectives as possible. If we are to make meaningful structural (i. e. rather than cosmetic) progress as a discipline with respect to diversity and inclusion, we need to be having substantive conversations about what we seek to change and why we seek to change it before we dive into well-intentioned attempts to implement such changes.

4.2 Against Status Quoism

Our second recommendation is for MIR to reject the inertia that has enable these quantizations to remain so central and unchallenged to its methodologies for so long. Questioning received wisdom is necessary and crucial to the long-term success of any research enterprise, and the ubiquity and utility of any quantization should not exempt it from critique. In many ways status quoism is the dual of solutionism, and to reject one while embracing the other would be hypocritical to say the least. Practically and institutionally speaking, we recognize that status quoism is a stance rewarded by the mechanisms of publication, recognition, career advancement, etc. in MIR and adjacent disciplines. It therefore merits conscious effort our on part as researchers to actively and explicitly posit the need for change while simultaneously acknowledging that changes must be thoughtfully and meticulously planned before they are implemented if they are to last.

4.3 Intellectual Honesty in Metatheoretical Discourse

Our third and final recommendation is for increased intellectual honesty in metatheoretical discourse. It is easy to perceive critiques of methodologies one uses or has used

as critiques of one’s entire research agenda and this can only have negative consequences. By displacing positive emotional investment in one’s research into negative responses to methodological criticism, productive conversations often are extinguished before they can begin and shouting into the void on all sides can be encouraged. We therefore call above all for dialogue, open-mindedness, and lucidity in terms of understanding the potential for a meaningful research agenda to be built on assumptions that one may not have previously questioned and for improvements to be made possible by exploring those assumptions. Reflection and even more importantly self-reflection are crucial to the present and future of inquiry, and quantization just as much as anything else ought to be the subject of such consideration.

5. ACKNOWLEDGEMENTS

The author is supported by a Legacy Fellowship from Florida State University and would like to thank the anonymous reviewers for their invaluable feedback on the paper.

6. REFERENCES

- [1] B. McFee et al., “librosa: Audio and music signal analysis in python,” in *Proc. of the 14th Python in Science Conf.*, Austin, TX, 2015, pp. 18–25.
- [2] D. Huron. The Humdrum Toolkit: Software for music researchers. Software package, 1993.
- [3] M. Cuthbert and C. Ariza, “music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data,” in *Proc. of the 11th Int. Society for Music Information Retrieval Conf.*, Utrecht, Netherlands, 2010, pp. 637–642.
- [4] A. Blake, “Computational analysis of quarter-tone compositions by Charles Ives and Ivan Wyschnegradsky,” M.A. thesis, College of the Arts, Kent State Univ., Kent, Ohio, 2020. Accessed on: July 26, 2021. [Online]. Available: http://rave.ohiolink.edu/etdc/view?acc_num=kent1588259095079479
- [5] M. Morris et al., “Views from inside and outside: integrating emic and etic culture and justice judgement,” *Academy of Management Review*, vol. 24, no. 4, pp. 781–796, October 1999.
- [6] H. Olson, *Music, Physics and Engineering*. New York: Dover Publications, 1967.
- [7] G. Boenn, “Rhythm quantization,” in *Computational Models of Rhythm and Meter*. Cham, Switzerland: Springer, 2018, pp. 147–173.
- [8] D. Huron, “Perceptual and cognitive applications in music information retrieval,” in *Proc. of the 1st Int. Society for Music Information Retrieval Conf.*, Plymouth, MA, 2000, pp. 83–92.
- [9] C.-E. Cella, “Music Information Retrieval and Contemporary Classical Music: A Successful Failure,” *Trans. of the Int. Society for Music Information Retrieval*, vol. 3, no. 1, pp. 126–136, September 2020.
- [10] G. Sala and F. Gobet, “Cognitive and academic benefits of music training with children: A multi-level meta-analysis,” *Memory & Cognition*, vol. 48, pp. 1429–1441, July 2020.
- [11] A. Bitton, P. Esling, and T. Harada, “Vector-quantized timbre representation,” 2007, *arXiv:2007.06349*.
- [12] C. Callender, “Continuous harmonic spaces,” *J. Music Theory*, vol. 51, no. 2, pp. 277–332, Fall 2007.
- [13] G. Wakefield, “Mathematical representation of joint time-chroma distributions,” in *Proc. of the SPIE Conf. on Advanced Signal Processing Algorithms, Architectures, and Implementations IX*, Denver, CO, 1999, pp. 637–645.
- [14] D. Moelants, O. Cornelis, and M. Leman, “Exploring African tone scales,” in *Proc. of the 10th Int. Society for Music Information Retrieval Conf.*, Kobe, Japan, 2009, pp. 489–494.
- [15] J. Six and O. Cornelis, “Tarsos – A platform to explore pitch scales in non-Western and Western music,” in *Proc. of the 12th Int. Society for Music Information Retrieval Conf.*, Miami, FL, 2011, pp. 169–174.
- [16] M. Panteli, “Pitch Patterns of Cypriot Folk Music between Byzantine and Ottoman Influence,” M. S. thesis, Dept. of Information and Communication Technologies, Univ. Pompeu Fabra, Barcelona, Spain, 2011. Accessed on: July 26, 2021. [Online]. Available: <http://mtg.upf.edu/node/2333>
- [17] B. Bozkurt, “A System for Tuning Instruments Using Recorded Music Instead of Theory-Based Frequency Presets,” *Computer Music Journal*, vol. 36, no. 3, pp. 43–56, Fall 2012.
- [18] X. Serra, “A Multicultural Approach in Music Information Research,” in *Proc. of the 12th Int. Society for Music Information Retrieval Conf.*, Miami, FL, 2011, pp. 151–156.
- [19] A. Hankinson, P. Roland, and I. Fujinaga, “The Music Encoding Initiative as a Document-Encoding Framework,” in *Proc. of the 12th Int. Society for Music Information Retrieval Conf.*, Miami, FL, 2011, pp. 293–298.
- [20] H. G. Ranjani et al., “A compact pitch and time representation for melodic contours in Indian art music,” *J. Acoustical Society of America*, vol. 145, no. 1, pp. 597–603, January 2019.
- [21] V. S. Viraraghavan et al., “State-based transcription of components of Carnatic music,” in *Proc. of the 45th Int. Conf. on Acoustics, Speech, and Signal Processing*, Online, 2020, pp. 811–815.
- [22] W. de Haas and F. Wiering, “Hooked on Music Information Retrieval,” *Empirical Musicology Review*, vol. 5, no. 4, pp. 176–185, October 2010.

- [23] A. C. Gedik and A. Holzappel, “The Meaning of Music in Ethnomusicology and Music Information Retrieval: Obstacles Against Computational Ethnomusicology,” presented at the Conference on Interdisciplinary Musicology–CIM18, Poznań, Poland, 2018.
- [24] E. Morozov, *To Save Everything, Click Here: The Folly of Technological Solutionism*. New York: PublicAffairs, 2013.

A UNIFIED MODEL FOR ZERO-SHOT MUSIC SOURCE SEPARATION, TRANSCRIPTION AND SYNTHESIS

Liwei Lin¹ Qiuqiang Kong² Junyan Jiang¹ Gus Xia¹

¹ Music X Lab, New York University Shanghai

² ByteDance, Shanghai, China

linliwei3916@gmail.com, kongqiuqiang@bytedance.com, jj2732@nyu.edu, gxia@nyu.edu

ABSTRACT

We propose a unified model for three inter-related tasks: 1) to *separate* individual sound sources from a mixed music audio, 2) to *transcribe* each sound source to MIDI notes, and 3) to *synthesize* new pieces based on the timbre of separated sources. The model is inspired by the fact that when humans listen to music, our minds can not only separate the sounds of different instruments, but also at the same time perceive high-level representations such as score and timbre. To mirror such capability computationally, we designed a pitch-timbre disentanglement module based on a popular encoder-decoder neural architecture for source separation. The key inductive biases are vector-quantization for pitch representation and pitch-transformation invariant for timbre representation. In addition, we adopted a query-by-example method to achieve *zero-shot* learning, i.e., the model is capable of doing source separation, transcription, and synthesis for *unseen* instruments. The current design focuses on audio mixtures of two monophonic instruments. Experimental results show that our model outperforms existing multi-task baselines, and the transcribed score serves as a powerful auxiliary for separation tasks.

1. INTRODUCTION

Music source separation (MSS) is a core problem in music information retrieval (MIR), which aims to separate individual sound sources, either instrumental or vocal, from a mixed music audio. A good separation benefits various of downstream tasks of music understanding and generation [1, 2] since many music-processing algorithms call for “clean” sound sources.

With the development of deep neural networks, we see significant performance improvements in MSS. The current mainstream methodology is to train on pre-defined music sources and then infer a mask on the spectrogram (or other data representations) of the mixed audio. More recently, we see several new efforts in MSS research, includ-

ing query-based method [3–6] for unseen (not pre-defined) sources, semantic-based separation that incorporates auxiliary information such as score or video [7–13], and multi-task settings [14].

This study conceptually combines the aforementioned new ideas but follows a very different methodology — instead of directly applying masks, *we regard MSS an audio pitch-timbre disentanglement and reconstruction problem*. Such strategy is inspired by the fact that when humans listen to music, our minds not only separate the sounds into different sources but also perceive high-level pitch and timbre representations that generalize well during both music understanding and creation. For example, humans can easily identify the same timbre in other pieces or identify the same piece played by other instruments. People can even mimic the learned timbre using human voice and sing (i.e., to synthesize via voice) the learned pitch sequence.

To mirror such capability computationally, we propose a zero-shot multi-task model jointly performing MSS, automatic music transcription (AMT), and synthesis. The model comprises four components: 1) a query-by-example (QBE) network, 2) a pitch-timbre disentanglement module, 3) a transcriber, and 4) an audio encoder-decoder network. First, the QBE network summarizes the clean query example audio (which contains only one instrument) into a low-dimensional query vector, conditioned on which the audio encoder extracts the latent representation of an individual sound source. Second, the model disentangles the latent representation into pitch and timbre vectors while transcribing the score using the transcriber. Finally, the audio decoder takes in both the disentangled pitch and timbre representations, generating a separated sound source. When the model further equips the timbre representation with a pitch-transformation invariance loss, the decoder becomes a synthesizer, capable of generating new sounds based on an existing timbre vector and new scores.

The current model focuses on audio mixtures of two monophonic instruments and performs in a frame-by-frame fashion. Also, it only transcribes pitch and duration information. We leave polyphonic and vocal scenarios as well as a more complete transcription for future work. In sum, our contributions are:

- **Zero-shot multi-task modeling:** To the best of our knowledge, it is the first model that jointly performs separation, transcription, and synthesis. It works



© Liwei Lin, Qiuqiang Kong, Junyan Jiang and Gus Xia. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Liwei Lin, Qiuqiang Kong, Junyan Jiang and Gus Xia, “A unified model for zero-shot music source separation, transcription and synthesis”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

for both previously seen and unseen sources using a query-based method.

- **Well-suited inductive bias:** The neural structure is analogous to the “hardware” of the model, which alone is inadequate to achieve good disentanglement. We designed two extra inductive biases: vector-quantization for pitch representation and pitch-transformation invariant for timbre representation, which serves as a critical part of the “software” of the model.
- **None-mask-based MSS:** Our methodology regards MSS an audio pitch-timbre disentanglement and re-creation problem, unifying music understanding and generation in a representation learning framework.

2. RELATED WORK

Most effective music source separation (MSS) methods are based on well-designed neural networks, such as U-Net [15] and MMDenseLSTM [16]. Here, we review three new trends of MSS related of our work: 1) multi-task learning, 2) zero-shot for unseen sources, and 3) taking advantage of auxiliary semantic information.

2.1 Multi-task Separation and Transcription

Several recent studies [14, 17, 18] conduct multi-task separation and transcription by learning a joint representation for both tasks. These works demonstrated that a multi-task setting benefits one or both of the two tasks due to the better generalized capability of the learned joint representation. Our model is a multi-task and can further disentangle pitch and timbre representation for sound synthesis.

2.2 Query-based Separation

Few-shot and zero-shot learning are becoming popular in MIR. For the MSS task, It is meaningful to separate unseen rather than pre-defined sources since it is unrealistic to collect training data that covers all the sources with considerable amounts. Query-by-example (QBE) network is one of solutions for zero-shot learning and recent researches [3, 4, 6, 19–22] show its nice performance. In this study, we adopt a QBE as in [3].

2.3 Semantic-based Separation

Many researches demonstrate that semantic information is a useful auxiliary for MSS. For example, Gover et al. [10] designs a score-informed wave-U-Net to separate choral music; Jeon et al. [23] performs the lyrics-informed separation; Meseguer-Brocal et al. [11] develops a phoneme-informed C-U-Net [24]; Zhao et al. [12] takes advantage of visual information to separate homogeneous instruments. But these methods cannot separate sources without additional semantic groundtruths during inference. Our study can also be regarded as score-informed MSS, but our model does not call for ground truth score during the inference time.

3. METHODOLOGY

In this section, we describe our proposed 1) multi-task and QBE model for source separation; 2) pitch-timbre disentanglement module; 3) pitch-translation invariance loss.

3.1 Multi-task Separation and Transcription

Different from previous works that tackle the music separation and music transcription problems separately, we learn a joint representation for both of them. Previous works [14, 17] have shown that the representation learnt by a joint separation and transcription task can generalize better than the representation learnt by single-task models.

We denote the waveform of two single-source audio segments from different sources as $s_c \in \mathbb{R}^L$ and $s_i \in \mathbb{R}^L$, respectively. We denote their mixture as:

$$x = s_c + s_i. \tag{1}$$

Our aim is to separate s_c from x . We denote the spectrogram of x and s_c as $\mathbf{X} \in \mathbb{R}^{T \times F}$ and $\mathbf{S}_c \in \mathbb{R}^{T \times F}$, respectively.

We first formalize the general MSS model using an encoder-decoder neural architecture. For instance, UNet [15] is an encoder-decoder architecture which is widely used in MSS. By ignoring the skip connections of UNet, the output of the encoder (the bottleneck of U-Net) can be used as a joint representation for separation and transcription. Different from previous MSS methods that estimate a single-target mask on the mixture spectrogram, we design the separation model to directly output spectrograms. In this way, the model can not only separate a source from a mixture, but can also synthesize new audio recordings from joint representations.

For the source separation system, we denote the encoder and decoder as follows:

$$\mathbf{h} = \text{Encoder}(\mathbf{X}), \tag{2}$$

$$\widehat{\mathbf{S}}_c = \text{Decoder}_c(\mathbf{h}), \tag{3}$$

where \mathbf{h} is the learned joint representation and Decoder_c is the decoder for the target source c . The joint representation \mathbf{h} is used as input to the a transcription model:

$$\widehat{\mathbf{y}}_c = \text{Transcriptor}_c(\mathbf{h}), \tag{4}$$

where $\widehat{\mathbf{y}}_c \in [0, 1]^{T \times N}$ are probabilities of the predicted MIDI roll. Typically, we set $N = 89$ including 88 notes on a piano and a silence state.

When designing neural networks, to remain the transcription resolution, we do not apply temporal pooling operation in the encoder, decoder and transcriptor. So that the temporal resolution of \mathbf{h} is consistent with that of \mathbf{X} . We describe the details of the encoder, decoder, and transcriptor in Section 4.2.

3.2 Query-by-example Separation and Transcription

As described in Equation (3) and (4), we need to build J decoders to separate J target sources. With the number of target sources increases, the parameters number will

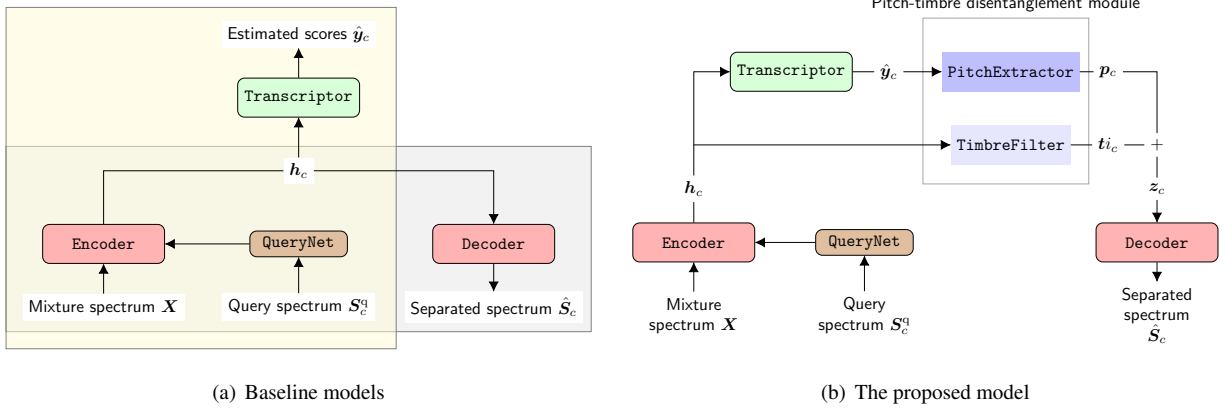


Figure 1. The baseline models and the proposed model. In the left figure, the large orange and gray box indicate a QBE transcription-only and QBE separation-only model respectively. The whole figure indicates a QBE multi-task model.

also increase. More importantly, the model trained for pre-defined sources can not adapt to unseen sources. To tackle these problems, we design a QBE module in our model. The advantage of using QBE is that we can separate unseen target sources. That is, we achieve a zero-shot separation.

Similar to the QueryNet [3], we design a QueryNet module as shown in Figure 1(a). The QueryNet module extracts the embedding vector $\mathbf{q}_c \in \mathbb{R}^M$ of an input spectrogram $\mathbf{S}_c^q \in \mathbb{R}^{T \times F}$, where M is the dimension of the embedding vector:

$$\mathbf{q}_c = \text{QueryNet}(\mathbf{S}_c^q). \quad (5)$$

Audio recordings from the same source will be learnt to have similar embedding vectors. We propose a contrastive loss to encourage embedding vectors from the same sources to be close, and embedding vectors from different sources to be far:

$$L_{\text{query}} = \frac{1}{C} \left\{ \left\| \mathbf{q}_c - \mathbf{q}'_c \right\|_2 + \sum_{j \neq c} \max(m - \left\| \mathbf{q}_c - \mathbf{q}_j \right\|_2, 0) \right\}, \quad (6)$$

where $m > 0$ is a margin and \mathbf{q}'_c and \mathbf{q}_c are from the same source c , and \mathbf{q}_c and \mathbf{q}_j are from different sources. We set M to 6 and m to 0.125 in our experiments.

We input the embedding vector \mathbf{q}_c as a condition to each layer of the encoder using Feature-wise Linear Modulation (FiLM) [25] layers. Then the encoder outputs a representation $\mathbf{h}_c \in \mathbb{R}^{T \times D}$. The embedding vector \mathbf{q}_c controls what source to separate or transcribe. There are only one encoder, one decoder, and one transcripator to separate any sources:

$$\mathbf{h}_c = \text{Encoder}(\mathbf{X}, \mathbf{q}_c), \quad (7)$$

$$\hat{\mathbf{S}}_c = \text{Decoder}(\mathbf{h}_c), \quad (8)$$

$$\hat{\mathbf{y}}_c = \text{Transcripator}(\mathbf{h}_c). \quad (9)$$

3.3 Pitch-timbre Disentanglement Module

Previous MSS works do not disentangle pitch and timbre for separation. That is, those MSS methods implement

separation systems without estimating pitches. In this section, we propose a pitch-timbre disentanglement module based on the query-based encoder-decoder architecture described in previous sections to learn interpretable representations for MSS. Such interpretable representations enable the model to achieve score-informed separation based on predicted scores.

As shown in Figure 1(b), the proposed pitch-timbre disentanglement module consists of a PitchExtractor and a TimbreFilter module. The output of PitchExtractor \mathbf{p}_c only contains pitch information of \mathbf{s}_c , and the output of TimbreFilter is expected to only contain timbre information of \mathbf{s}_c . The PitchExtractor is modeled by an embedding layer $\mathbf{V} = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_N\}$, where N is the number of vectors, which equals to the number of pitches 89 in our experiment. To explain, $\mathbf{e}_n \in \mathbf{V}$ ($\mathbf{e}_n \in \mathbb{R}^K$) denotes the quantized pitch vector for the n -th MIDI note. Then, we calculate the disentangled pitch representation for $\hat{\mathbf{y}}$ as $\mathbf{p}_c = [\mathbf{p}_c^{(1)}; \mathbf{p}_c^{(2)}; \dots; \mathbf{p}_c^{(T)}]$, where $\mathbf{p}_c^{(t)} \in \mathbb{R}^K$:

$$\mathbf{p}_c^{(t)} = \sum_n^N \hat{y}_c^{(t,n)} \cdot \mathbf{e}_n, \quad (10)$$

where $\hat{y}_c^{(t,n)}$ is the output of the transcripator containing predicted presence probability of the n -th MIDI note or the silence state at time t , and K is the dimension of the disentangled pitch representation. During synthesis, we can replace $\hat{\mathbf{y}}_c$ with one-hot encodings of new scores as input to Equation (10) to obtain pitch representation \mathbf{p}_c for synthesizing audio recordings.

TimbreFilter is used to filter timbre information $\mathbf{ti}_c \in \mathbb{R}^{T \times D}$ from \mathbf{h}_c :

$$\mathbf{ti}_c = \text{TimbreFilter}(\mathbf{h}_c). \quad (11)$$

Here, TimbreFilter is modeled by a convolutional neural network. Then, we can synthesize $\hat{\mathbf{s}}_c$ using disentangled pitch \mathbf{p}_c and timbre \mathbf{ti}_c . Inspired by the FiLM [25], we first split \mathbf{p}_c into \mathbf{p}_c^γ and \mathbf{p}_c^β , where $\mathbf{p}_c = [\mathbf{p}_c^\gamma; \mathbf{p}_c^\beta]$. Then, we entangle \mathbf{p}_c and \mathbf{ti}_c together to produce $\hat{\mathbf{S}}_c$:

$$\mathbf{z}_c = \mathbf{p}_c^\gamma \odot \mathbf{ti}_c + \mathbf{p}_c^\beta, \quad (12)$$

$$\widehat{S}_c = \text{Decoder}(z_c), \quad (13)$$

and the separation loss is:

$$L_{\text{separation}} = \left\| S_c - \widehat{S}_c \right\|_1. \quad (14)$$

Different from previous MSS works, we apply a separation loss and a transcription loss to train the proposed model. The transcription loss is:

$$L_{\text{transcription}} = \text{Cross_entropy}(\mathbf{y}_c, \widehat{\mathbf{y}}_c). \quad (15)$$

Here, $\mathbf{y}_c \in [0, 1]^{T \times N}$ is the groundtruth of scores. The aggregated loss function is:

$$L = L_{\text{query}} + L_{\text{transcription}} + L_{\text{separation}}. \quad (16)$$

The aggregated loss L drives the proposed model to be a *multi-task score-informed* model rather than a synthesizer due to the lack of inductive biases for further timbre disentanglement.

3.4 Pitch-translation Invariance Loss

We propose a pitch-translation invariance loss to further improve the timbre disentanglement performance. Based on the pitch-translation invariance, we assume that when the audio pitches with the corresponding MIDI is shifted within a certain interval, the timbre is unchanged.

We shift the pitch of s_c to generate an augmented audio s'_c . The augmented audio s'_c has the same timbre as s_c . According to Equation (1), we have a new mixture audio x' :

$$x' = s'_c + s_i. \quad (17)$$

We denote the X' and S'_c as the spectrograms of x' and s'_c respectively. We extract the disentangled timbre vector of s_c , and denote it as ti'_c . Because s'_c is pitch shifted s_c , so that the timbre ti'_c should be consistent with that of ti_c . Therefore, the reconstructed spectrogram by the timbre ti'_c and the pitch p_c should be consistent with S_c :

$$\widehat{z}'_c = p_c^\gamma \odot ti'_c + p_c^\beta, \quad (18)$$

$$\widehat{S}_c'' = \text{Decoder}(z'_c), \quad (19)$$

$$L_{\text{PTI}} = \left\| S_c - \widehat{S}_c'' \right\|_1. \quad (20)$$

where S_c'' is the reconstructed spectrogram. We denote L_{PTI} as a pitch-translation invariance loss. With L_{PTI} , our proposed model is capable of learning the disentanglement of pitch and timbre. A byproduct of the disentanglement system is that, the decoder of our system becomes a *synthesizer*, which can be used to synthesize audio recordings using timbre and pitches as input. When we change $\widehat{\mathbf{y}}_c$ to arbitrary scores, our model can synthesis a new piece of music with the timbre of S_c .

In total, the objective function we exploit to train the proposed model with further disentanglement includes a QueryNet loss, a transcription loss, and a pitch-translation invariance loss:

$$L' = L_{\text{query}} + L_{\text{transcription}} + L_{\text{PTI}}. \quad (21)$$

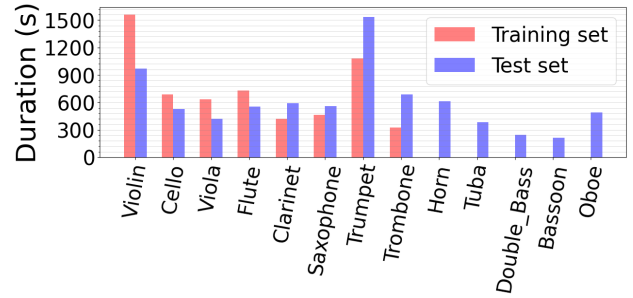


Figure 2. Duration of each instrument in the dataset.

4. EXPERIMENTS

4.1 Dataset and Pre-processing

We utilize the University of Rochester Multimodal Music Performance (URMP) dataset [26] as the experimental dataset. The URMP dataset is a multi-instrument audiovisual dataset covering 44 classical chamber music pieces remixed from 115 single-source tracks of 13 different *monophonic* instruments. The dataset provides note annotations for each single track. As shown in Figure 2, we divide these instruments into two groups (8 seen and 5 unseen instruments) and tracks into two sub-sets (55 tracks of 8 seen instruments for training and 32 songs by remixing 60 tracks of 13 instruments for test). Note that we calculate the duration of repeated tracks of different songs in the test set and do not exclude silence segments of all the tracks.

We resample all the tracks with a sample rate of 16KHz and extract them into Short-time Fourier transform (STFT) spectrograms with a window size of 1024 and 10ms overlap ($n_{\text{FFT}} = 2048$). During training, we randomly remix *2 arbitrary clips of different instruments* to generate a mixture. All the training data are augmented using pitch shifting (± 4 semitones) mentioned in Section 3.4.

4.2 Model Architecture

We design our models based on U-Net, the current prominent model in MSS. Figure 1(b) and 3 elaborates details of the proposed multi-task score-informed model (MSI) described in Section 3.3 and model with further disentanglement (MSI-DIS) illustrated in Section 3.4.

4.2.1 The Architecture of the MSI and MSI-DIS Model

The combination of the encoder and decoder is a general U-Net without temporal pooling. The QueryNet comprises 2 CNN blocks, each of which consists of 2 convolution layers and a 2×2 max pooling module. A fully-connected layer and a tanh activation layer are applied to the last feature maps. We then average output vectors over the temporal axis to get a 6-dimensional query embedding vector q_c . The architecture of the transcriptor is similar to the QueryNet but without temporal pooling. Each blue block in TimbreFilter depicted in Figure 3 is a 2-dimension convolutional layer, the shape of the tensor output by which is as same as that of the input tensor. Each deep blue block in PitchExtractor is a 1-dimension 1×1 convolutional layer.

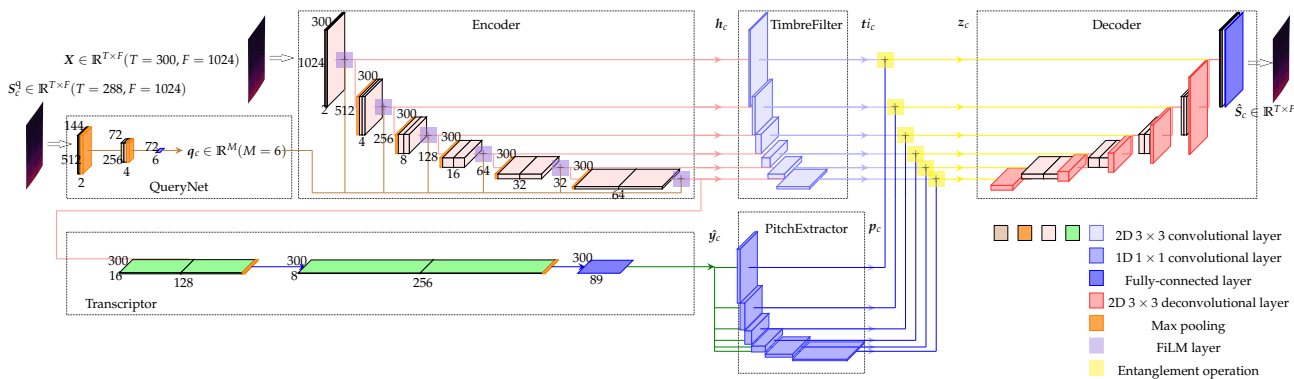


Figure 3. The model architecture with detailed hyper-parameter configuration.

Model	Separation (SDR)				Transcription (Precision)			
	MSS-only	Multi-task	MSI(ours)	MSI-DIS(ours)	AMT-only	Multi-task	MSI(ours)	MSI-DIS(ours)
Seen	4.69 ± 0.31	3.32 ± 0.18	6.33 ± 0.17	5.04 ± 0.16	0.72 ± 0.01	0.72 ± 0.04	0.71 ± 0.01	0.72 ± 0.01
Unseen	6.20 ± 0.26	4.63 ± 0.34	5.53 ± 0.15	3.99 ± 0.22	0.61 ± 0.01	0.58 ± 0.02	0.61 ± 0.01	0.59 ± 0.01
Overall	5.07 ± 0.22	3.65 ± 0.22	6.13 ± 0.15	4.77 ± 0.14	0.69 ± 0.01	0.69 ± 0.03	0.69 ± 0.01	0.68 ± 0.00

Table 1. The separation and transcription performance of all models..

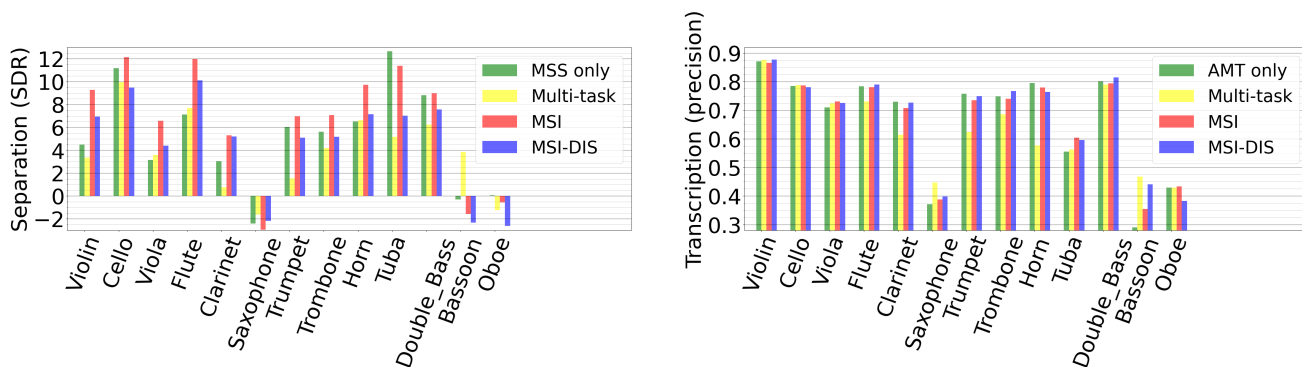


Figure 4. Instrument-wise performance of models. The last 5 instruments are unseen in the training set.

Typically, the bottleneck of U-Net is regarded as h_c . However, when constructing disentangled timbre representations, we regard the set of concatenate residual tensors as h_c to avoid non-disentangled representations leaking into the decoder.

Note that the kernel size of each 2-dimension convolutional layer is 3×3 and each 2-dimension convolutional layer (excepting TimbreFilter) is followed by a ReLU activation layer and a Batch Normalization layer.

4.2.2 Baseline Design

As shown in Figure 1(a), besides the proposed models illustrated above, we also report the performance of 3 extra baseline models in our experimental results. The QBE transcription-only baseline model (AMT-only) is composed of the queryNet, encoder, and transcripter; the QBE separation-only baseline model (MSS-only) is a general U-Net; the QBE multi-task baseline model is composed of a U-Net and a transcripter. All the hyper-parameters of components in these models are consistent with those of corresponding components in our models.

4.3 Training and Evaluation

All the models are trained with a mini-batch of 12 audio pairs for 200 epochs. All the models are evaluated with source-to-distortion (SDR) computed by mir_eval package [27] for separation and precision computed by sklearn package [28] for transcription. During training, each audio pair comprises 2 single-track audio clips of different instruments to generate a mixture, 2 correspondings augmented samples for pitch-transformation invariance loss, and 3 single-track audio clips that exclude silence segments for contrastive loss. During inference, each test pair comprises a 4-second audio mixture and query sample. During synthesis, we employ Griffin Lim Algorithm (GLA) [29] as the phase vocoder using torchaudio library. Since we do not divide a validation set to chose the best-performance model among all the training epochs, we report Micro-average results with a 95% confidence interval (CI) of models at the last 10 epochs. All the experimental results are reproducible via our released source code ¹.

¹ <https://github.com/kikyo-16/a-unified-model-for-zero-shot-musical-source-separation-transcription-and-synthesis>

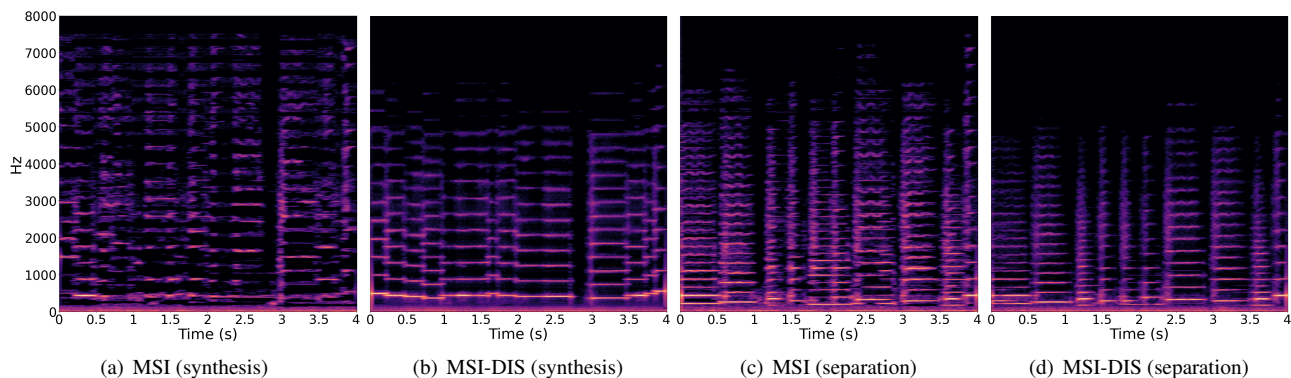


Figure 5. Spectrograms of audios synthesized and separated by the MSI and MSI-DIS model respectively. Models are expected to separate a viola source from the mixture of clarinet and viola. During synthesis, the two models are given the same new scores and are expected to synthesis new pieces with these scores and the separated viola timbre.

5. RESULTS

Experimental results shown in Table 1 demonstrate that the proposed MSI model outperforms baselines on separation without sacrificing performance on transcription. The instrument-wise performance on unseen instruments depicted in Figure 4 demonstrates that the proposed models are capable of performing zero-shot transcription and separation. We also release synthesized audio demos online². These demos demonstrate the success of the proposed inductive biases for disentanglement.

5.1 Multi-task Baseline vs Single-task Baselines

As shown in Table 1, the multi-task baseline performs worse than the separation-only baseline, suggesting that the joint representation requires extra inductive biases to learn better generalization, i.e. deep clustering in Cerberus [14]. Our disentanglement strategy provides such inductive biases.

5.2 MSI model vs Baselines

With the auxiliary of the proposed pitch-timbre disentanglement module, compared with the multi-task baseline, the performance of MSI on separation becomes better. This indicates that the disentanglement module improve the generalization capability of the joint representation, leading to better separation results. Meanwhile, MSI outperforms the MSS-only baseline on separation by 1.06 points. This demonstrates that inaccurate scores transcribed by the model itself sever as a powerful auxiliary for separation.

5.3 MSI Model vs MSI-DIS Model

As depicted in Figure 5(a) and 5(b), it is interesting that despite the same “hardware” (neural network design) of the two models, the MSI model fails to synthesis but the MSI-DIS model achieves. It exactly demonstrates that the designed “soft-ware” (the pitch translation loss) takes effect on the success of the disentanglement. As for separation performance shown in Table 1, the MSI-DIS model

falls behind the MSI model. The observation that better synthesis quality does not implies better separation performance suggests a trade-off between disentanglement and reconstruction. It indicates that extra (well-suited) inductive biases are required to further improve pitch and timbre disentanglement at the same time reduce the loss of information necessary for reconstruction.

Comparing the performance on seen with unseen instruments shown in Table 1, we find that the separation quality of the MSI-DIS model is more sensitive to the accuracy of transcription results than that of the MSI model. This is because the MSI-DIS model synthesizes instead of separating sources, for which the separation performance of it relies more on the accuracy of transcription results and the capability of the decoder than the MSI model does. However, when comparing separated spectrograms shown in Figure 5(c) and 5(d), we find that the MSI model sometimes separates multiple pitches at the same time while the MSI-DIS model yields monophonic results that sound more “clean”. We release more synthesized and separated audio demos online.

6. CONCLUSION AND FUTURE WORKS

We contributed a unified model for zero-shot music source separation, transcription, and synthesis via pitch and timbre disentanglement. The main novelty lies in the disentanglement-and-reconstruction methodology for source separation, which naturally empowers the model with transcription and synthesis capabilities. In addition, we designed well-suited inductive bias including pitch vector quantization and pitch-translation invariant timbre loss to achieve better disentanglement. Lastly, we successfully integrate the model with a query-based networks, so that all three tasks can be achieved in a zero-shot fashion for unseen sound sources. Experiments demonstrated the zero-shot capability of the model and the powerful auxiliary of disentangled pitch information to separation. Results of synthesized audio pieces further exhibit that the disentangled factors are well generalized. In the future, we plan to extent the proposed framework for multi-instrument and vocal scenarios as well as high-fidelity synthesis.

² <https://kikyo-16.github.io/demo-page-of-a-unified-model-for-separation-transcription-synthesis>

7. REFERENCES

- [1] T. Yoshioka, H. Erdogan, Z. Chen, and F. Alleva, "Multi-microphone neural speech separation for far-field multi-talker speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5739–5743.
- [2] T. Yoshioka, H. Erdogan, Z. Chen, and F. Alleva, "Multi-microphone neural speech separation for far-field multi-talker speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5739–5743.
- [3] J. H. Lee, H.-S. Choi, and K. Lee, "Audio query-based music source separation," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [4] P. Seetharaman, G. Wichern, S. Venkataramani, and J. Le Roux, "Class-conditional embeddings for music source separation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 301–305.
- [5] D. Samuel, A. Ganeshan, and J. Naradowsky, "Meta-learning extractors for music source separation," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 816–820.
- [6] E. Manilow, G. Wichern, and J. Le Roux, "Hierarchical musical instrument separation," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [7] J. F. Woodruff, B. Pardo, and R. B. Dannenberg, "Remixing stereo music with score-informed source separation." in *ISMIR*. Citeseer, 2006, pp. 314–319.
- [8] M. Miron, J. Janer Mestres, and E. Gómez Gutiérrez, "Monaural score-informed source separation for classical music using convolutional neural networks," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.
- [9] S. Ewert and M. B. Sandler, "Structured dropout for weak label and multi-instance learning and its application to score-informed source separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2277–2281.
- [10] M. Gover, "Score-informed source separation of choral music," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [11] G. Meseguer-Brocal and G. Peeters, "Content based singing voice source separation via strong conditioning using aligned phonemes," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [12] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The sound of pixels," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 570–586.
- [13] C. Gan, D. Huang, H. Zhao, J. B. Tenenbaum, and A. Torralba, "Music gesture for visual sound separation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 478–10 487.
- [14] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 771–775.
- [15] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 23–27.
- [16] N. Takahashi, N. Goswami, and Y. Mitsufuji, "Mmdenselstm: An efficient combination of convolutional and recurrent neural networks for audio source separation," in *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 2018, pp. 106–110.
- [17] Y.-N. Hung and A. Lerch, "Multitask learning for instrument activation aware music source separation," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [18] K. Tanaka, T. Nakatsuka, R. Nishikimi, K. Yoshii, and S. Morishima, "Multi-instrument music transcription based on deep spherical clustering of spectrograms and pitchgrams," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [19] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 31–35.
- [20] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 686–690.
- [21] Y. Luo, Z. Chen, and N. Mesgarani, "Speaker-independent speech separation with deep attractor network," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 4, pp. 787–796, 2018.
- [22] R. Kumar, Y. Luo, and N. Mesgarani, "Music source activity detection and separation using deep attractor network." in *INTERSPEECH*, 2018, pp. 347–351.

- [23] C.-B. Jeon, H.-S. Choi, and K. Lee, “Exploring aligned lyrics-informed singing voice separation,” in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [24] G. Meseguer-Brocal and G. Peeters, “Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [25] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [26] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, “Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications,” *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.
- [27] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, “Mir_eval: A transparent implementation of common mir metrics,” in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.
- [28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [29] A. Sharma, P. Kumar, V. Maddukuri, N. Madamshetti, K. Kishore, S. S. S. Kavuru, B. Raman, and P. P. Roy, “Fast griffin lim based waveform generation strategy for text-to-speech synthesis,” *Multimedia Tools and Applications*, vol. 79, no. 41, pp. 30 205–30 233, 2020.

PITCH-INFORMED INSTRUMENT ASSIGNMENT USING A DEEP CONVOLUTIONAL NETWORK WITH MULTIPLE KERNEL SHAPES

Carlos Lordelo^{1,2}

Emmanouil Benetos¹

Simon Dixon¹

Sven Ahlbäck²

¹ Queen Mary University of London, UK

² Doremir Music Research AB, Sweden

c.p.viannalordelo@qmul.ac.uk

ABSTRACT

This paper proposes a deep convolutional neural network for performing note-level instrument assignment. Given a polyphonic multi-instrumental music signal along with its ground truth or predicted notes, the objective is to assign an instrumental source for each note. This problem is addressed as a pitch-informed classification task where each note is analysed individually. We also propose to utilise several kernel shapes in the convolutional layers in order to facilitate learning of timbre-discriminative feature maps. Experiments on the MusicNet dataset using 7 instrument classes show that our approach is able to achieve an average F-score of 0.904 when the original multi-pitch annotations are used as the pitch information for the system, and that it also excels if the note information is provided using third-party multi-pitch estimation algorithms. We also include ablation studies investigating the effects of the use of multiple kernel shapes and comparing different input representations for the audio and the note-related information.

1. INTRODUCTION

Automatic Music Transcription (AMT) is the process of creating any form of notation for a music signal and is currently one of the most challenging and discussed topics in the Music Information Retrieval (MIR) community [1]. Most AMT systems are designed to transcribe a single monophonic or a single polyphonic source into a musical score (or piano-roll). In this case, the main sub-task involved in the process is Multi-Pitch Estimation (MPE), where predictions regarding the pitch and time localisation of the musical notes are carried out. However, when analysing polyphonic multi-instrumental recordings, not only each note should have its pitch and duration properly estimated, but the information regarding the timbre of sounds should also be correctly processed [2]. It is mandatory to have a way of recognising the instrument that played each note.

In this paper, we propose a pitch-informed instrument assignment approach, where the main objective is to asso-

ciate each note event of a music signal to one instrument class. In contrast to other state-of-the-art instrument recognition approaches, which are usually addressed on a frame-level [3, 4] or clip-level [5, 6] basis, our approach analyses each note event individually. Therefore, it is possible to say that we perform a note-level instrument recognition.

Previous work has shown that the use of pitch information can help frame-level instrument recognition [3]. Inspired by this, we propose a framework that uses an auxiliary input based on note-event pitch information. Our system is trained using the note annotations provided in the MusicNet [7] dataset. However, our main motivation is to create a modular framework that can be combined with any MPE algorithm in order to obtain multi-instrumental pitch predictions, which allows for transcribing music in staff notation, corresponding to the perception of pitch events. Therefore, we also show that our approach can obtain good performance when the note information is predicted by state-of-the-art MPE algorithms such as [8, 9].

Furthermore, the utilisation of multiple kernel shapes in the filters of a Convolutional Neural Network (CNN) has been proven to be an efficient strategy of applying domain knowledge in several MIR tasks [10–12]. In particular, [12] applied this strategy with a dense connectivity pattern of skip-connections in order to learn even more efficient feature maps and reduce the number of trainable parameters for the task of source separation. In our work, we build our CNN adapting the architecture in [12] for the classification (instrument assignment) task and verified that it can also improve its performance. In summary, the main contributions of this paper are as follows:

- Pitch-informed instrument assignment: Proposal of a Deep Neural Network (DNN) that associates each note from a music signal to its instrumental source.
- Modular Framework: Approach works with any MPE method. We evaluate the performance when using ground-truth note labels as well as 2 state-of-the-art MPE algorithms [8, 9].
- Multiple Kernel Shapes: Proposal of a CNN architecture for instrument assignment that uses multiple kernel shapes for the convolutions, facilitating learning representations for different instruments and note sound states. We show that their use improves instrument assignment performance.



Set	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	Harps.	Bass	Oboe	Flute	Total
Train	628549	197229	88446	89356	10770	13874	22873	4914	3006	8624	8310	1075951
	58.4%	18.3%	8.2%	8.3%	1.0%	1.2%	2.1%	0.5%	0.3%	0.8%	0.8%	100%
Test	5049	3238	842	1753	557	873	1277	0	0	0	0	13589
	37.2%	23.8%	6.2%	12.9%	4.1%	6.4%	9.4%	0	0	0	0	100%

Table 1. Statistics of the note labels provided by MusicNet across train and test sets.

2. RELATED WORK

The instrument recognition task is usually formulated as a multi-label classification task that can be addressed either on a frame-level [3, 4, 13], where the purpose is to obtain the instrument activations across time, or on a clip-level basis [5, 6, 14], where the purpose is to estimate the instruments that are present in an audio clip. However, our objective in this work is to approach the instrument recognition task note by note, assigning an instrument class to each. Such a task requires note-event annotations and, in the literature, it is also known as *instrument assignment* [15] or *multi-pitch streaming* [2].

Just few works have explored this particular task. For instance, Duan et al. [2] approached it using a constrained clustering of frame-level pitch estimates obtained from an MPE algorithm via the minimisation of timbre inconsistency within each cluster. They tested different timbre features for both music and speech signals. In [16], a similar method was proposed, where the authors applied Probabilistic Latent Component Analysis (PLCA) to decompose the audio signal into multi-pitch estimates and to extract source-specific features. Then, clustering was performed under the constraint of cognitive grouping of continuous pitch contours and segregation of simultaneous pitches into different source streams using Hidden Markov Random Fields. Both of those works, however, assume that each source is monophonic, i.e., each instrument could only play a single note at a time.

An alternative approach is to model the temporal evolution of musical tones [15]. This method is based around the use of multiple spectral templates per pitch and instrument source that correspond to sound states. The authors used hidden Markov model-based temporal constraints to control the order of the templates and streamed the pitches via shift-invariant PLCA. In a more recent work, Tanaka et al. [17] also approached the task via clustering, but applied on a joint input representation combined of the spectrogram and the pitchgram, which was obtained using an MPE algorithm. In their proposal, each bin of the joint input was encoded onto a spherical latent space taking into account timbral characteristics and the piano-rolls of each instrument were later estimated via masking of the pitchgram based on the results of a deep spherical clustering technique applied on the latent space.

Recent multitask deep-learning based works have successfully proposed multi-instrumental AMT methods that are able to directly estimate pitches and associate them to their instrumental source jointly [4, 18, 19]. In [18],

a multitask deep learning network jointly estimated outputs for various tasks including multiple-pitch, melody, vocal and bass line estimation. The Harmonic Constant-Q Transform (HCQT) of the audio signal was used as input and the data used for training was semi-automatically labelled by remixing a diverse set of multitrack audio data from the MedleyDB [20] dataset. In [4] a DNN was used to jointly predict the pitch and instrument for each audio frame. They used the Constant-Q Transform (CQT) as input to their system and trained using a large amount of audio signals synthesised from MIDI piano-rolls. Manilow et al. [19], on the other hand, were able to jointly transcribe and separate an audio signal into up to 4 instrumental sources — piano, guitar, bass and strings. However, their system was trained with only synthesised signals.

Our approach is closely related to that of Hung and Yang [3], where a frame-level instrument recogniser is proposed using the CQT spectrogram of the music signal allied with the pitch information of the note events. We also use the pitch annotations to guide the instrument classifier, but our work differs from [3] in the fact that we perform a classification for each note event individually, while Hung and Yang use the whole piano-roll at once to guide frame-level instrument recognition. While Hung and Yang are able to obtain the instrument activations leveraging from the pitch information, they cannot stream the note events into their corresponding instruments.

3. PROPOSED METHOD

In our method, we use the same definition of note events as in the MIREX MPE task¹. Each note N is considered an event with a constant pitch f_0 , an onset time T_{on} and an offset time T_{off} . Therefore, if a music signal has a total of M notes, any note N_i , with $i \in \{1, \dots, M\}$, can be uniquely defined by the tuple $(f_0^i, T_{on}^i, T_{off}^i)$. In our experiments, we use two ways of obtaining this note information. The first using ground-truth pitch labels provided by the employed dataset (MusicNet) [7] and the second using pitch estimates predicted by state-of-the-art MPE algorithms [8, 9]. We consider the f_0 granularity to follow the semitone scale, ranging from $A0$ to $G\sharp7$ (MIDI #21 – 104).

In our framework polyphony is allowed, so, most of the time more than a single note will be active, but our objective is to analyse each note of the audio signal separately in order to be able to assign an instrument class to it. This is done by using two inputs to the model: the main in-

¹ <https://www.music-ir.org/mirex/>

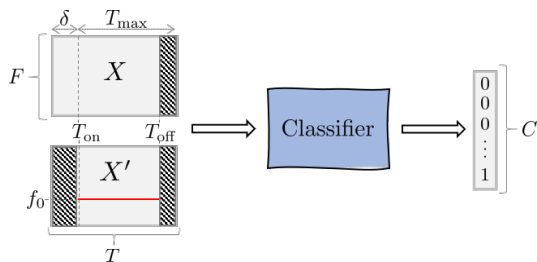


Figure 1. Overview of the proposed framework for instrument assignment. See Section 3 for the detailed explanation of the variables in the figure.

put $X(f, t)$, with f representing frequency and t representing time, is a time-frequency representation of a segment of the audio signal around the value of T_{on} , and an auxiliary input $X'(f, t)$, which carries information regarding f_0 , T_{on} and T_{off} . The two inputs are concatenated into a two-channel input $\mathbf{X}(f, t, c)$, where $c \in \{1, 2\}$ represents the channel dimension, that is fed to the model. In Figure 1 an overview of the proposed framework is shown.

3.1 Main Audio Input

The main input is a time-frequency representation $X(f, t) \in \mathbb{R}^{F \times T}$ of a small clip of the music signal, where F is the number of frequency bins and T is the number of time frames. The clip is generated by first setting a maximum duration T_{max} for the note. We tested values of T_{max} ranging from 400 ms to 1 s (see Section 7 for details) and 400 ms obtained the best results, so we kept this value in all of the other experiments. If any note N_i has a duration D_i greater than T_{max} , i.e., $D_i = T_{\text{off}} - T_{\text{on}} > T_{\text{max}}$, only its initial time span of T_{max} seconds is considered.

Next, for every note N_i , $X_i(f, t)$ is constructed by picking a segment of duration $T = T_{\text{max}} + \delta$ from the original music signal starting from $T_{\text{on}}^i - \delta$, where δ is a small interval to take into account deviations between the true onset value and the value we use. The inclusion of the extra window of δ from the music signal also helps the convolutional layers since it brings some context of the signal before the note onset value. We set $\delta = 30$ ms after initial tests. Lastly, if the note duration D_i is less than T_{max} , we set the values of $X(f, t > D_i)$ to zero, where $D_i = T_{\text{off}}^i - T_{\text{on}}^i$.

3.2 Auxiliary Note-Related Input

The auxiliary input $X'(f, t) \in \mathbb{R}^{F \times T}$ is a harmonic comb representation using the pitch value f_0 as the first harmonic², such that,

$$X'(f, t) = \begin{cases} 1, & \text{if } f = hf_0 \text{ and } T_{\text{on}} \leq t \leq T_{\text{off}} \\ 0, & \text{otherwise} \end{cases}, \quad (1)$$

where $h = \{1, 2, 3, \dots, H\}$ with H being the total number of harmonics in the representation. We tested multiple values for H (see Section 6). In practice, we use a tolerance of half a semitone for each harmonic value when con-

² We use the definition that f_0 corresponds to the first harmonic.

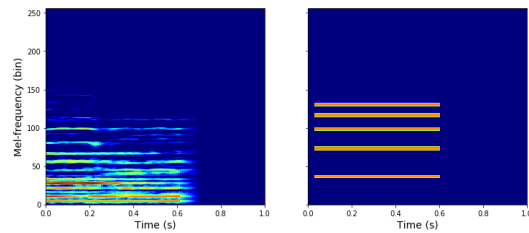


Figure 2. A pair of inputs using 256 mel-frequency spectrogram. In the left is depicted $X(f, t)$, where three pitches are simultaneously activated (MIDI # 58, 62 and 74) and in the right $X'(f, t)$, where the note with pitch # 74, is modelled using an harmonic comb of $H = 5$. In this example, $D_i = 600$ ms and $T_{\text{max}} = 1$ s.

structing $X'(f, t)$ as a mel-spectrogram. Therefore, even though this representation starts as binary, the final mel-spectrogram is not binary due to the mel-filtering procedure. Moreover, it is important to note that we also set the values of X' before T_{on} and after T_{off} to zero. In Figure 2 we show an example of a pair of inputs for our framework.

3.3 Output

The note-level instrument assignment task is tackled as a multi-class single-label classification task. Given \mathbf{X} , our objective is to classify it as belonging to one of C instrument classes. We use a deep neural network that receives \mathbf{X} as input and outputs a C -dimensional vector \hat{y} . A softmax activation function is applied in the final layer of the network to ensure the values of \hat{y} represent probabilities that sum up to 1. The model is trained using the cross-entropy loss. At inference time, the class corresponding to the dimension with the highest value in \hat{y} is predicted. See Section 4 for details regarding the network architecture.

In the cases where two or more instruments are playing the same pitch simultaneously, the small differences between the notes' onset and offset values can generate different inputs \mathbf{X} . Thus, it would still allow the instrument assignment task to be properly executed as a single label classification scenario. However, when the pitch, onset and offset values of notes from different instruments exactly match, our system will consider them as a single note and only a single instrument will be estimated. This case rarely happens in real-world scenarios for many musical styles. For instance, in MusicNet only 0.9% of the notes had the same pitch, onset, and offset values. For our experiments, we have considered notes in MusicNet that were performed by a single instrument, and discarded the notes that were concurrently produced (in terms of the same pitch, onset, and offset times) by multiple instruments. As a proof of concept, we believe that this is not a severe limitation for our framework and we leave multi-labelled approaches as future work.

4. ARCHITECTURE

When processing music spectrograms by CNNs, the strategy of combining vertical and horizontal kernel shapes in

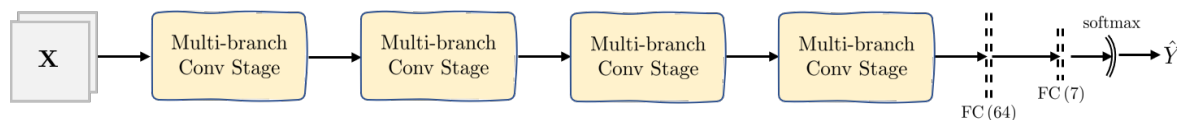


Figure 3. Proposed network architecture. FC represents a Fully Connected layer with LeakyRelu activation function.

Main Input	Aux Input	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	Mean
CQT	—	0.960	0.732	0.116	0.725	0.512	0.296	0.681	0.575
	$H = 1$	0.994	0.934	0.763	0.955	0.783	0.888	0.950	0.895
	$H = 2$	0.993	0.939	0.771	0.960	0.785	0.858	0.929	0.891
	$H = 3$	0.992	0.946	0.772	0.957	0.754	0.884	0.952	0.894
	$H = 4$	0.993	0.938	0.766	0.958	0.784	0.869	0.950	0.894
	$H = 5$	0.993	0.939	0.767	0.952	0.769	0.874	0.949	0.892
Mel STFT	—	0.967	0.742	0.222	0.730	0.607	0.306	0.690	0.609
	$H = 1$	0.996	0.939	0.759	0.958	0.780	0.867	0.958	0.895
	$H = 2$	0.994	0.945	0.779	0.956	0.809	0.864	0.946	0.899
	$H = 3$	0.997	0.944	0.775	0.958	0.8104	0.879	0.967	0.904
	$H = 4$	0.996	0.935	0.747	0.945	0.839	0.891	0.960	0.902
	$H = 5$	0.996	0.947	0.783	0.954	0.801	0.876	0.954	0.902

Table 2. Evaluation of instrument assignment task when using CQT or Mel spectrograms as input representation for the network as well as a comparison between models trained with no auxiliary input and models trained with different number of harmonics in the auxiliary input. This experiment was performed using $T_{max} = 400$ ms

the model architecture can facilitate learning of timbre-discriminative feature maps [10–12]. In our work, we propose a CNN adapted from the 3W-MDenseNet [12]. This architecture was originally proposed for harmonic-percussive source separation and consists of an encoder-decoder model that estimates spectrograms for two sources. Thus, the outputs of the 3W-MDenseNet have the same shape as the mixture spectrogram that is used as input. In this CNN architecture, three MDenseNets [21] run in parallel in separated branches, each with a unique kernel shape (vertical, square and horizontal). The MDenseNets are only combined at the final layer, i.e., after both the encoding and decoding procedure are performed. In our work, we adopt a similar methodology by taking only the encoder layers from [12] and adding fully connected layers at the end in order to perform classification rather than separation. Also, we propose modifications to the original encoder layers: instead of combining the branches using a concatenation layer only at the final stage, we concatenate their feature maps at the end of each down-sampling stage. By doing so, we allow each branch to have access to feature maps computed using all different choices of kernel shapes from a previous stage.

Figure 3 shows a summary of the architecture we adopt in our work. It consists of a stack of 4 multi-branch convolutional stages and 2 fully connected layers. In Figure 4 the internal structure of the multi-branch convolutional stage is shown. Internally, each multi-branch convolutional stage contains 3 separate branches whose convolu-

tions have unique kernel shapes. We use a branch with horizontal (1×9), a branch with square (3×3), and a branch with vertical (9×1) convolutions. In each path, a Densely connected convolutional Network (DenseNet) [22] with growth rate $k = 25$ and number of layers $L = 4$ is used. In short, a DenseNet is a stack of L k -channel convolutional layers — each with its own activation function — with a dense pattern of skip connections, where each layer receives the concatenation of all previous layers’ outputs as input. We used the LeakyRelu function as the activation function for all layers. The reader is referred to [22] for the detailed internal structure of a DenseNet. After the DenseNet, a (2×2) max pooling layer is applied in order to reduce the feature maps’ dimensions and increase the receptive field at each branch. Afterwards, the three branches are concatenated and the batch is normalised. The final feature maps are used as input for the next multi-branch convolutional stage. Since we need to concatenate feature maps that were originated by multiple kernel shapes we use padding on the convolution and on the max pooling to ensure the feature maps maintain the same dimensions across branches. The number of trainable parameters is approximately 1.1 million.

5. DATASET

We used the MusicNet dataset [7] in our experiments. MusicNet is the largest publicly available dataset with non-synthesised data that is strongly labelled for the task of

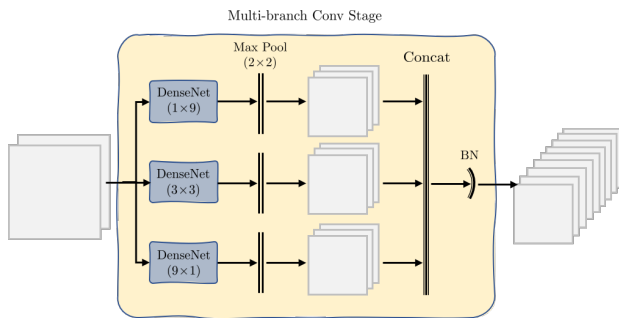


Figure 4. Internal structure of a multi-branch conv. stage. Each DenseNet has a growth rate $k = 25$ and $L = 4$ layers. BN represents a Batch Normalisation layer. We use LeakyRelu as activation function after each conv. layer.

instrument recognition. This means that we know the exact frames where the instruments are active in the signal, which permits the training of supervised models to perform instrument recognition at the frame-level, note-level, and clip-level. The dataset contains 330 freely-licensed classical music recordings by 10 composers, written for 11 instruments, along with over 1 million annotated labels indicating the precise time and pitch of each note in the recordings and the instrument that plays each note.

The instrument taxonomy of MusicNet is: piano, violin, viola, cello, french horn, bassoon, clarinet, harpsichord, bass, oboe and flute. However, the last 4 instruments (harpsichord, bass, oboe and flute) do not appear in the original test set provided by the authors. Therefore, in all our experiments we ignored all the labels related to those instruments and we performed a 7-class instrument classification using the following classes: piano, violin, viola, cello, french horn, bassoon and clarinet. Table 1 shows the statistics of the note labels provided by MusicNet. The dataset is heavily biased towards piano and violin given their usual presence in Western classical music recordings.

6. EXPERIMENTAL SETUP

In all experiments we used the original train/test split provided by MusicNet with the original sampling frequency of 44100 Hz. For experiments that involved the computation of Short Time Fourier Transform (STFT) we used Blackman-Harris windows of 4096 samples to compute the Discrete Fourier Transform (DFT). The hop size was always set to 10 ms in every experiment.

From the training set we picked 5% of the notes of each class and created a validation set. We trained the models using the Adam optimiser with an initial learning rate of 0.001 and reduced it by a factor of 0.2 if the cross-entropy loss stopped improving for 2 consecutive epochs on the validation set. If no improvement happened after 10 epochs, the training was stopped early. The experiments were performed using the Tensorflow/Keras Python package.

The classification performance was evaluated by computing the note-level F-score (F_s), which is directly related

to the precision (P), recall (R) according to:

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN}, F_s = \frac{2PR}{P + R} \quad (2)$$

where TP is the number of true positives, FP the false positives and FN the false negatives.

For the cases when the instrument assignment is done on top of MPE algorithms, we provide 2 groups of metrics that are generated following the MIREX evaluation protocol for the music transcription task. In the first group, an estimated note is assumed correct if its onset time is within 50 ms of a reference note and its pitch is within quarter tone of the corresponding reference note. The offset values are ignored. In the second group, on top of those requirements, the offsets are also taken into consideration. An estimate note is only considered correct if it also has an offset value within 50 ms or within 20% of the reference note's duration around the original note's offset, wherever is largest. After all notes are verified, the F-score is computed note-wise across time and the average value is provided here. This evaluation method was computed using the `mir_eval.transcription`³ toolbox.

7. RESULTS

7.1 Effects of the Kernel Shapes

First we analysed the effects of the inclusion of multiple kernel shapes in the architecture of the CNN. The top part of Table 3 compares 3 versions of the model: one that uses only square filters in a single branch; a version using the branched structure, but with (3×3) kernels in each; and another model with the proposed multi-branch structure with horizontal, square, and vertical kernels. For the single-branched case we increased the growth factor of the DenseNets to 57 channels in order to keep the number of trainable parameters of the network close to the original.

Analysing the results we see that the addition of new kernel shapes improved the average F-score across all classes. Regarding each instrument class, we can say that for string instruments (piano, violin, viola and cello) there is a gain in performance, while for non-string instruments (horn, bassoon, clarinet) the performance either drops or remains with a negligible gain if compared to the models that used only square filters. This suggests that the inclusion of vertical kernel shapes helped the model in learning the percussive characteristics of the timbre of string musical instruments.

7.2 Evaluation of the Input Size

We also tested different values for the input size. More specifically, we compared multiple values for T_{\max} , which is the maximum valid window of analysis for a note event. The results are shown in the lower part of Table 3. We can see that the shortest input size of 400 ms obtained the best results. We believe that it is due to the fact that the average duration of a note event in the test set of MusicNet is

³ https://craffel.github.io/mir_eval/

Kernel / T_{max}	Piano	Violin	Viola	Cello	Horn	Bassoon	Clarinet	Mean
(3×3)	0.994	0.936	0.757	0.954	0.826	0.864	0.954	0.898
3×(3×3)	0.995	0.939	0.764	0.945	0.819	0.896	0.965	0.903
Multiple	0.997	0.944	0.775	0.958	0.810	0.879	0.967	0.904
400 ms	0.997	0.944	0.775	0.958	0.810	0.879	0.967	0.904
600 ms	0.996	0.942	0.771	0.954	0.826	0.881	0.959	0.904
800 ms	0.996	0.944	0.772	0.957	0.814	0.868	0.965	0.902
1 s	0.997	0.931	0.740	0.954	0.742	0.871	0.948	0.883

Table 3. Instrument assignment performance based on the kernel shapes used in network (first three rows) and based on the value used for the maximum valid note duration T_{max} . The metric shown is the F-score achieved by each class and the average value across all instruments.

260 ms and the 90th percentile is 0.464 ms. So, the value of 400 ms is already enough to represent the vast majority of the notes. Moreover, when the analysed note event is longer than 400 ms, the 400 ms initial window contains most of the important features for the model.

7.3 Auxiliary Input and Types of Representations

To test the importance of the auxiliary input and how its modification would affect the performance of the model, we also tested a version of the model using only the main mel spectrogram input and versions using different numbers of harmonics H in the auxiliary input (from $H = 1$ to $H = 5$). We also tested two types of input representation for the model, the Constant-Q Transform (CQT) and the mel-frequency spectrogram. The CQT was computed using 12 bins per octave and a total of 115 bins starting from $G\sharp_0$ (MIDI #20). The mel-frequency spectrogram was computed by a linear transformation of an STFT onto a mel-scaled frequency axis, using 256 mel-bins. The results are provided in Table 2.

Analysing the results, it is possible to say that the auxiliary input is extremely necessary for the framework. Without it, the average F-score only reaches 60.9%, while with it the performance improves up to 90.4%. Apart from piano, all other classes have a large decrease in performance when we exclude the auxiliary input from \mathbf{X} . We believe that the results for the piano class continue to be high not only because of the MusicNet bias towards piano, but also because some recordings of the test set are solo piano recordings, which facilitates the classification of piano notes when analysing the main input signal due to the absence of other classes. Regarding the number of harmonics used in the auxiliary input, we can see that, in general, the CQT works best with few harmonics, while the Mel-STFT prefers higher values. A possible explanation for this is the fact that it is harder to represent odd harmonics on the CQT using a log-frequency resolution of 12 bins per octave. However, more experiments are needed in order to better investigate this assumption.

7.4 Streaming of Multi-Pitch Estimations

Once we verified that our model obtains impressive performance when the original ground-truth labels are used, we tested the classifier in a more realistic environment, where

no note-event labels are readily available. We estimated frame-level pitch values using two third-party MPE algorithms [8, 9]. For the algorithm in [8] we obtained an implementation from the original authors while an implementation of [9] is available via the project Omnizart⁴. We ran both algorithms on the music recordings to obtain the note events in order to construct the input to the classifier.

It is important to observe that errors in the MPE estimation will be carried over to the instrument assignment task. If a note is wrongly estimated, no ground-truth class for the instrument assignment exists, so it is hard to evaluate the results in the same way we did for the other experiments. So, in this experiment we used the transcription metrics that we explained in the last paragraph of Section 6. The results appear in Table 4. Given the limitations of each MPE method we used, we can see that our approach can successfully generate multi-instrument transcriptions.

Instr.	Onset			Onset + Offset		
	GT	[8]	[9]	GT	[8]	[9]
MPE-only	1	0.633	0.480	1	0.423	0.200
piano	0.942	0.745	0.451	0.942	0.497	0.196
violin	0.997	0.529	0.499	0.997	0.381	0.225
viola	0.775	0.366	0.308	0.775	0.227	0.116
cello	0.954	0.596	0.570	0.954	0.507	0.258
horn	0.804	0.460	0.429	0.804	0.232	0.166
bass.	0.874	0.473	0.373	0.874	0.193	0.130
clar.	0.967	0.616	0.456	0.967	0.344	0.165

Table 4. Transcription results when using Ground-Truth (GT) labels and when using two different MPE methods. In the row "MPE-only" no instrument assignment is done, we evaluate the multi-pitch estimates using the reference ground-truth notes ignoring the instrument annotations.

8. CONCLUSIONS

In this work we presented a convolutional neural network for note-level instrument assignment. We approach this problem as a classification task and proposed a framework that uses the pitch information of the note-events to guide the classification. Our approach can also successfully classify notes provided by a MPE algorithm, which permits generating multi-instrument transcriptions. Our method also shows the benefits of including different kernel shapes in the convolutional layers.

As future work we plan to investigate more deeply the interaction of our method with MPE algorithms as well as how the final estimations can be improved by including a clip-level analysis. The adoption of multi-label classification approaches is also planned.

9. ACKNOWLEDGEMENT

This work is supported by the European Union’s Horizon 2020 research and innovation programme under the Marie

⁴ <https://github.com/Music-and-Culture-Technology-Lab/omnizart>

Skłodowska-Curie grant agreement No. 765068 (MIP-Frontiers).

10. REFERENCES

- [1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [2] Z. Duan, J. Han, and B. Pardo, "Multi-pitch streaming of harmonic sound mixtures," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 22, no. 1, pp. 138–150, 2014.
- [3] Y.-N. Hung and Y.-H. Yang, "Frame-level instrument recognition by timbre and pitch," in *International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, Sep 2018.
- [4] Y.-N. Hung, Y.-A. Chen, and Y.-H. Yang, "Multitask learning for frame-level instrument recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019.
- [5] Y. Han, J. Kim, and K. Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 25, no. 1, pp. 208–221, 2017.
- [6] S. Gururani, M. Sharma, and A. Lerch, "An attention mechanism for musical instrument recognition," in *International Society for Music Information Retrieval Conference (ISMIR)*, Delft, Netherlands, Nov 2019.
- [7] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning Features of Music from Scratch," in *International Conference on Learning Representations (ICLR)*, New Orleans, USA, May 2017.
- [8] C. Thomé and S. Ahlback, "Polyphonic pitch detection with convolutional recurrent neural networks," in *Music Information Retrieval Evaluation eXchange (MIREX)*, 2017.
- [9] Y.-T. Wu, B. Chen, and L. Su, "Polyphonic music transcription with semantic segmentation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019.
- [10] J. Pons, T. Lidy, and X. Serra, "Experimenting with musically motivated convolutional neural networks," in *International Workshop on Content-Based Multimedia Indexing (CBMI)*, Bucharest, Romania, Jun 2016.
- [11] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *European Signal Processing Conference (EUSIPCO)*, Kos Island, Greece, Aug 2017.
- [12] C. Lordelo, E. Benetos, S. Dixon, and S. Ahlback, "Investigating kernel shapes and skip connections for deep learning-based harmonic-percussive separation," in *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, USA, Oct 2019.
- [13] Y. Wu, B. Chen, and L. Su, "Multi-instrument automatic music transcription with self-attention-based instance segmentation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, pp. 2796–2809, 2020.
- [14] A. Solanki and S. Pandey, "Music instrument recognition using deep convolutional neural networks," *International Journal of Information Technology*, pp. 1–10, Jan 2019.
- [15] E. Benetos and S. Dixon, "Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model," *Journal of the Acoustical Society of America*, vol. 133, no. 3, pp. 1727–1741, 2013.
- [16] V. Arora and L. Behera, "Multiple f0 estimation and source clustering of polyphonic music audio using PLCA and HMRFs," *IEEE/ACM Transactions on Audio Speech and Language Processing (TASLP)*, vol. 23, pp. 278–287, 2015.
- [17] K. Tanaka, T. Nakatsuka, R. Nishikimi, K. Yoshii, and S. Morishima, "Multi-instrument music transcription based on deep spherical clustering of spectrograms and pitchgrams," in *International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, Canada, Oct 2020.
- [18] R. Bittner, B. Mcfee, and J. Bello, "Multitask learning for fundamental frequency estimation in music," *arXiv e-prints*, pp. 1–13, 2018.
- [19] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, May 2020.
- [20] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music," in *International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, Oct 2017.
- [21] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band DenseNets for audio source separation," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, USA, Oct 2017.
- [22] G. Huang, Z. Liu, L. Van Der Maaten, and K. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, Jul 2017.

SpecTNT: A TIME-FREQUENCY TRANSFORMER FOR MUSIC AUDIO

Wei-Tsung Lu¹ Ju-Chiang Wang¹ Minz Won^{1,2} Keunwoo Choi¹ Xuchen Song¹

¹ ByteDance, Mountain View, California, United States

² Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain

{weitsung.lu, ju-chiang.wang, minzwon, keunwoo.choi, xuchen.song}@bytedance.com

ABSTRACT

Transformers have drawn attention in the MIR field for their remarkable performance shown in natural language processing and computer vision. However, prior works in the audio processing domain mostly use Transformer as a temporal feature aggregator that acts similar to RNNs. In this paper, we propose SpecTNT, a Transformer-based architecture to model both spectral and temporal sequences of an input time-frequency representation. Specifically, we introduce a novel variant of the Transformer-in-Transformer (TNT) architecture. In each SpecTNT block, a spectral Transformer extracts frequency-related features into the frequency class token (FCT) for each frame. Later, the FCTs are linearly projected and added to the temporal embeddings (TEs), which aggregate useful information from the FCTs. Then, a temporal Transformer processes the TEs to exchange information across the time axis. By stacking the SpecTNT blocks, we build the SpecTNT model to learn the representation for music signals. In experiments, SpecTNT demonstrates state-of-the-art performance in music tagging and vocal melody extraction, and shows competitive performance for chord recognition. The effectiveness of SpecTNT and other design choices are further examined through ablation studies.

1. INTRODUCTION

Deep learning models have been actively used in recent music information retrieval (MIR) research. Although the spirit of deep learning is end-to-end learning, however, various assumptions are made during making design choices of deep learning models.

Regarding assumptions on spectrograms, the most popular form of music audio representation in deep learning, the time-axis is often considered to be the axis of *sequence* while the frequency-axis is the axis of *feature*. For example, in [1, 2], recurrent layers were applied to model a spectrogram as a sequence of spectra. In [3], convolutional layers were used to aggregate features over time after the

first convolutional layer models multiple frames of spectra as feature. On the other hand, in [4], two-dimensional convolutional layers were used, equating the frequency- and time-axes. There are also hybrid approaches, such as convolutional recurrent neural networks (CRNN) [5] and convolutional Transformer [6], in which recurrent layers or Transformer are applied along the time axis.

In spectrograms, it is well known that there are meaningful spectral patterns. Different music components exist in different frequency ranges, and there is a very strong spectral correlation called harmonics. Since a normal convolutional layer can model local patterns only, several approaches have been proposed to model harmonics along the frequency axis. Harmonic Constant-Q Transform (HCQT) is a novel multi-channel time-frequency representation that was proposed to overcome the limitation by improving the input representation of audio [7]. Harmonic CNNs (convolutional neural networks) [8] are designed to model the harmonic pattern by modifying the convolutional filters. However, these solutions only model some of the spectral patterns, reminding the need for a more general solution with higher flexibility.

Transformers have successfully demonstrated their ability to model the sequential data with long-term (inter-) dependency and invariance. This is achieved by multiple aspects of Transformers. First, the key-query mechanism enables modeling the relationship of every combination of the instances. Second, positional encoding helps the model to take the order of instances into account. Through stacked attention layers, the input sequence is transformed into a sequence of representations that are based on the inter-dependency of the input. The prior works in audio analysis are mostly based on a similar, naive approach where Transformer is used as a temporal feature aggregator that acts similar to RNNs (recurrent neural networks), with few exceptions such as [9].

Recently, Transformer in Transformer (TNT), a variant of Transformer that arranges two Transformers in a hierarchical manner, was proposed [10] for image recognition. In TNT, an inner (lower-level) Transformer is applied to extract the local pixel-level embeddings, and then the pixel-level embeddings are projected to the patch-level embedding space which is later handled by an outer (higher-level) Transformer to summarize a global representation. One can simply apply TNT for audio by treating a song as an image, which is comprised of a sequence of frames (patches) while the frequency bins within frames are con-



sidered as pixels. However, from our pilot study, we find this approach results in unstable training and only achieves similar performance compared to using the original Transformer. This is possibly because the amount of training data is insufficient or the interpretation of frequency sequences is different from that of pixel sequences. Therefore, non-trivial modifications from the original idea of TNT should be made.

In this paper, we propose *SpecTNT*, a time-frequency transformer that models spectrograms as a sequence along both time- and frequency-axes. Similar to TNT, SpecTNT uses two Transformers hierarchically. However, the temporal local embeddings extracted from the inner Transformer are not directly sent to the outer Transformer. Instead, a special token called *frequency class token* (FCT) is appended to aggregate the important spectral features of each frame. The FCT is then projected to the global (temporal) embedding space to enable the information exchange across the time axis. This design allows the important local information is passed to the outer Transformer through FCT while reducing the dimensionality of the data flow compared to the original TNT. As a result, it helps SpecTNTs to perform well on audio-related tasks even with smaller datasets.

Our contributions can be summarized as follows: (1) to the best of our knowledge, our work is the first attempt to leverage TNT-based architecture to learn the representations for audio; (2) we propose SpecTNT, a novel modification of TNT to better fit the music data for MIR tasks; (3) we conduct extensive experiments to demonstrate the capability of SpecTNT in various MIR tasks – vocal melody extraction, music auto-tagging, and chord recognition.

2. RELATED WORK

In this section, we review the literature in music tagging, vocal melody extraction, and chord recognition – three well-defined MIR tasks adopted in the experiments to evaluate SpecTNT. Due to space limitation, we focus on the recent trends since the adoption of deep learning approaches.

Music tagging is a multi-label classification task that annotates a music audio clip with various types of labels such as genres (rock, jazz), instruments (vocal, guitar, drums), and mood (happy, sad) [11]. Since a CNN-based approach has been first introduced [3], various advanced architectures have been used including a two-dimensional CNN [4], a sample-level CNN [12], and a two-dimensional ResNet [13]. Due to the open nature of the tag set, among MIR tasks, music tagging is relatively a *vague* task – The exact mechanism of annotating tags is not fully known. This aspect suits well for the fundamental motivation of deep learning, which is, to reduce inductive bias and let the data speak [14].

The goal of vocal melody extraction is to estimate the F0 frequency of the (dominant) vocal track in given mixtures. Various deep learning methods have been adopted: a fully-connected neural network with Hidden Markov Model [15], a bidirectional long short-term memory network [1], a CNN [7], encoder-decoder networks [16, 17]



Figure 1. The block diagram of the whole SpecTNT. The details of positional encoding and SpecTNT module are illustrated in Figure 2 and 3, respectively.

and a CRNN [18]. Recently, a frequency-temporal attention module was introduced in [19] to learn the relevant regions for predictions. Some special representations are proposed including HCQT [7], a combination of frequency and periodicity [20], and source-separated tracks [21, 22].

Chord recognition is a MIR task to “produce a time-varying symbolic representation of the signal in terms of chord labels” [23]. Compared to music tagging, we clearly understand how chords of music signals can be decided – They are based on the combination of the present musical notes. Therefore, models have been designed to take advantage of note representations such as constant-Q transform (CQT) or chromagram. The early deep learning-based chord recognition models are based on a RNN [24] and a CNN [25]. Later, a CRNN has been used in [23] to combine the merits of RNNs and CNNs. More recently, (bi-directional) Transformer was used, achieving state-of-the-art performance [26, 27].

3. METHODS

As illustrated in Figure 1, the proposed SpecTNT architecture consists of a convolutional module, positional encoding, SpecTNT module, and output module.

The input time-frequency representation is first processed with a stack of convolutional layers for local feature aggregation. Then, the positional information is added to the data. In the SpecTNT module, the intermediate representation is fed into a stack of SpecTNT blocks. Lastly, the output module projects the final embedding into the desired dimension for different tasks. We detail each module in the following subsections.

3.1 Convolutional module

The purpose of this convolutional module is to employ different strategies for generating intermediate representations with pooling or striding convolution techniques depending on the nature of the task. Let the input time-frequency representation be $S \in \mathbb{R}^{T \times F \times K}$ where T is the number of time-steps, F is the number of frequency bins, and K is the number of channels. S is first passed into a stack of convolutional layers. We utilize the residual unit proposed in [28] to be the basic building block of the convolutional module. The representation after the convolutional module is denoted as $S' = [S'_1, S'_2, \dots, S'_T] \in \mathbb{R}^{\hat{T} \times \hat{F} \times \hat{K}}$, where \hat{F} , \hat{T} , and \hat{K} are the numbers of frequency bins, time-steps, and channels, respectively.

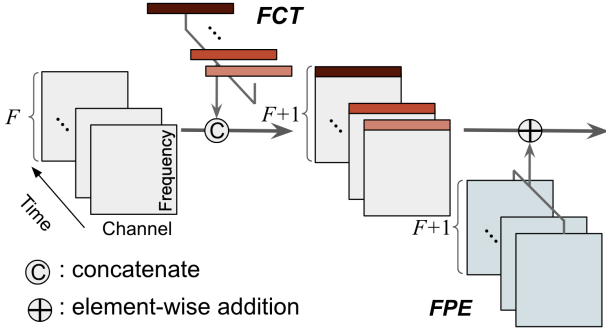


Figure 2. An illustration of the application of Frequency class token (FCT) and frequency positional encoding (FPE). F refers to the number of frequency bins of the input time-frequency representation.

3.2 Frequency Class Token

As depicted in Figure 2, *Frequency class token* (FCT) is an embedding vector initialized with all zeros to serve as the placeholder and defined as $c_t = \mathbf{0}^{1 \times \hat{K}}$. Let $S'_t \in \mathbb{R}^{\hat{F} \times \hat{K}}$ denote the input data at each time-step t . The input data and FCT are concatenated as following:

$$S''_t = \text{Concat}[c_t, S'_t]. \quad (1)$$

Here, the role of FCT c_t is similar to the classification token [29]. It is expected to extract spectral features from each frequency bin of the t -th frame during the spectral self-attention in the later stages.

3.3 Positional encoding

In the original Transformer paper, a sinusoidal positional encoding was added to the input sequence to make the following layers aware of the order of input elements [30]. From a similar motivation, we adopt a learnable positional embedding to encode the sequence order of frequency bins.

We encode the positional information of frequencies by adding the frequency positional embedding (FPE) to the data S'' . FPE is a learnable matrix $E^\phi \in \mathbb{R}^{(\hat{F}+1) \times \hat{K}}$. The addition process is done at each time-step t :

$$\hat{S}_t = S''_t \oplus E^\phi, \quad (2)$$

where \oplus is the element-wise addition, and the resulting FCTs are denoted by $\hat{C} = [\hat{c}_1, \hat{c}_2, \dots, \hat{c}_{\hat{T}}]$. Then, the resulting representation \hat{S}_t is able to carry information about pitch and timbre to the following attention layers. For example, a pitch in the signal can lead to high energy at a specific frequency bin, and the positional embedding makes FCT aware of the position of that frequency

3.4 Transformer in Transformer (TNT)

Inspired by the architecture in [10], we design a SpecTNT block to handle audio data, as depicted in Figure 3. The SpecTNT block holds two data flows: *spectral embedding* (SE) and *temporal embedding* (TE). The two data flows are respectively processed with two Transformer encoders,

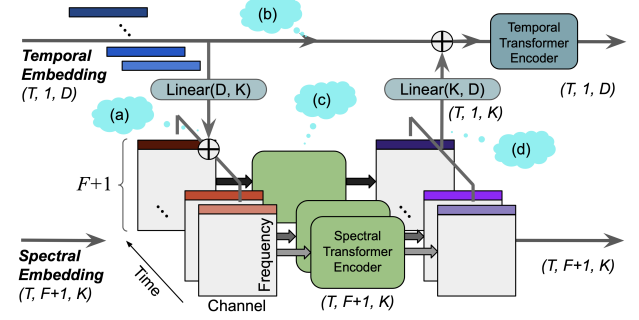


Figure 3. The block diagram of a SpecTNT block. Tensors and modules are illustrated with non-rounded and rounded rectangles, respectively. We specify the non-batch shape of tensors for clarity, and explain (a) – (d) in the main text.

namely *temporal Transformer* and *spectral Transformer*. Because the SpecTNT block is repeated multiple times in the SpecTNT module, we introduce a notation l to specify the layer index for both SE and TE.

In the following sections, we explain each component of a SpecTNT block (Section 3.4.1 through Section 3.4.3) and the entire procedure (Section 3.4.4).

3.4.1 Temporal Embedding

In the proposed model, we introduce the temporal embedding (TE) to distribute the information of FCTs across the time axis. We can write the TE at layer l as:

$$E^l = [e_1^l, e_2^l, \dots, e_{\hat{T}}^l], \quad (3)$$

where $e_t^l \in \mathbb{R}^{1 \times D}$ is a TE vector at time t and D is the number of features. In practice, TE is a learnable matrix and is initialized randomly as $E^0 \in \mathbb{R}^{\hat{T} \times D}$ prior to entering the first SpecTNT block.

There are two bridges between the spectral and temporal data flows. We use FCTs, the first frequency bin of SEs, for this communication. First, TE sends information to FCTs by passing e_t^l to a linear projection layer. Then, the projected D -dimensional vectors are added to FCTs (Figure 3-(a)). Second, after spectral transformer encoder (Figure 3-(c)), FCTs (purple arrays) are projected back to K -dimension (Figure 3-(d)). Note that TE also has a skip-connection (Figure 3-(b)).

3.4.2 Spectral Embedding

The output from the positional encoding, \hat{S} , will serve as an input SE for the first SpecTNT block and is denoted as \hat{S}^0 . As mentioned above, SE includes FCTs, which help aggregate useful spectral information from the local. As a general notation, we write the data flow of SE as:

$$\hat{S}^l = \left[[c_1^l, \hat{S}_1^l], [c_2^l, \hat{S}_2^l], \dots, [c_{\hat{T}}^l, \hat{S}_{\hat{T}}^l] \right], \quad (4)$$

where $l = 0, 1, \dots, L$, and c_t^l and \hat{S}_t^l are respectively the FCTs of l -th layer and spectral data at time-step t . Then, SE can interact with the TE through FCTs, so the local spectral features can be processed in a temporal and global manner.

3.4.3 Transformer Encoder

A Transformer encoder is composed of three components: multi-head self-attention (MHSA), feed-forward network (FFN), and layer normalization (LN).

Self-attention (SA) [30] plays the pivotal role in a Transformer encoder. It takes three inputs: $Q \in \mathbb{R}^{T \times d_q}$, $K \in \mathbb{R}^{T \times d_k}$ and $V \in \mathbb{R}^{T \times d_v}$ which represent the queries, keys, and values, respectively. T is the number of time-steps, and d_q , d_k and d_v indicate the dimension of features for Q , K , and V , respectively. The output is the weighted sum over the values based on the dot product similarity between queries and keys at the corresponding time-step.

The MHSA module [30] is an extension of SA. It splits the three inputs Q , K and V along their feature dimension into h number of ‘‘heads’’ and performs multiple SA’s, each on a head, in parallel. The outputs of heads are then concatenated and linearly projected into the final output. The FFN module has two linear layers with a GELU activation function in the middle. We also adopt the pre-norm residual units [31] to stabilize the training.

With the three components, the Transformer encoder (either spectral or temporal) is built and denoted by

$$X_l = \text{Enc}(X_{l-1}), \quad (5)$$

where the operations within it can be written as

$$\begin{aligned} X'_{l-1} &= X_{l-1} + \text{MHSA}(\text{LN}(X_{l-1})), \\ X_l &= X'_{l-1} + \text{FFN}(\text{LN}(X'_{l-1})). \end{aligned} \quad (6)$$

3.4.4 Stacking SpecTNT Blocks

We stack three SpecTNT blocks for the SpecTNT module. The module starts with inputting the initial SE, \hat{S}^0 , and the initial TE, E^0 , to the first SpecTNT block.

For a SpecTNT block, there are four steps. First, each FCT vector in \hat{S}^{l-1} is updated by adding the linear projection of the associated TE vector (Figure 3-(a)):

$$\hat{c}_t^{l-1} = \hat{c}_t^{l-1} \oplus \text{Linear}(e_t^{l-1}), \quad (7)$$

where $\text{Linear}(\cdot)$ is a shared linear layer. Second, the SE \hat{S}^{l-1} (with the updated FCTs $[\hat{c}_t^{l-1}]_{t=1}^{\hat{T}}$ at the first row) is passed through the spectral Transformer (Figure 3-(c)):

$$\hat{S}^l = \text{SpecEnc}(\hat{S}^{l-1}). \quad (8)$$

Third, each FCT vector in \hat{S}^l is linearly projected and added back to the corresponding TE vector (Figure 3-(d)):

$$\tilde{e}_t^{l-1} = e_t^{l-1} \oplus \text{Linear}(\hat{c}_t^l). \quad (9)$$

Finally, we propose to encode only the updated TE (i.e., $\tilde{E}^{l-1} = [\tilde{e}_t^{l-1}]_{t=1}^{\hat{T}}$), instead of TE + SE, with the temporal Transformer:

$$E^l = \text{TempEnc}(\tilde{E}^{l-1}). \quad (10)$$

This operation builds up the relationship along the time axis and is the key role that leads to better model and data efficiency. We consider the temporal Transformer only needs to see the information of the frequency bins which are attended by the FCT and such design largely reduces the size of the model and also improves the performance on smaller datasets in preliminary experiments.

Task	(p_f, p_t)	(k, d)	(h_k, h_d)	o_d
Music tagging	(1, 4)	(96, 96)	(4, 8)	50
Vocal melody extraction	(4, 1)	(128, 128)	(8, 8)	(T , 481)
Chord recognition	(1, 1)	(64, 256)	(4, 8)	(T , 25)

Table 1. Settings of SpecTNT for different tasks, where p_f and p_t represent the pooling ratio we apply along the frequency and time axis in the convolutional module, k and d are the feature dimension, h_k and h_d denotes the number of heads for the spectral and temporal transformer encoder respectively, and finally, o_d represents the output dimension, T indicates frame-wise predictions.

3.5 Output Module

The output TE of the 3rd SpecTNT block, E^3 , can be used towards the final output. For frame-wise prediction tasks such as vocal melody extraction and chord recognition, we feed each TE vector e_t^3 into a shared fully-connected layer with sigmoid or softmax function for final output. For song-level prediction tasks such as music tagging, we initialize a temporal class token ϵ^l ($l = 0$) concatenated at the front of E^l :

$$\hat{E}^l = [\epsilon^l, e_1^l, e_2^l, \dots, e_{\hat{T}}^l], \quad (11)$$

Note that ϵ^l does not have an associated FCT in SE, but is for aggregating TE vectors along the time axis. Finally, we feed ϵ^3 to a fully-connected layer, followed by a sigmoid layer, to get the probability output.

4. EXPERIMENTS

In this section, we evaluate SpecTNT on various types of MIR tasks to demonstrate its effectiveness and versatility. We choose three MIR tasks – music tagging, vocal melody extraction, and chord recognition.

4.1 Implementation

SpecTNT is implemented using Pytorch [32]. Due to the difference in dataset sizes and the natures of tasks, we use different hyper-parameters for the tasks as shown in Table 1. All models include dropout with a rate of 0.15 in the Transformers of the TNT modules. We use AdamW [33] as the learning optimizer. The initial learning rates are set to 10^{-3} for vocal melody extraction, 5×10^{-4} for music tagging and chord recognition, and a weight decay of 5×10^{-3} is set for all the tasks.

For the input representation of music tagging, we re-sample the audio at the 22,050 Hz and use an input length of 4.54 second. Log-magnitude mel-spectrograms are computed with 128 mel filter banks, 1024 samples of Hann window, and a hop size of 512 samples. For vocal melody extraction, input waveforms are re-sampled at the 16,000 Hz sample rate. We take 3-second segments input and their log-magnitude spectrograms are computed with 2048 samples of Hann window and a hop size of 320 samples. For chord recognition, we try two types of input representation. The first input type is 24-dimensional chroma features with a frame rate of 46 ms [34]. Out of the whole

track, we use 400 frames as an input. The second input type is CQT, which is computed from a 18.2 second audio at the 22,050 Hz sample rate. The CQT includes six octaves starting from C1 (32.70 Hz) with 24 bins per octave, and is based on a hop size of 2048.

4.2 Ablations Study

To validate the design choices we make, we consider three various models by progressively removing the components of SpecTNT as follows.

A1: Remove the operation of (a) in Figure 1 (i.e., Eq. 7) and initialize the FCTs as learnable vectors.

A2: Neglect the FCTs but use the full spectral embeddings for operations (a) and (c) in Figure 1 (i.e., Eq. 7 and Eq. 9). The resulting model can be seen as using the original TNT block [10].

A3: Remove the data flow of spectral embedding, so the model is reduced to the original Transformer [30] for aggregating the input sequence in a traditional way.

In the following evaluations of different tasks, we will include the results of the three variants for comparison.

4.3 Music Auto-tagging

Datasets Million song dataset (MSD) [35] consists of one million audio previews and a subset of it has crowd-sourced music tags. Typically, a subset with the 50 most frequent tags are used with randomly split train/validation/test sets [4]. However, these tags are noisy and the random split without considering artist overlaps may cause unintended information leakage. Therefore, we take advantage of manually cleaned 50 tags from a previous work [36] and split the dataset based on artist names so that there is no overlapped artists among the training/validation/test sets. As a result, we use 233,147 tracks, of which 70%, 15%, and 15% are allocated for training, validation, and test sets, respectively. During training, we apply random data augmentation to the input waveform following the pipeline introduced in [37].

Baseline Models Two baselines methods are compared. The first is CNNSA [6], which employs a convolutional front-end and a transformer encoder to aggregate the temporal feature. The second baseline [13] uses 7-layer short-chunk CNN with residual connection, followed by a fully-connect layer for final output. This model has shown state-of-the-art performance in music auto-tagging. We utilize the original implementation of [13] to train the baseline under the same configuration as our proposed model.

Evaluation Metrics Area Under Precision Recall Curve (PR-AUC) and Area Under Receiver Operating Characteristic curve (ROC-AUC) are used.

Results The results of music auto-tagging are summarized in Table 2. SpecTNT outperforms prior state-of-the-art models in both metrics. In the ablation study, A1 performs the worst, while A2 and A3 show similar results to SpecTNT. This can be explained from the perspective of data distribution: the top 50 tags of MSD dataset

Method	ROC-AUC	PR-AUC
Short-chunk CNN + Res	91.55	37.08
CNNSA	91.57	37.09
SpecTNT	92.08	38.62
A1	91.92	37.85
A2	92.07	38.59
A3	92.06	38.46

Table 2. Results (in %) for automatic music tagging.

are mostly related to genre and style, both of which need enough temporal information to characterize. In A1, the process of updating FCTs with TEs is removed and this may interfere the temporal information flow being shared with the spectral data and cause the performance drop. By looking into the precision scores of individual tags where SpecTNT outperforms A3, we observe that instrumental tags such as “piano” and “guitar” can benefit from SpecTNT, because they may require more spectral information to model well. This shows the benefit of adding the spectral transformer. Also, the smaller performance difference among SpecTNT, A2, and A3 indicates that the size of MSD dataset might be enough to support architectures with less prior knowledge. That is, A2 and A3 are able to sufficiently learn from MSD the useful information without further utilizing FCTs to interact with the temporal embeddings.

4.4 Vocal Melody Extraction

Datasets We use two datasets to train the models: MIR1K [38], which includes 1000 Chinese karaoke clips, and a 48-song subset of MedleyDB [39] that includes vocal tracks. Since the training sets are relative small, we adopt a pipeline with four steps of augmentation techniques. There is a chance for each step to be applied to a training sample: *i*) pitch-shifting by up to ± 2 semitones (with 100% chance), *ii*) replacing the original background track with a randomly selected, different background track (with 50% chance), *iii*) changing the gain of the vocal within $[-4, 2]$ dB (with 100 % chance), and *iv*) completely removing the background track (with 10% chance).

We choose three test sets for evaluation: ADC2004, MIREX05, and MedleyDB. For ADC2004 and MIREX05, we only use the samples that have melody sung by human voice. This results in 12 samples from ADC2004 and 9 samples from MIREX05. For MedleyDB, we only use the songs that have singing voice included in their “MELODY2” annotations, yielding 12 songs. The ground-truth pitches are obtained from the MELODY2 annotations within the intervals marked as “female singer” or “male singer.” These 12 songs are not included in training.

Baseline Models We compare our model with two baseline models. The first baseline is the joint detection and classification model (JDC) [18] based on CRNN. We use the most representative architecture, called “Main” in [18]. The second baseline is the frequency-temporal attention

Dataset	ADC2004			MIREX05			MedelyDB		
	OA	RPA	VR	OA	RPA	VR	OA	RPA	VR
JDC	71.2	68.1	73.1	86.0	80.7	85.8	77.0	64.8	73.9
FTANet	71.2	69.3	72.9	89.9	86.5	91.2	79.4	66.0	72.0
SpecTNT	85.3	85.0	88.3	91.7	90.4	95.2	78.4	77.9	87.4
A1	84.8	84.2	89.1	90.2	88.3	94.1	77.9	75.0	85.4
A2	84.9	84.5	88.3	89.7	87.7	93.1	79.3	75.6	84.0
A3	84.5	83.7	87.2	88.9	87.2	92.0	74.5	72.7	83.1

Table 3. Results (in %) for vocal melody extraction.

network (FTANet) [19], which is a CNN-based model that employs attention mechanism along the frequency and time axis. We re-implemented JDC and FTANet using Pytorch and used the suggested hyper-parameters in [18, 19]. Both models are trained under the same configuration (e.g., data split and augmentation process) as our model.

Evaluation Metrics Overall Accuracy (OA), Raw Pitch Accuracy (RPA), and Voice Recall (VR) are adopted for evaluation. We use mir_eval library [40] to compute the performance values with a tolerance range of 50 cents.

Results Table 3 shows the results for vocal melody extraction. SpecTNT outperforms the baselines by a large margin in terms of RPA and VR. To the best of our knowledge, this is the first attempt to apply Transformers to this task and the results demonstrate its superiority over the CNN and CRNN counterparts. It is worth noting that FTANet is trained with an input representation specifically designed for pitch detection [20], but our model works well with spectrogram input. In addition, A3 shows the largest performance drop, and this demonstrates the usefulness of spectral Transformer when training on smaller data.

4.5 Chord Recognition

Datasets We use the Billboard dataset to evaluate SpecTNT for the chord recognition task. The dataset contains 890 pieces selected from the Billboard chart slots [34]. Following [27], duplicates pieces are first removed to leave 739 unique pieces in total. The official release of the dataset only comes with 24-D chroma vectors, which might be insufficient to fully demonstrate the effectiveness of SpecTNT. Therefore, we manually collected the audio files based on the provided meta-data. Due to the potential version mismatch between our audio files and that for official chord annotations, we applied dynamic time warping (DTW) [41] to validate each song. Specifically, we first replicated the chroma features of the official release using Sonic Annotator [42] on our audio files, and then calculated the alignment cost between the two versions of chromagrams for each song using DTW. We selected 462 songs with the lowest alignment costs. The songs with ID’s smaller than 1000 are used for training and the remaining for testing. To augment the training data (chroma and audio), we shifted the pitches by up to ± 6 semitones. For evaluation, we adopt the “maj/min” label set with 25 classes, where 24 are major and minor triads across the 12

Method	Chroma	CQT
CR2	78.92	73.38
BTC	77.98	73.92
HT	82.68	-
SpecTNT	80.47	75.62
A1	80.10	74.83
A2	78.76	74.44
A3	77.69	74.99

Table 4. Results (in %) for chord recognition task

semitones plus an additional “no chord” class.

Baseline Models We compare to three baseline models: *i)* CR2 model from [23], which is a CRNN-based model, *ii)* a bi-directional Transformer (BTC) [26], and *iii)* Harmony Transformer (HT) [27]. BTC and HT are known to be the current state-of-the-art models for chord recognition. For CR2 and BTC, we use the official implementations with the suggested default settings for both chromagram and CQT inputs. For HT, we report the chromagram-based results in [27], since the train/test data split and data augmentation are very similar to us. We did not conduct experiments using HT with CQT input because non-trivial modifications are required for the model.

Evaluation Metrics The Weighted Chord Symbol Recall (WCSR) score is reported as evaluation metric. WCSR is the percentage of correctly identified frames and can be computed by $\frac{t_c}{t_a} \times 100(\%)$, where t_c is the duration of the correctly predicted segments, and t_a is the total duration of the test segments.

Results Table 4 shows the results for chord recognition. For “Chroma” case, the full Billboard dataset is used. For “CQT” case, the 462 songs with audio are used. From the results, SpecTNT can outperform all the baselines except HT (with chromagram input). However, HT may benefit from joint training with an additional segmentation loss, so the comparison could be unfair. Compared to BTC and CR2, SpecTNT achieves better performance for both types of input. For the ablation study, since we used less data for CQT input, A2, which is the largest model, may suffer from over-fitting and thus performs the worst.

5. CONCLUSION

We proposed SpecTNT, a novel Transformer architecture that models spectrograms along both the time and frequency axes. The introduction of FCT enables effective communication between the spectral embeddings and temporal embeddings, maximizing the benefit of Transformer encoder for flexible, local, and global modeling. In experiments, SpecTNT has demonstrated state-of-the-art performance in music tagging and vocal melody extraction and shown competitive performance in chord recognition. For future work, we plan to apply SpecTNT to other MIR tasks, such as beat tracking and structure segmentation.

6. REFERENCES

- [1] F. Rigaud and M. Radenen, "Singing voice melody transcription using deep neural networks." in *17th International Society of Music Information Retrieval conference (ISMIR), New York, USA, 2016*, pp. 737–743.
- [2] S. Leglaive, R. Hennequin, and R. Badeau, "Singing voice detection with deep recurrent neural networks," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 121–125.
- [3] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6964–6968.
- [4] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *17th International Society of Music Information Retrieval conference (ISMIR), New York, USA, 2016*.
- [5] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2392–2396.
- [6] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.
- [7] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music." in *18th International Society of Music Information Retrieval conference (ISMIR), Suzhou, China, 2017*, pp. 63–70.
- [8] M. Won, S. Chun, O. Nieto, and X. Serra, "Data-driven harmonic filters for audio representation learning," *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [9] A. Zadeh, T. Ma, S. Poria, and L.-P. Morency, "Wildmix dataset and spectro-temporal transformer model for monoaural audio source separation," *arXiv:1911.09783*, 2019.
- [10] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *arXiv preprint arXiv:2103.00112*, 2021.
- [11] P. Lamere, "Social tagging and music information retrieval," *Journal of new music research*, vol. 37, no. 2, pp. 101–114, 2008.
- [12] J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," in *Sound and Music Computing Conference (SMC)*, 2017.
- [13] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," *In Proc. of Sound and Music Computing (SMC)*, 2020.
- [14] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE signal processing magazine*, vol. 36, no. 1, pp. 41–51, 2018.
- [15] S. Kum, C. Oh, and J. Nam, "Melody extraction on vocal segments using multi-column deep neural networks." in *17th International Society of Music Information Retrieval conference (ISMIR), New York, USA, 2016*, pp. 819–825.
- [16] W. T. Lu, L. Su *et al.*, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning." in *19th International Society of Music Information Retrieval conference (ISMIR), Paris, France, 2018*, pp. 521–528.
- [17] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A streamlined encoder/decoder architecture for melody extraction," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 156–160.
- [18] S. Kum and J. Nam, "Joint detection and classification of singing voice melody using convolutional recurrent neural networks," *Applied Sciences*, vol. 9, no. 7, p. 1324, 2019.
- [19] S. Yu, X. Sun, Y. Yu, and W. Li, "Frequency-temporal attention network for singing melody extraction," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021.
- [20] L. Su and Y.-H. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 10, pp. 1600–1612, 2015.
- [21] A. Jansson, R. M. Bittner, S. Ewert, and T. Weyde, "Joint singing voice separation and f0 estimation with deep u-net architectures," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.
- [22] Y. Gao, X. Zhang, and W. Li, "Vocal melody extraction via hrnet-based singing voice separation and encoder-decoder-based f0 estimation," *Electronics*, vol. 10, no. 3, p. 298, 2021.
- [23] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition." in *18th International Society of Music Information Retrieval conference (ISMIR), Suzhou, China, 2017*, pp. 188–194.

- [24] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks." in *14th International Society of Music Information Retrieval conference (ISMIR), Curitiba, Brazil*. Citeseer, 2013, pp. 335–340.
- [25] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *2012 11th International Conference on Machine Learning and Applications*, vol. 2. IEEE, 2012, pp. 357–362.
- [26] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, "A bi-directional transformer for musical chord recognition," in *20th International Society for Music Information Retrieval Conference (ISMIR), Delft, The Netherlands*, 2019.
- [27] T.-P. Chen and L. Su, "Harmony transformer: Incorporating chord segmentation into harmony recognition," *neural networks*, vol. 12, p. 15, 2019.
- [28] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.
- [29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *31st Conference on Neural Information Processing Systems*, 2017.
- [31] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 524–10 533.
- [32] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* 32. Curran Associates, Inc., 2019, pp. 8024–8035.
- [33] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.
- [34] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An expert ground truth set for audio chord recognition and music analysis." in *12th International Society of Music Information Retrieval conference (ISMIR), Miami, USA*, vol. 11, 2011, pp. 633–638.
- [35] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of International Conference on Music Information Retrieval (ISMIR)*, 2011.
- [36] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Toronto, Canada*, 2020.
- [37] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *arXiv preprint arXiv:2103.09410*, 2021.
- [38] C.-L. Hsu and J.-S. R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 2, pp. 310–319, 2009.
- [39] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *15th International Society of Music Information Retrieval conference (ISMIR), Taipei, Taiwan*, vol. 14, 2014, pp. 155–160.
- [40] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval conference (ISMIR), Taipei, Taiwan*. Citeseer, 2014.
- [41] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.
- [42] C. Cannam, M. O. Jewell, C. Rhodes, M. Sandler, and M. d’Inverno, "Linked data and you: Bringing music research software into the semantic web," *Journal of New Music Research*, vol. 39, no. 4, pp. 313–325, 2010.

AUGMENTEDNET: A ROMAN NUMERAL ANALYSIS NETWORK WITH SYNTHETIC TRAINING EXAMPLES AND ADDITIONAL TONAL TASKS

Néstor Nápoles López **Mark Gotham** **Ichiro Fujinaga**
McGill University, CIRMMT Universität des Saarlandes McGill University, CIRMMT
nestor.napoleslopez@mail.mcgill.ca mark.gotham@uni-saarland.de ichiro.fujinaga@mcgill.ca

ABSTRACT

AugmentedNet is a new convolutional recurrent neural network for predicting Roman numeral labels. The network architecture is characterized by a separate convolutional block for bass and chromagram inputs. This layout is further enhanced by using *synthetic training examples* for data augmentation, and a *greater number of tonal tasks* to solve simultaneously via multitask learning. This paper reports the improved performance achieved by combining these ideas. The *additional tonal tasks* strengthen the shared representation learned through multitask learning. The *synthetic examples*, in turn, complement key transposition, which is often the only technique used for data augmentation in similar problems related to tonal music. The name ‘AugmentedNet’ speaks to the increased number of both training examples and tonal tasks. We report on tests across six relevant and publicly available datasets: ABC, BPS, HaydnSun, TAVERN, When-in-Rome, and WTC. In our tests, our model outperforms recent methods of functional harmony, such as other convolutional neural networks and Transformer-based models. Finally, we show a new method for reconstructing the full Roman numeral label, based on common Roman numeral classes, which leads to better results compared to previous methods.

1. INTRODUCTION

Automatic Chord Recognition (ACR) has been extensively explored in the field of Music Information Retrieval (MIR). ACR systems typically seek to predict the root and quality of the chords throughout a piece of music via either an audio or a symbolic representation. A more specific type of chordal analysis, particularly relevant for Western classical music, is functional harmony. The main difference between ACR and functional harmony is that the latter requires other adjacent tasks to be solved simultaneously, notably including the detection and identification of key changes (modulations [1, 2] and tonicizations [3]).

The analytical process of functional harmony is often described through Roman numeral annotations. This an-

notation system is particularly popular in Western music theory for the analysis of ‘common-practice’ tonal music. Roman numeral annotations encode a great deal of information about tonality, in a compact syntax. For instance, an annotation like **C:vii^o/V** encodes the local key (C-major), quality of the chord (diminished triad), chord inversion (first), and any existing tonicization (G-major).

This ‘modular’ nature of Roman numeral annotations has been beneficial to MIR research. In recent years, functional harmony has been approached by dividing the main task in several sub-tasks. Thus, as a machine learning problem, functional harmony can be expressed as the task of correctly predicting sufficient sub-tasks to reconstruct the full Roman numeral label. Furthermore, recent work in the standardization and conversion of Roman numeral analyses has provided MIR researchers with a larger meta-corpus of annotations for training new models [4, 5]. Yet, despite these developments and the growing interest in the field, the performance of functional harmony models for predicting Roman numeral labels remains relatively low.

In this paper, we propose a new neural network architecture that improves the prediction of functional harmony and its relevant features. Besides the architecture itself, our model benefits from increased data augmentation (beyond key transpositions), and an additional set of output tasks that enhance the effects of multitask learning demonstrated by other researchers [6]. To facilitate the work of other researchers, we release all of our preprocessed datasets, data splits, experiment logs, and the full source code of our network in <https://github.com/napulen/AugmentedNet>.

2. RELATED WORK

For a summary of general ACR strategies, see Pauwels et al. [7]. Here we focus on prior work in the specific area of automatic functional harmonic analysis.

The first computational works on Roman numeral analysis were by Winograd [8] and Maxwell [9]. Later, the independent contributions by Temperley, Sleator, and Sapp led to the first end-to-end automatic Roman numeral analysis system: a program named *Melisma* [10–12]. Notable subsequent studies include Raphael and Stoddard [13], Illescas et al. [14], and Magalhães and de Haas [15], who proposed Hidden Markov Models (HMMs), dynamic programming, and grammar-based approaches, respectively.

More recently, deep neural networks have become the

preferred tool for approaching this problem. Chen and Su [6] were the first to introduce ‘multitask learning’ (MTL) [16] to the problem as a suitable way for the neural network to share representations between related tonal tasks. Chen and Su’s model consists of a bidirectional LSTM [17] followed by task-specific dense layers, which implement the MTL configuration. In this work, the authors also introduced the ‘Beethoven Piano Sonata Functional Harmony’ dataset for evaluating such models. The MTL layout outperformed single-task configurations, and it has continued to be the best-performing approach in subsequent deep learning studies. Recently, the same authors have adopted Transformer-based networks to deal with functional harmony and ACR [18,19]. The work with these networks has explored the capability of attention mechanisms to improve the performance of ACR, paying special consideration to chord segmentation and its evaluation.

Micchi et al. [20], in turn, proposed a convolutional recurrent neural network (CRNN). The recurrent component consists of a bidirectional GRU [21] connected to task-specific dense layers, similar to those of Chen and Su [6]. In their experiments, a DenseNet-like [22] convolutional component outperformed other configurations (e.g., dilated convolutions or a GRU with pooling). Micchi et al. also demonstrated the positive effect of using pitch spelling in the inputs and outputs. This confers at least two advantages: it provides a more informative output (e.g., not only the correct key, but the correct spelling between two enharmonic keys), and it increases the theoretical number of transpositions available for data augmentation.

Here, we propose improvements along the line of CRNNs. Due to our focus on extended data augmentation and tonal tasks, we named our network *AugmentedNet*.

3. AUGMENTEDNET

The AugmentedNet is a similar network in size and design to the one by Micchi et al. [20]. It is characterized by a different layout of the convolutional layers, a new representation of pitch spelling, and a separation of the bass and chroma inputs into independent convolutional blocks.

3.1 Inputs

Reference note per timestep. The input to the network consists of a sequence of timesteps, which are sampled from the score at symbolically regular note duration values. In this study, we use the thirty-second note (‘demisemiquaver’) as this atomic value (i.e., eight timesteps per quarter note in the score), in order to match the most fine-grained frame sampling seen in previous work. The length of the sequence is set by a fixed number of timesteps. Following Micchi et al., we set that number at 640 frames (or 80 quarter notes) per sequence example.

Bass and spelled chroma features. The representation of each timestep is conceptually the same as in Micchi et al. [20], a vector containing a spelled bass note and spelled chroma features. However, the length of our vectors is different. In the Micchi et al. representation, each timestep

has 70 features: 35 for the bass and 35 for the chroma features. We consolidate this information in 38 features: 19 for the bass, and 19 for the chroma features. The reduction in number of features is due to an alternative encoding of pitch spelling, which we explain below.

Encoding the pitch spelling. We split the representation of a pitch spelling into two components: the pitch class (0–12) and the generic note letter (A–G). Each spelled pitch thus leads to a two-hot encoded vector with 19 features (1 of 12 pitch classes, and 1 of 7 note names). This reduces the number of parameters in the network without any observable compromise in performance. Furthermore, the spelled bass and chroma inputs are connected to the network separately, in their own convolutional blocks. The input to each block is a tensor of pitch spelling sequences.

3.2 Convolutional block

Using the feature maps of previous layers as an input to a convolutional layer has proven beneficial, for instance, by strengthening feature propagation and reducing the number of parameters [22]. Moreover, DenseNet-like architectures have shown to work well for the specific task of functional harmony [20].

We follow similar methods, reusing the feature maps computed for a given timestep in subsequent convolutions. Figure 1 provides a schematic diagram of our network, with the convolutional block on the top left area of the figure. In our preliminary experiments, we noticed that different tonal tasks have different time dependencies. For example, losing information about a specific timestep often leads to poor performance in predicting the inversion, whereas losing long-term context hinders the performance in key estimation. Our architecture implicitly prioritizes short-term dependencies in the initial convolutional layers, by having more filters and covering less timesteps. Going further, the convolutions provide more context about future timesteps, but output less filters. These increments (in window size) and decrements (in number of filters) are done in powers of 2. Using six convolutional layers in each block (as shown in Figure 1), the first layer convolves a window of a single timestep (a thirty-second note), whereas the sixth layer utilizes a window of 32 timesteps (a whole note). The output shape of each block is the original length of the sequence, with 82 features per timestep.

3.3 Dense and recurrent layers

Two time-distributed dense layers are applied to the concatenated outputs of the convolutional blocks. The dense layers help to reduce the number of features before the GRU layers. These have 64 and 32 neurons, respectively.

Two bidirectional GRU [21] layers are applied after the second dense layer. Both GRU layers return outputs at every timestep. Throughout the entire network, the dimensionality of the timesteps axis remains constant. That is, our input and output sequences have the same length, and the model predicts one Roman numeral label per timestep.

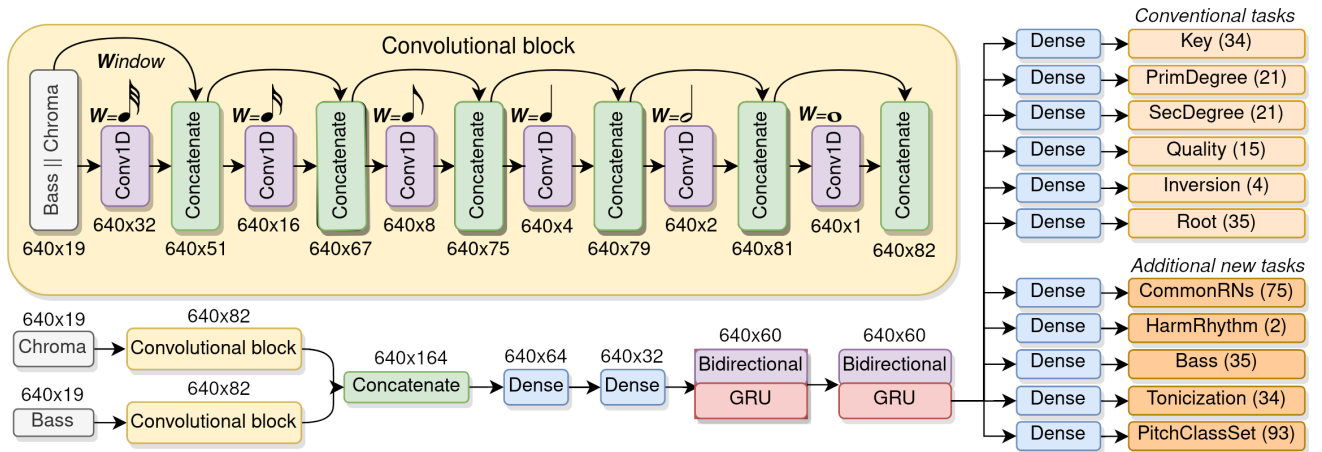


Figure 1. *AugmentedNet*. The bass and chroma inputs are processed through independent convolutional blocks and then concatenated. Both convolutional blocks are identical and expanded on the top of the figure. A convolutional block has six 1D convolutional layers. Each layer doubles the length of the convolution window and halves the number of output filters. On the right, the MTL layout with eleven tasks. Each task indicates the number of output classes in parentheses.

3.4 Multitask learning

The output of the network follows an MTL approach with hard parameter sharing, similar to the one by Chen and Su [6]. For each of the output tasks, a time-distributed dense layer is attached to the second GRU and used to predict its corresponding task. In the past, MIR researchers have reconstructed Roman numeral annotations using six tasks: the (local) key, the primary and secondary degrees, the chord quality, the chord inversion, and the chord root [6, 20]. In our network, these six ‘conventional’ tasks are learned as well, plus five new additional ones. All eleven tasks and their number of output classes are shown on the right side of Figure 1.

All the conventional tasks, except for the key, have the same number of output classes described by Michi et al. [20]. The key includes four additional classes: $\{F\flat, G\sharp, d\flat, e\sharp\}$. These were included because our dataset, larger than previous ones, revealed modulations reaching $G\sharp$ major. Thus, the number of allowed key signatures was extended by one sharp and one flat, in both modes.

3.4.1 Five additional new tasks

It is argued that MTL may improve the performance of a model by preferring representations that are useful to related tasks, acting as an implicit form of data augmentation and regularization method [16]. Roman numeral labels can be divided into different parts, of which the six conventional tasks are known examples. Motivated by the prospective improvement of our network, we included five new additional tasks, which have relevance to harmonic analysis. One of these tasks, *CommonRNs*, was used to design an alternative method to reconstruct the full Roman numeral label. The remaining four are included to strengthen the shared MTL layers. We hypothesize that these additional tasks (e.g. pitch class sets) improve the accuracy of the model because of the MTL layout, even if they are not explicitly used to predict the Roman numeral.

CommonRNs: during data exploration, we found that,

1–15	16–30	31–45	46–60	61–75
I	V/V	Ger	vii ^{o7} /v	V+
V ⁷	v	N	vii ^{o7} /iii	vii ^o /vi
V	V ⁷ /ii	vii ^{o7} /vi	IV/V	III+
i	III	V/ii	I+	V/iii
IV	ii ^{o7}	vii ^{o7}	I ⁷	ii/V
ii	iii	V ⁹	vii ^o /IV	I/bVI
vi	ii ^o	vii ^o /ii	V/III	vii ^{o7} /IV
iv	vii ^o /V	V/iv	V ⁷ /iii	V ⁷ /v
vii ^{o7}	V ⁷ /vi	Cad/V	vii ^o /iv	i ⁷
vii ^o	VII	iv ⁷	ii ^{o7}	iii ⁷
V ⁷ /V	vii ^{o7} /ii	vii ^{o7} /iv	VI ⁷	Fr
V ⁷ /IV	I/V	IV ⁷	I/III	V/IV
vii ^{o7} /V	V ⁷ /iv	V ⁷ /III	V ⁷ /VI	vii
VI	V/vi	vii ^{o7} /V	bVII	V/v
ii ⁷	vi ⁷	It	bVI	II

Table 1. *CommonRNs*. The 75 most-common Roman numeral classes found across the training set. Note that the inversion has been omitted and learned as a separate task.

when inversions were removed and synonyms (e.g., $b\mathbf{II}_6$ and \mathbf{N}_6) were standardized, a set of 75 Roman numeral classes spanned $\sim 98\%$ of all the annotations across the training set (see Table 1). This was a motivation to predict these classes directly as an additional task. The correct prediction of this new task is equivalent to predicting the primary and secondary degrees, chord root, and chord quality simultaneously. As an additional experiment (see Section 4.3), we tested an alternative new method to reconstruct the Roman numeral labels, using the key, inversion, and *CommonRNs* tasks. We refer to this method as RN_{alt} .

Harmonic Rhythm: a binary classification task that indicates whether a Roman numeral annotation starts at a given timestep. It may be relevant for chord segmentation.

Bass: a multiclass classification task that indicates the bass note in the Roman numeral label. This task has 35 output classes representing a pitch spelling, as in the chord

root task [20]. It is an alternative chord inversion task.

Tonicization: a multiclass classification task that indicates a tonicized key implied by the Roman numeral label (if any). The output classes are 34 keys, as in the key task, and it is an alternative way to learn the secondary degrees.

Pitch Class Sets: a multiclass classification task that indicates the set of pitch classes implied by the Roman numeral chord. The number of classes (93) results from computing all pitch class sets in all diatonic triad and seventh chords, plus all augmented sixth chords in all keys. This task is related to the chord quality, primary degree, and to non-chord tones [23].

3.5 Data augmentation

3.5.1 Transposition

As in most automatic tonal music analysis research, we transpose each piece to different keys as a form of data augmentation. Particularly, we transpose to all the keys that lie within a range of key signatures, in both modes. When we transpose a piece, we verify that all the modulations within the piece fall in the target range of key signatures. This process of transposition and data augmentation was introduced and described by Micchi et al. [20]. In our data exploration, we found G \sharp major to be the furthest key to the center of the *line-of-fifths* [24] in the training set. Thus, we transposed each piece across the keys with 8-flats and 8-sharps in their key signatures.

3.5.2 Synthetic data

In addition to transposition, we implemented a variation of a previous data-augmentation technique by Nápoles López and Fujinaga [25]. Starting with the Roman numeral analyses of our dataset, we synthesized ‘new’ training examples by realizing the chords implied by each Roman numeral annotation. The synthesis was done using the music21 Python library [26], which converts RomanText [4] files into scores of *block chord* realizations.

We found the default block chord texture of the synthetic examples to be only slightly beneficial for the model, possibly because it did not capture the complex texture of real keyboard music, for example. In order to account for this difference, we artificially ‘texturized’ the generated training examples, departing from the default block chords. The texturization was done by applying three note patterns recursively (see Figure 2). These patterns were designed intuitively, pursuing certain goals in the resulting texture.

Bass-split (measure 1): a pattern where the original chord duration is divided by half, playing the bass in the first half, and the remaining notes in the second. The goal is to occasionally separate the bass from all other notes.

Alberti bass (measure 2): a 4-note melodic pattern with a pitch contour of *low-high-middle-high*. The goal is to occasionally play chords using a monophonic texture.

Syncopation (measure 3): a pattern where the highest note is played first, followed by the rest of the notes, played in syncopation. The goal is to occasionally shift the onset of the bass from the onset of the Roman numeral label.

Figure 2. An example of texturization. The *block chord* texture (b) was synthesized using music21 [26] from an input RomanText file [4]. The texturized output (c) was generated by recursively applying note patterns to the block chord scores. The three musical patterns of *bass-split*, *Alberti bass*, and *syncopation* are indicated in measures 1–3, respectively. The original music score (a) is shown for reference: mm. 1–4 of Beethoven’s Piano Sonata Op.2 No.1.

Mixture (measure 4): we applied the three patterns randomly and recursively. For example, the *mixture* in measure 4 displays a *bass-split* pattern over the whole-note chord, followed by a *syncopation* pattern applied over the three upper notes, in the second half of the measure.

As part of the randomization, some chords were left unaltered (e.g., the anacrusis of Figure 2), and the patterns were applied across different duration values. To constrain the depth of the recursion, we applied these patterns only to the slices of the score that contained 3–4 simultaneous notes. This process resulted in the generation of ‘new pieces’ that showed improvements in the learning process of the model, further than the block chord synthetic scores.

4. EXPERIMENTS

4.1 Datasets

We ran experiments using six datasets: Annotated Beethoven Corpus (ABC) [27], Beethoven Piano Sonatas (BPS) [6], Haydn “Sun” Quartets (HaydnSun) [28], Theme and Variation Encodings with Roman Numerals (TAVERN) [29], When-in-Rome¹ (WiR) [4, 5], and the Well-Tempered Clavier (WTC) [4]. Table 2 shows a summary of all datasets. The summary indicates the number of files in each split and the number of sequences (each of 640 frames) that were collected from that split.

For all datasets, the same procedure was followed regarding data splits. Training, validation, and test splits were produced randomly (except in BPS, where they were provided by Chen and Su [6]). Preliminary experiments were conducted in the training set, using the validation set

¹ Note that, in practice, WiR is also a meta-collection and standardization effort, where several of these datasets (e.g., TAVERN) have been converted into a common representation. Here, we list the academic sources of the datasets. For the annotation files, please refer to the relevant literature [4, 5] as well as the accompanying source code of this paper.

Dataset	Files (Seqs)		
	Training	Validation	Test
ABC [27]	50 (448)	10 (97)	10 (99)
BPS [6]	18 (155)	7 (75)	7 (82)
HaydnSun [28]	16 (91)	4 (19)	4 (19)
TAVERN [29]	38 (404)	8 (68)	8 (64)
WiR [4, 5]	107 (301)	21 (61)	21 (51)
WTC [4]	12 (25)	6 (13)	6 (14)
Total	241 (1424)	56 (333)	56 (329)

Table 2. The functional harmony datasets used in our experiments. The splits were generated randomly (except for BPS). For each split, the number of files and the number of sequences (in parentheses) are indicated.

to assess the performance, adjust the hyperparameters, and inform the design of the network architecture. The best-performing version of our model was run once in the test set, this time including the validation portion as part of the training. The results obtained for all the rows labeled *Full dataset* in Table 4 report the results obtained on the corresponding test split.

Data augmentation. For every training example, we synthesized and texturized an additional file, using only the Roman numeral annotations (and ignoring the original score). The original and texturized training examples were transposed to different keys for further data augmentation. Both forms of data augmentation were applied to the training set of a particular experiment, leaving the validation and test sets intact, in order to prevent any data leakage.

4.2 Training procedure

Epochs. We set a fixed number of 100 epochs in all experiments. We found that the use of early stopping was unreliable to determine the end of the training process. Instead, we saved the weights after each epoch. At the end, we selected the weights that maximized the mean accuracy across the six conventional tasks.

Other hyper-parameters. Each of the layers in the network is accompanied by batch normalization [30] before the activation function. In the recurrent layers, we apply the batch normalization after the activation function. All convolutional and dense layers use the rectified linear unit (ReLU) as their activation function. However, the two GRU layers use a hyperbolic tangent. In all of our experiments, we used 16 sequences per batch and the *rmsprop* optimizer [31], with a learning rate of 10^{-3} .

Computing time. The network was trained on a personal laptop² with a Linux operating system, Tensorflow v2.4.1 [32], and GPU acceleration. With these hardware and software conditions, the training times are approximately 30 minutes (BPS only), 40 minutes (BPS+WTC), and 250 minutes (Full dataset). The number of trainable parameters in the network is close to 90,000. This number already includes all the parameters introduced by the addi-

Model	Key	Deg.	Qual.	Inv.	Root	RN
AugN ₆	82.7	64.4	76.6	77.4	82.5	43.3
AugN ₆₊	83.0	65.1	77.5	78.6	83.0	44.6
AugN ₁₁	81.3	64.2	77.2	76.1	82.9	43.1
AugN ₁₁₊	83.7	66.0	77.6	77.2	83.2	45.0

Table 3. Average accuracy (in %) of four configurations of our model, where {6, 11} indicate the number of MTL tasks and ‘+’ indicates the use of synthetic training data.

tional output tasks. Therefore, the model is similar in size to recent approaches [19, 20].

4.3 Results

AugmentedNet configurations. Table 3 summarizes the performance of different AugmentedNet configurations. For example, with or without the additional tasks, and with or without synthetic data. The configurations were trained on each of the six datasets individually, for a total of 24 experiments. The accuracy reported is the average accuracy obtained by each model configuration across all six datasets. Based on the reported accuracy values, the AugmentedNet₁₁₊ (with additional tasks and synthetic training examples) is the best-performing configuration. Thus, in subsequent experiments, we compare this configuration against the current state-of-the-art models.

In the past, functional harmony models have been evaluated using the Beethoven Piano Sonatas (BPS) dataset [6, 18, 20]. The most recent model [19] has also been evaluated using the Well-Tempered Clavier (WTC) dataset [4]. We provide direct comparisons in these two datasets, in so far as that is possible, replicating the experimental conditions of the previous models. In addition, we also report the results of our model across the remaining datasets.

Beethoven Piano Sonatas. The last rows of Table 4 show the results on the BPS dataset. Single lines in the table delimit experiments that are directly comparable. For example, the upper rows of BPS compare the results of Micchi et al. [20] using all of their available training data and our model using the larger dataset available to us now.

Well-Tempered Clavier. Above the BPS rows, in Table 4, we show the evaluation on the WTC dataset. The CS21 model presented the results over 4-fold cross validation [19]. We replicate this study for direct comparison.³ In these rows, we report the average accuracy across the four folds, as well as the standard deviation.

The results show that our model outperforms both the recent convolutional methods [20] and Transformer-based ones [19] in the reconstruction of the full Roman numeral labels. For ABC, HaydnSun, TAVERN, and WiR, we show the generalization of our model when using all the training data available on the corresponding test set. Finally, we show the overall performance of our model in a composite test set that includes all six datasets (first row of Table 4).

² Intel i7 10750h, Nvidia RTX 2070, 32 GB DDR4.

³ But we used our test split for the *Full dataset* experiment in WTC.

Test set	Training set	Model	Key	Degree	Quality	Invers.	Root	ComRN	RN _{conv}	RN _{alt}
Full test set	Full dataset	AugN	82.9	67.0	79.7	78.8	83.0	65.6	46.4	51.5
WiR	Full dataset	AugN	81.8	69.2	85.9	90.3	90.3	70.2	56.4	62.4
HaydnSun	Full dataset	AugN	81.2	62.9	80.2	82.7	86.5	60.4	48.6	52.1
ABC	Full dataset	AugN	83.6	65.6	78.0	76.9	78.9	62.6	44.5	48.4
TAVERN	Full dataset	AugN	88.7	60.0	77.4	78.8	81.5	66.3	42.6	52.9
WTC	Full dataset	AugN	77.2	69.7	75.0	74.4	82.7	61.7	46.2	47.9
WTC _{crossval}	BPS+WTC	AugN	85.1 _(4.0)	62.9 _(5.5)	69.1 _(1.9)	70.1 _(3.7)	79.2 _(1.8)	59.9 _(3.4)	42.9 _(4.2)	46.9 _(4.7)
WTC _{crossval}	BPS+WTC	CS21	56.3 _(2.5)	-	-	-	-	-	26.0 _(1.7)	-
BPS	Full dataset	AugN	85.0	73.4	79.0	73.4	84.4	68.3	45.4	49.3
BPS	All data	Mi20	82.9	68.3	76.6	72.0	-	-	42.8	-
BPS	BPS+WTC	AugN	82.9	70.9	80.7	72.0	85.3	67.6	44.1	47.5
BPS	BPS+WTC	CS21	79.0	-	-	-	-	-	41.7	-
BPS	BPS	AugN	83.0	71.2	80.3	71.1	84.1	68.5	44.0	47.4
BPS	BPS	Mi20	80.6	66.5	76.3	68.1	-	-	39.1	-
BPS	BPS	CS19	78.4	65.1	74.6	62.1	-	-	-	-
BPS	BPS	CS18	66.7	51.8	60.6	59.1	-	-	25.7	-

Table 4. Accuracy of five functional harmony models: Chen and Su (2018, 2019, and 2021), Micchi et al. (2020), and AugmentedNet₁₁₊. In the WTC test set, the comparison against CS21 replicated the 4-fold cross validation [19]. In this case, the standard deviation is indicated in parentheses. For all other rows, the results report the performance on the held test set. The values in the RN_{alt} column indicate the performance using an alternative method for reconstructing the full Roman numeral, as explained in Section 3.4.1.

The RN_{conv} and RN_{alt} methods. As discussed in Section 3.4.1, it is possible to use the 75 most-common Roman numeral classes as an alternative task to the chord root, chord quality, and primary and secondary degree tasks. Thus, an alternative resolution of the Roman numeral label (RN_{alt}) is presented in the last column of the AugmentedNet results. This accuracy corresponds to the reconstruction of the full Roman numeral using the key, chord inversion, and CommonRNs. We found that this, in fact, leads to better results compared to the conventional method (RN_{conv}), which reconstructs the full Roman numeral labels using the six conventional tasks. Table 4 shows the accuracy values for both methods. For completeness, we also show the accuracy of the CommonRNs output task. For this task, note that the maximum achievable accuracy is ~98%, because any class that is not present in the set of 75 CommonRNs will be misclassified.

In summary, the *Full dataset* rows show the best results achieved by our model for each dataset. In all cases, the results of our model are higher than existing methods in the final reconstruction of the Roman numeral label. Additionally, the best results in the reconstruction are achieved via the RN_{alt} method, instead of the conventional one (RN_{conv}).

5. CONCLUSION

We present advances in the use of CRNNs to predict Roman numeral labels. In Beethoven Piano Sonatas (BPS) and the Well-Tempered Clavier (WTC) datasets, our net-

work outperforms existing models in the prediction of the conventional tonal tasks and the reconstruction of the full Roman numeral labels. Furthermore, we demonstrate that the use of an additional task, CommonRNs, is helpful to achieve better results in the final reconstruction step, compared to the conventional method used in previous research. Although we present these general improvements in accuracy, we have not yet assessed the chord segmentation of our model, leaving that for future work. Furthermore, we consider that several ideas presented here may be useful in future automatic tonal music analysis research, notably: (1) the use of additional tonal tasks in functional harmony and, (2) the use of texturized synthetic training examples. Although five additional tasks were presented, there are more potential tasks that can be examined, such as triad vs. seventh classification, tonal function (T, D, and SD), or cadence detection. Our method for synthesizing ‘new’ training examples applied three note patterns to *texturize* block chords. We developed this method based on observation and music theory domain-knowledge. A more sophisticated approach could offer better texturization outputs. We consider this to have the most potential impact on functional harmony research, because the data is still scarce and expensive to annotate. Although current models have yet to reach the expectations of MIR researchers and musicologists alike, we hope that this goal is not too far. Through the use of new techniques, deep learning models may soon achieve unprecedented results in complex music analytical processes, such as Roman numeral analysis.

6. ACKNOWLEDGMENTS

This research has been supported by the Social Sciences and Humanities Research Council of Canada (SSHRC) and the Fonds de recherche du Québec–Société et culture (FRQSC). We would also like to thank user *ClassicMan* from the MuseScore community, who allowed us to use their MusicXML scores of all Piano Sonatas by Beethoven. This saved us a great deal of time during this research.

7. REFERENCES

- [1] L. Feisthauer, L. Bigo, M. Giraud, and F. Levé, “Estimating keys and modulations in musical pieces,” in *Proceedings of the Sound and Music Computing Conference*, 2020.
- [2] H. Schreiber, C. Weiss, and M. Müller, “Local key estimation in classical music recordings: A cross-version study on Schubert’s Winterreise,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 501–505.
- [3] N. Nápoles López, L. Feisthauer, F. Levé, and I. Fujinaga, “On local keys, modulations, and tonicizations: A dataset and methodology for evaluating changes of key,” in *Proceedings of the 7th International Conference on Digital Libraries for Musicology*, 2020, pp. 18–26.
- [4] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, “The RomanText format: A flexible and standard method for representing roman numeral analyses,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 123–129.
- [5] M. Gotham and P. Jonas, “The openscore lieder corpus,” in *Poster at the Music Encoding Conference*, 2021.
- [6] T.-P. Chen and L. Su, “Functional harmony recognition of symbolic music data with multi-task recurrent neural networks,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2018, pp. 90–97.
- [7] J. Pauwels, K. O’Hanlon, E. Gómez, and M. Sandler, “20 Years of automatic chord recognition from audio,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019.
- [8] T. Winograd, “Linguistics and the computer analysis of tonal harmony,” *Journal of Music Theory*, vol. 12, no. 1, pp. 2–49, 1968.
- [9] H. J. Maxwell, “An expert system for harmonizing analysis of tonal music,” in *Understanding Music with AI: Perspectives on Music Cognition*. Cambridge, MA, USA: MIT Press, 1992, pp. 334–353.
- [10] D. Temperley, *The Cognition of Basic Musical Structures*. MIT Press, 2004.
- [11] D. Sleator, “The melisma music analyzer,” <https://www.link.cs.cmu.edu/music-analysis/>, Accessed: 2021-07-02.
- [12] C. Sapp, “tsroot manpage,” <http://extras.humdrum.org/man/tsroot/>, Accessed: 2021-08-01.
- [13] C. Raphael and J. Stoddard, “Functional harmonic analysis using probabilistic models,” *Computer Music Journal*, vol. 28, no. 3, pp. 45–52, 2004.
- [14] P. R. Illescas, D. Rizo, and J. M. I. Quereda, “Harmonic, melodic, and functional automatic analysis,” in *Proceedings of the Sound and Music Computing Conference*, 2007.
- [15] J. P. Magalhães and W. B. de Haas, “Functional modelling of musical harmony: an experience report,” *ACM SIGPLAN Notices*, vol. 46, no. 9, pp. 156–162, 2011.
- [16] S. Ruder, “An overview of multi-task learning in deep neural networks,” *arXiv:1706.05098*, 2017.
- [17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
- [18] T.-P. Chen and L. Su, “Harmony Transformer: Incorporating chord segmentation into harmony recognition,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 259–267.
- [19] —, “Attend to chords: Improving harmonic analysis of symbolic music using Transformer-based models,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.
- [20] G. Micchi, M. Gotham, and M. Giraud, “Not all roads lead to Rome: Pitch representation and model architecture for automatic harmonic analysis,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, pp. 42–54, 2020.
- [21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv:1406.1078*, 2014.
- [22] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2261–2269.
- [23] Y. Ju, N. Condit-Schultz, C. Arthur, and I. Fujinaga, “Non-chord tone identification using deep neural networks,” in *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, 2017, pp. 13–16.
- [24] D. Temperley, “The line of fifths,” *Music Analysis*, vol. 19, no. 3, pp. 289–319, 2000.

- [25] N. Nápoles López and I. Fujinaga, “Harmonic reductions as a strategy for creative data augmentation,” in *Late-Breaking Demo at 21st International Society for Music Information Retrieval Conference*, 2020.
- [26] M. S. Cuthbert and C. Ariza, “music21: A toolkit for computer-aided musicology and symbolic music data,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2010, pp. 637–642.
- [27] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, “The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets,” *Frontiers in Digital Humanities*, vol. 5, 2018.
- [28] N. Nápoles López, “Automatic harmonic analysis of classical string quartets from symbolic score,” Master’s thesis, Universitat Pompeu Fabra, 2017.
- [29] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, “Theme and variation encodings with Roman numerals (TAVERN): A new data set for symbolic music analysis.” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2015, pp. 728–734.
- [30] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the International Conference on Machine Learning*, 2015, pp. 448–456.
- [31] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv:1609.04747*, 2016.
- [32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 2016, pp. 265–283.

MINGUS: MELODIC IMPROVISATION NEURAL GENERATOR USING SEQ2SEQ

Vincenzo Madaghiele Pasquale Lisena Raphaël Troncy
EURECOM, Sophia Antipolis, France

[vincenzo.madaghiele, pasquale.lisena, raphael.troncy]@eurecom.fr

ABSTRACT

Sequence to Sequence (Seq2Seq) approaches have shown good performances in automatic music generation. We introduce MINGUS, a Transformer-based Seq2Seq architecture for modelling and generating monophonic jazz melodic lines. MINGUS relies on two dedicated embedding models (respectively for pitch and duration) and exploits in prediction features such as chords (current and following), bass line, position inside the measure. The obtained results are comparable with the state of the art of music generation with neural models, with particularly good performances on jazz music.

1. INTRODUCTION

Natural Language Processing (NLP) techniques are achieving remarkable results when applied to MIR tasks [1]. Music can indeed be interpreted as a language, and automatic music generation has been a showcase for the NLP technologies in MIR. Among these techniques, Transformer models [2] have succeeded in complex tasks related to language understanding, overcoming the performances of more established architecture such as Recurrent Neural Networks (RNN) when huge amounts of data are available [3,4].

In this paper, we introduce MINGUS¹ (Melodic Improvisation Neural Generator Using Seq2seq), a transformer architecture for modelling and generating monophonic jazz melodic lines. MINGUS handles pitch and duration as separate features, using two distinct transformer models. In addition, it exploits the whole available information by conditioning on other musical features, such as harmonic structure and rhythmic properties. An implementation of MINGUS is available in open-source at <https://git.io/mingus>.

The remaining of this paper is structured as follows. After pointing to some related work in Section 2, we will describe MINGUS in Section 3. Section 4 reports about

¹ Named in honour of Charles Mingus (1922 – 1979), American jazz composer, double bassist and pianist.

an evaluation experiment, whose results are discussed in Section 5. In Section 6, we carried on a qualitative evaluation with a user survey. Finally, conclusions and future work are outlined in Section 7.

2. STATE OF THE ART

Data representation Musical data can be represented symbolically with different levels of abstraction and precision, involving features such as pitch and duration of the notes, relative position in the bar, harmonic structure, intensity (velocity), timbre. Different approaches have been experimented with in literature for duration representation, among which time-step encoding² [5–8], note duration encoding [5,9,10] and note beat position encoding [5,11]. The duration information can be also modelled as a sequence and independently learned [9,12].

Model architecture Multiple different models have been used for jazz music generation, among those Hidden Markov Models [13], melodic grammar learning [14] and genetic algorithms [15] have achieved notable results. In this paper, we have focused on deep learning approaches to this task. The simplest approach for monophonic music generation with neural networks is jointly learning the dependencies between features. This has been implemented in different architecture, such as RNNs [6], Generative Adversarial Networks (GAN) [5], combinations of GAN and RNN [16] or Transformer models [17].

An alternative strategy is to train separately to learn specific features of the data, then conditioning them on the other features. In [18] and [19], two LSTM models are trained separately on pitch and duration of the notes in the melodies. LSTM are also used in [9], in which different conditioning combinations – inter-conditioning between pitch and duration, chord, next chord and relative position in the bar – are compared. In **BebopNet** [12], a unique embedding representation of pitch and duration feeds a unique Transformer module. **SeqAttn** [6] obtained good performances using a modified conditioned LSTM attention unit.

Transformer-based architecture can be used for overcoming the problem of vanishing gradient of RNNs [20]. In polyphonic music generation, training transformer models on massive amounts of data produced impressive re-

² Sampling over time and using using a sustain character (s) for pitch continuation.



sults, as in OpenAI’s MuseNet [3] and Magenta’s Music transformer [4].

Evaluation Methods Evaluating the performance of a generative task is an open problem, with several metrics and methods proposed. Classical **machine-learning metrics** – **loss** and **accuracy** – are applied in evaluating music generation from a sequence-modelling point of view, measuring the capability of predicting the most probable class (e.g. pitch) given the sequence of all the previous ones. It could be argued that the purpose of generation tasks goes beyond the completely accurate prediction of the next token and so these metrics can capture only partially the quality of music generation systems. Nevertheless, they are useful to make a comparison among models.

Common metrics for generative NLP task evaluation can be used also for music generation, such as **perplexity** [21] and **BLEU** [22]. The latter in particular has been applied to music generation as a measure of similarity between two corpora of music [9]. A comprehensive evaluation of NLP metrics for music generation is performed in [23].

Other metrics have been proposed for measuring how realistic a generated melody is by comparing it with the training corpus in musical terms. In [5], the authors propose a collection of metrics, which includes counting pitch repetitions, rhythmic variations and measuring harmonic consistency. Similar features are involved in **MGEval**³, which computes the degree of similarity (KL-divergence) between two corpora of MIDI files by extracting the distribution of each metric on a reference corpus from the original data and on a corpus of generated musical sequences [24]. More metrics are proposed in [17], focusing on the structural coherence of the generated musical phrases. Other common metrics are purely music-related, such as **harmonic coherence** [5, 12], the measurement of the percentage of chord and scale tones among the generated notes. Finally, **focus groups and user surveys** have been extensively used for qualitative evaluation [7, 8].

3. APPROACH

In this section, we will describe in detail MINGUS, focusing on the strategy for data representation and its architecture.

3.1 Data representation

The required input formats are *MusicXML* or *abc* notation. We require that the chords – when available – are explicitly expressed by their signature in the right place in the measure⁴.

Each melodic line is represented as sequences of the following features – with the range of possible values reported in square brackets:

1. **Pitch (P)**: pitch of each note, as MIDI pitch number (from 0 to 127). Rests are represented with the additional character R [0-128]
2. **Duration (D)**: duration of each note [0-12]
3. **Chord (C)**: current chord in the starting beat of the note [0-128 x 4]
4. **Next Chord (NC)**: next chord in the progression [0-128 x 4]
5. **Bass (B)**: current bass in the beat the note starts on [0-128]
6. **Beat (BE)**: number of beat in the measure the note starts on [0-3]
7. **Offset (O)**: offset of the note from the start of the measure [0-95]

An example of input format for note sequences can be seen in Figure 1. The duration value D is extracted by sampling each measure into 96 equally sized parts and assigning to each note the closest duration from a dictionary of possible duration values chosen in advance; for example, the "quarter note" value is assigned to notes whose duration is closer to $d_{measure}/96 * 24$, where $d_{measure}$ is the duration of the measure in seconds. In this specific case, the choice to divide a measure into 96 equal parts allows for representation precision up to 8th note triplets and dotted 16th notes. By using a greater number of samples it would be possible to represent more precisely many different duration values. This method of time division ensures flexibility to different music styles which could require the use of more complex time divisions such as quintuplets or septuplets. Chords (C and NC) are always represented by their four fundamental notes in MIDI encoding, as already seen in [9, 12]; chords with more than four notes have been cropped, the VII degree has been added to chords with less than four notes.

Given that language models are normally trained on batches of phrases with a maximum fixed length, we included a melody segmentation strategy for dividing tracks into meaningful musical phrases. For this purpose, we consider a melodic phrase to end when a long rest – longer than a threshold r , equivalent to a quarter note triplet – is encountered or when the maximum sequence length $l = 35$ is reached. The thresholds for the long rest duration and the sequence length have been chosen experimentally. It has been found that a different choice of maximum duration, if not extreme, do not have much influence on the result, while the sequence length has a greater effect on the result. Another observation is that the long rests at the end of each segment must be included in the sequence, otherwise the model will not learn to include them. Segments shorter than 35 tokens are padded with specific "pad" tokens.

³ The MGEval toolbox – which we use in this work – is available in its original implementation at <https://git.io/mgeval>

⁴ Future work includes the possibility of automatically inferring the chords from the played notes.

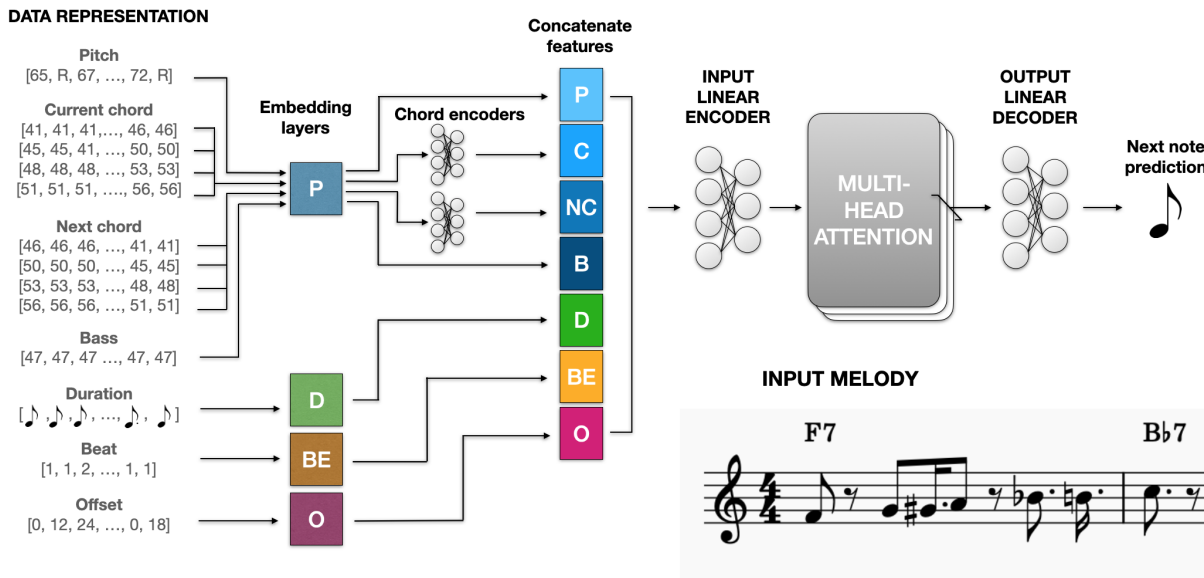


Figure 1: MINGUS model architecture and data representation. The example melodic sequence is extracted from Charlie Parker’s improvisation on Billie’s Bounce, Weimar Jazz DB

3.2 Model Architecture

MINGUS is structured as two parallel transformer models with the same structure, respectively predicting pitch and duration, composed by the sole encoder module with a forward mask and a pad mask. This structure was chosen because it allows capturing the rhythmic variation with great precision, by allowing the model to learn different embedding and weights for pitch and duration prediction. The architecture used in this experiment is shown in Figure 1. The structure is the same for pitch and duration models, the only difference is the output of the prediction, which can be either from the pitch or the duration dictionary.

Batched data is encoded with feature-specific embedding layers. Pitch-related data (melody pitch, chord pitches, next-chord pitches and bass) are encoded with a pitch embedding layer while duration, offset and beat have their own embedding dictionary. After embedding, chord pitches are grouped in a linear layer which is then combined with the other embedded features and fed into a four-layer, four-heads self-attention module.

The model was conditioned on all the features mentioned in Section 3.1. We performed an ablation study in order to understand the contribution of each feature, choosing the combination maximising the accuracy score. In particular, the optimal combination for the pitch model included features D, C, B, BE, and O, while for the duration model B, BE, and O. However, it should be noticed that the feature combination that maximises accuracy might not be the one that generates the most convincing music samples. More results about this ablation study are included in the repository.

4. EXPERIMENT

MINGUS was trained on two different datasets to evaluate its adaptability to different styles of music and compare it with other models. The **Weimar Jazz Database** (WjazzDB) [25] is a collection of annotated transcriptions of jazz solos, composed of 456 improvisations on famous jazz standards. It is a very diverse set of improvisations played on multiple instruments, including multiple jazz styles with different degrees of complexity. The dataset is complete with chords and bass information. The **Nottigham Database** (NottinghamDB) is a collection of 1034 folk songs⁵. The harmony of the music in this dataset is less complex with respect to the Weimar Jazz DB, nevertheless its smaller dimensions could be useful to show how the size of the dataset influences the generation results.

Each dataset was split into three subsets for training (70%), validation (10%) and testing (20%). The 35-token sequences were grouped into batches of 20 melodies for training and 10 melodies for validation and testing.

The network is trained for next-token prediction task using sequential information. Both pitch and duration are trained using cross-entropy loss function and Stochastic Gradient Descent optimiser. More details on the training parameters are available in the repository.

Music generation is done by sampling the trained network given an input note sequence of variable length. The input melody is split into pitch and duration sequences and each sequence is given as input to the respective trained model. The output of the model consists of the probabilities for each token in the dictionary to be the next token. The most probable token is selected and added to the

⁵ <https://github.com/jukedeck/nottingham-dataset>

original sequence, which is then given back as input to the model, together with other required features for conditioning – features 3–7 in Section 3.1. This process is repeated as many times as the number of notes to be generated, which of course must be the same for pitch and duration. After generating the new sequences of pitch and duration separately, they are combined and exported to MIDI.

5. RESULTS

This section reports the performance of MINGUS and compares it to SeqAttn [6] and BebopNet [12]. These two models have been chosen because they represent different state-of-the-art architecture for music generation. BebopNet is based on transformer and SeqAttn is a bi-directional LSTM model conditioned on chords, with different features and duration representation. To obtain comparable results MINGUS⁶, BebopNet and SeqAttn have been re-trained on the two datasets⁷, and evaluated on the same metrics.

The perplexity and accuracy of SeqAttn have been computed using the functions available in the authors’ implementation. However, the prediction of the sustain token (*s*) is considered accurate even if the note that is being sustained is not correct; similarly, (*s*) is considered as a distinct token in the computation of the perplexity. Instead, in MINGUS and BebopNet duration is represented with a separate dictionary and this allows to have note-specific perplexity and accuracy.

5.1 Perplexity

The perplexity scores of the three models computed on the test set are collected in Table 1. For MINGUS, it is reported for pitch and duration models, while for BebopNet and SeqAttn it is computed on the summed entropy of pitch and duration.

Perplexity Dataset	MINGUS		BebopNet	SeqAttn
	pitch	duration		
WjazzDB	11.01	4.14	44.70	3.71
NottinghamDB	11.03	1.88	13.46	<i>1.40</i>

Table 1: Test perplexity scores for MINGUS, BebopNet and SeqAttn. Values in *italic* are reported from the original paper

The perplexity can give a general idea of the degree of the uncertainty of the model when predicting the next token, it is however not a good indicator of music generation quality. The perplexity of all models changes according to the dataset. The greatest perplexities have been obtained

⁶ In the best configuration obtained from the ablation study

⁷ WjazzDB has been converted from the original csv format into musical formats compatible with the studied implementations, namely into musicXML – for BebopNet and MINGUS – and MIDI – for SeqAttn. 28 songs have been removed from the original dataset due to incompatibility with BebopNet, which was not recognising chords outside its internal dictionary; all models have been trained on this reduced version. The csv has been selected as starting format because it is the only one including all mentioned information (notes, chords, bass line).

for all models on the WjazzDB, despite the fact that Nottingham DB has fewer data, proving that the complexity of the music is an important factor for language modelling tasks.

5.2 Accuracy

The accuracy measures how many times the model prediction for the next note is correct. This metric could be useful to have an overall representation of the model performance but it does not guarantee a realistic music generation. The accuracy values of MINGUS and SeqAttn on all datasets are reported in Table 2. The implementation of BebopNet is not providing the computation of accuracy on a test set, therefore its scores are not shown in the table.

Accuracy [%]	MINGUS		SeqAttn
Dataset	pitch	duration	
WjazzDB	16.32	32.34	74.43
NottinghamDB	35.82	76.62	<i>90.26</i>

Table 2: Test accuracy comparison between MINGUS and SeqAttn. Values in *italic* are reported from the original paper

It is difficult to compare the accuracy results because of the different note representations and the division of pitch and duration models in MINGUS. The model achieves a higher accuracy on the duration prediction and a lower accuracy in pitch predictions; this could be due to the different size of vocabulary, but it could also be related to the difference between the two tasks, with a higher difficulty for pitch prediction. When comparing MINGUS and SeqAttn it should be considered that a percentage of the accuracy of SeqAttn is due to the prediction of common sustain tokens.

5.3 MGEval

MGEval is a collection of metrics specifically proposed for evaluation of generative music tasks [24]. For MINGUS and BebopNet, we computed MGEval metrics comparing 15 reference tunes randomly selected from the original dataset – used as reference corpus – and a set of 15 tunes, generated from the same input by each model. SeqAttn generates music from internally selected songs from the dataset seen during training, instead of accepting a track in input for triggering the generation; for this reason, MGEval metrics for SeqAttn are comparing the whole output of the model and the whole studied corpus (reference corpus).

Table 3 collects the results obtained on MINGUS, BebopNet and SeqAttn trained on WjazzDB. MGEval metrics yield very diverse results and do not reveal a clear overall winner. While the LSTM-based model (SeqAttn) has better scores on avg IOI and comparable results on pitch range and total used pitches, transformer-based ones (MINGUS and BebopNet) largely over-perform it in total pitch class histogram and note length histogram. MINGUS stands out

MGEval Measure	MINGUS		BebopNet		SeqAttn	
	KL div	overlap area	KL div	overlap area	KL div	overlap area
total used pitch	0.172	0.7959	0.007	0.539	0.068	0.735
total used note	0.071	0.678	0.046	0.794	0.169	0.239
avg IOI	0.054	0.625	0.219	0.842	0.049	0.719
avg pitch shift	0.041	0.821	0.160	0.424	-	-
note length histogram	0.283	0.507	0.054	0.821	0.241	0.468
total pitch class histogram	0.088	0.864	0.137	0.786	0.405	0.658
note length transition matrix	0.149	0.695	0.210	0.850	0.261	0.388
pitch class transition matrix	0.038	0.836	0.118	0.737	0.183	0.744
pitch range	0.037	0.844	0.093	0.571	0.062	0.702

Table 3: MGEval comparison between MINGUS, BebopNet and SeqAttn on WjazzDB

in pitch class transition matrix and total pitch class histogram, whose results are largely better than the other two models. We interpret this result with a better modelling capability for transitions between notes, maybe due to MINGUS’ flexible duration vocabulary. On the other hand, SeqAttn performs much better on NottinghamDB⁸. This suggests the duration representation employed in SeqAttn does a better job in generalising on the music style of NottinghamDB, while on WjazzDB the additional information provided in MINGUS and BebopNet improve generation quality.

5.4 Harmonic coherence

The harmonic coherence measures how many notes of each solo are coherent to the related harmonic context. It is defined here as the percentage of generated notes that are tones belonging to the current chord, or to the scale associated with it. These metrics have been calculated on the generated tracks and on the entire original dataset. The results are reported in Table 4.

Harmonic coherence [%]	Chord	Scale
Original	49.17	72.16
MINGUS	51.81	77.49
BebopNet	40.66	64.55
SeqAttn	35.92	60.26

Table 4: Harmonic coherence on WjazzDB

These results confirm that MINGUS generated melodies tend to have greater harmonic coherence than other models, with BebopNet obtaining slightly worse performance and SeqAttn being less good on this metric. We may conclude that the conditional LSTM module proposed in SeqAttn is less able to capture the complex relationship between chords and melody with respect to Transformer-based architectures. Another reason may be identified in the presence of additional features such as duration and offset – in both MINGUS and BebopNet – which are beneficial for the harmonic coherence.

⁸ Results provided in the repository.

However, MINGUS generations are more harmonically coherent than the original dataset. We can claim here the model has been able to capture the general connection between melody and harmony, even if in jazz we can often find more complex harmonic relationships, for which there is space for improvement.

6. QUALITATIVE EVALUATION

6.1 Blind quiz

In order to evaluate our system from a user point of view, we performed a survey (blind quiz) involving listeners with different musical backgrounds and education levels. All the melodies have been exported into audio tracks and completed with a shuffle drum beat and chords for harmonic and rhythmic context.

Users were asked to rate a set of 15 short melodies (with an average duration of 20 seconds) with a score from 1 to 5 based on how much they liked it. The set was composed of 5 original melodies from the Weimar Jazz DB, 5 generated by MINGUS and 5 generated by BebopNet⁹. Users were unaware of which melodies were original and which ones were generated. The web app used for the quiz is available at <https://mingus.tools.eurecom.fr/>. Figure 2 reports the obtained scores, detailed for 3 categories of users: music lover (8 participants), music student (9), professional musician (11).

As expected, listeners are capable of identifying the original musical phrases with different degrees of confidence, proportional to their level of musical expertise. Overall the evaluation pointed out that MINGUS generations tend to be preferred by the users with respect to BebopNet generations, probably due to the greater harmonic coherence which makes the melodies more pleasing to the ear. There is still a clear difference between machine learning generated samples and original ones, especially when evaluated by high-skilled musicians. It should also be pointed out that this kind of evaluation takes into account very short, selected music segments: the difference between machine-generated and original samples may probably be more evident on long tracks.

⁹ The chosen melodies are available in the repository.

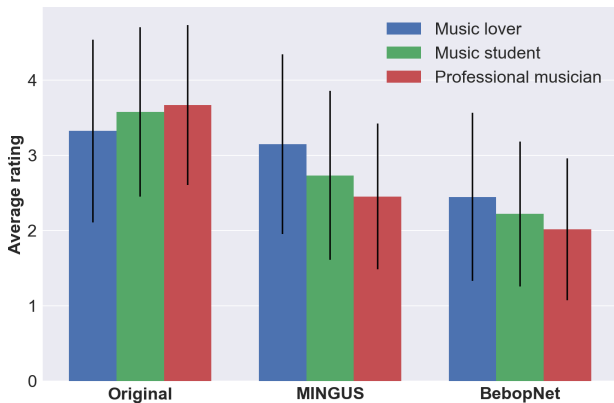


Figure 2: User evaluation summary

6.2 Musical insight on the generations

Taking a look at the generated tracks in musical terms could be useful to identify areas of improvement. In this section, we propose a musical analysis of a phrase generated by MINGUS in comparison with an original phrase. The two phrases are shown in Figure 3¹⁰.

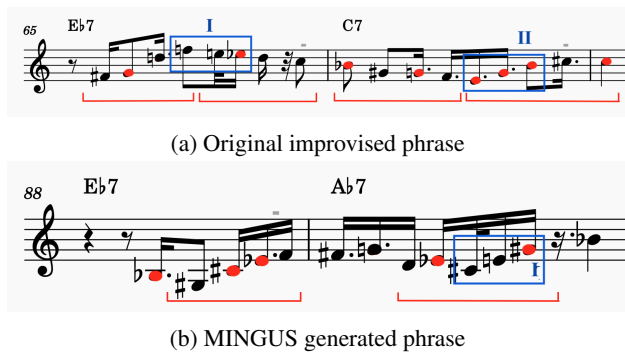


Figure 3: Comparison of original and generated musical phrases on Blues for Blanche by Art Pepper, Weimar Jazz DB. Chord tones in the melody are highlighted in red.

Pitch and duration The figure highlights in red the notes that are part of the underlying chords. In the original improvisation, the red notes are more frequent and have a longer duration. The last note of this phrase is indeed a note of the chord, a typical choice by jazz improvisers because it creates a feeling of tension release. On the other hand, in the phrase generated by MINGUS less importance is given to chord tones. We can observe that notes in phrase 3a present more homogeneous and repetitive duration patterns, while in phrase 3b note durations – although not far from each other – do not follow any specific pattern.

Patterns It is possible to spot some patterns in the construction of the phrases, highlighted in the figure by red lines. In the original one, we can see two clear upward

¹⁰ The two phrases have been chosen because they allow showing recurrent patterns in MINGUS generation, although they do not correspond to the same bars in the standard

and downward motions as the phrase progresses. Although MINGUS seems to have grasped a general idea of such behaviour, the note movement in the generated phrase is not very clear. Other interesting segments are highlighted in blue. In the first blue segment of phrase 3a, the melody is out of tune, but this is justified by chromatic downward motion in the melody. In the second blue segment, the improviser performed a $C7$ arpeggio to end the phrase. An interesting similar behaviour appears in the blue segment of phrase 3b, where also MINGUS performs an arpeggio at the end of the phrase. Unfortunately in this case it is a $C\#m7$ arpeggio, which is out of tune in the key of $A\flat7$, so the result is not quite as pleasing.

Overall, MINGUS has learned to generate musical phrases separated by longer rests with approximate upward and downward motion and approximate harmonic coherence. Nevertheless, the generated phrases still lack a strong internal structure and the typical call-and-response inter-phrase behaviour of jazz solo phrases, with few connections from one generated phrase to the other.

7. CONCLUSIONS

MINGUS uses a transformer architecture to generate music by separately predicting pitch and duration. The model was experimented on popular datasets and evaluated at different levels using a broad range of metrics, revealing comparable performances with respect to the state of the art. The experiment proved the capability of transformers to model and generate realistic melodic lines in the style of a jazz improvisation, with harmonically better results than LSTM. The MINGUS architecture proved to be particularly good at obtaining harmonically coherent melodies.

The choice of metrics has a crucial impact on the evaluation of generation models, making it necessary to use many metrics at different levels of abstraction to obtain a reliable quality estimation.

During the experiments, the conditioning features have shown to learn different hidden representations of the data, which brings to different models. These learned models should not necessarily be ranked on a better-worse scale, but can be considered as alternative sounding. In future work, we intend to further measure the impact of the different features, with the goal of enabling an aware use for generating specific styles and exploring conditioning on other features provided by WjazzDB, such as instrument, jazz style and rhythm feel. In addition, we want to explore techniques for expressive generation as in [26,27] and generation at *phrase* or *lick* level as in [11].

Even though MINGUS has been designed and trained specifically for music modelling and generation, we intend to improve and adapt it for other MIR tasks such as score music classification, bass line generation, automatic harmonisation, assisted composition, automatic music interpretation, conditional regression of musical features. Further research must be carried on for improving music generation systems to achieve long-term phrase-level coherence and to be applied in live conditions, including interacting with musicians for educational and artistic purposes.

8. REFERENCES

- [1] S. Oramas, L. Espinosa-Anke, S. Zhang, H. Saggion, and X. Serra, “Natural Language Processing for Music Information Retrieval (Tutorial),” in *17th International Society for Music Information Retrieval conference (ISMIR)*, New York, USA, 2016.
- [2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>
- [3] C. Payne, “Musenet,” 2019. [Online]. Available: <https://openai.com/blog/musenet/>
- [4] C.-Z. Anna Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music Transformer,” in *International Conference on Learning Representations (ICLR)*, New Orleans, USA, 2019. [Online]. Available: <https://openreview.net/forum?id=rJe4ShAcF7>
- [5] N. Trieu and R. Keller, “JazzGAN: Improvising with Generative Adversarial Networks,” in *6th International Workshop on Musical Metacreation (MUME)*, Salamanca, Spain, Jun. 2018, p. 8. [Online]. Available: <https://doi.org/10.5281/zenodo.4285166>
- [6] J. Jiang, G. Xia, and T. Berg-Kirkpatrick, “Discovering Music Relations with Sequential Attention,” in *1st Workshop on NLP for Music and Audio (NLP4MusA)*. Online: Association for Computational Linguistics, 16 Oct. 2020, pp. 1–5. [Online]. Available: <https://www.aclweb.org/anthology/2020.nlp4musa-1.1>
- [7] F. T. Liang, M. Gotham, M. Johnson, and J. Shotton, “Automatic Stylistic Composition of Bach Chorales with Deep LSTM,” in *18th International Society for Music Information Retrieval Conference (ISMIR)*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., Suzhou, China, 2017, pp. 449–456.
- [8] O. Peracha, “Improving Polyphonic Music Models with Feature-Rich Encoding,” in *21st International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2020. [Online]. Available: https://program.ismir2020.net/poster_2-01.html
- [9] B. Genchel, A. Pati, and A. Lerch, “Explicitly Conditioned Melody Generation: A Case Study with Interdependent RNNs,” in *7th International Workshop on Musical Meta-creation (MUME)*, Charlotte, NC, USA, 2019.
- [10] F. Carnovalini and A. Rodà, “A Multilayered Approach to Automatic Music Generation and Expressive Performance,” in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, Milan, Italy, 2019, pp. 41–48.
- [11] E. P. Nichols, S. Kalonaris, G. Micchi, and A. Aljanaki, “Modeling Baroque Two-Part Counterpoint with Neural Machine Translation,” in *International Computer Music Conference (ICMC)*, I. C. M. Association, Ed., Santiago, Chile, 2020. [Online]. Available: <https://arxiv.org/abs/2006.14221>
- [12] S. H. Hakimi, N. Bhonker, and R. El-Yaniv, “Bebop-Net: Deep Neural Models for Personalized Jazz Improvisations,” in *21st International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2020.
- [13] C.-i. Wang and S. Dubnov, “Context-Aware Hidden Markov Models of Jazz Music with Variable Markov Oracle,” in *8th International Conference on Computational Creativity (ICCC)*, 06 2017.
- [14] R. M. Keller and D. R. Morrison, “A Grammatical Approach to Automatic Improvisation,” in *6th Sound and Music Computing Conference, (SMC)*, Porto, Portugal, 2009.
- [15] J. Biles, “Genjam: A genetic algorithm for generation jazz solos,” in *International Computer Music Conference*, 1994, pp. 131–137.
- [16] O. Mogren, “C-RNN-GAN: A continuous recurrent neural network with adversarial training,” in *Constructive Machine Learning Workshop (CML) at NIPS 2016*, Barcelona, Spain, 2016, p. 1.
- [17] S.-L. Wu and Y.-H. Yang, “The Jazz Transformer on the Front Line: Exploring the Shortcomings of AI-composed Music through Quantitative Measures,” Online, 2020.
- [18] J. Franklin, “Jazz Melody Generation from Recurrent Network Learning of Several Human Melodies,” *International Journal on Artificial Intelligence Tools*, vol. 15, no. 04, pp. 623–650, Aug. 2006. [Online]. Available: <https://doi.org/10.1142/s0218213006002849>
- [19] F. Colombo, S. Muscinelli, A. Seeholzer, J. Brea, and W. Gerstner, “Algorithmic Composition of Melodies with Deep Recurrent Neural Networks,” in *1st Conference on Computer Simulation of Musical Creativity*, Huddersfield, UK, 06 2016.
- [20] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” pp. 1310–1318, June 2013.
- [21] N. Ranjan, K. Mundada, K. Phaltane, and S. Ahmad, “A Survey on Techniques in NLP,” *International Journal of Computer Applications (IJCAI)*, vol. 134, no. 8, pp. 6–9, January 2016. [Online]. Available: <http://doi.org/10.5120/ijca2016907355>

- [22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: <https://www.aclweb.org/anthology/P02-1040>
- [23] S. Kalonaris, T. McLachlan, and A. Aljanaki, “Computational linguistics metrics for the evaluation of two-part counterpoint generated with neural machine translation,” in *1st Workshop on NLP for Music and Audio (NLP4MusA)*. Online: Association for Computational Linguistics, 16 Oct. 2020, pp. 43–48. [Online]. Available: <https://aclanthology.org/2020.nlp4musa-1.9>
- [24] L. Theis, A. van den Oord, and M. Bethge, “A note on the evaluation of generative models,” in *International Conference on Learning Representations (ICLR)*, Apr 2016. [Online]. Available: <http://arxiv.org/abs/1511.01844>
- [25] M. Pfeleiderer, K. Frieler, J. Abeßer, W.-G. Zaddach, and B. Burkhart, Eds., *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017.
- [26] G. Widmer, “Machine Discoveries: A Few Simple, Robust Local Expression Principles,” *Journal of New Music Research*, vol. 31, no. 1, pp. 37–50, 2002. [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1076/jnmr.31.1.37.8103>
- [27] W. Goebel, S. Dixon, G. De Poli, A. Friberg, R. Bresin, and G. Widmer, “Sense in Expressive Music Performance: Data Acquisition, Computational Studies, and Models,” in *Sound to sense, sense to sound : a state of the art in sound and music computing*. Logos ; Sound and Music Computing, 2008, pp. 195–242.

USER-CENTERED EVALUATION OF LYRICS-TO-AUDIO ALIGNMENT

Ninon Lizé Masclef¹

Andrea Vaglio^{1,2}

Manuel Moussallam¹

¹ Deezer Research

² LTCI, Télécom Paris, Institut Polytechnique de Paris

research@deezer.com

ABSTRACT

The growing interest for Human-centered *Music Information Retrieval* (MIR) motivates the development of perceptually-grounded evaluation metrics. Despite remarkable progress of lyrics-to-audio alignment systems in recent years, one thing which remains unresolved is whether the metrics employed to assess their performance are perceptually grounded. Even if a tolerance window for errors was fixed at 0.3s for the *Music Information Retrieval Evaluation eXchange* (MIREX) challenge, no experiment was conducted to confer psychological validity to this threshold. Following an interdisciplinary approach, fueled by psychology and musicology insights, we consider the lyrics-to-audio alignment evaluation from a user-centered perspective. In this paper, we call into question the perceptual robustness of the most commonly used metric to evaluate this task. We investigate the perception of audio and lyrics synchrony through two realistic experimental settings inspired from karaoke, and discuss implications for evaluation metrics. The most striking features of these results are the asymmetrical perceptual thresholds of synchrony perception between lyrics and audio, as well as the influence of rhythmic factors on them.

1. INTRODUCTION

Nowadays, the machine learning community is raising the question of how to design explainable [1] and human-grounded algorithms [2]. Especially in the field of MIR, user studies and evaluation metrics plays a pivotal role in this shift towards Human-centered MIR. Subjective listening tests [3–5] and ethnomusicological studies [6] previously demonstrated the feasibility of including tasks in the real setting context of user experience. Regarding metrics, we are witnessing the transition from exclusively system-centered evaluation to user-aware evaluation. In the reference toolkit `mir_eval`, the lack of Human-centered metrics was justified by the complexity and cost required to develop robust subjective evaluation methods [7]. How-

ever there are a few limitations of system-based evaluation, such as their inability to capture the inherently subjective experience of MIR and the absence of necessary correlation between system-centered evaluation and users' perceptions [8]. One could, and indeed should, ask what is the meaning of the effectiveness of an algorithm without the presence of an embodied experience of human perception? In epistemic terms, how is the distance to the ground truth translated into an error measurement without the mediation of an individual? Since the advent in 2005 of the system-centered evaluation approach by the MIREX, there were several attempts at creating perceptually grounded metrics, notably among the field of music transcription [9–11], source separation [12] and audio similarity [8].

One application at the frontier of music perception and human machine interaction is karaoke. Currently, the majority of alignments used by karaoke systems are fully manually achieved, or partially corrected by human annotators. Obtaining manual annotations of lyrics-to-audio alignment is costly and time-consuming. To obtain such annotations automatically, one could turn to automatic lyrics to audio alignment system. Such system takes as input lyrics text and outputs timed position of their appearance in the audio signal, at the word, line, or paragraph level. Several recent automatic lyrics-to-audio alignment systems have achieved high performance taking inspiration from automatic speech recognition [13–15] and using large public singing voice annotation dataset like DALI [16]. Among the metrics developed for the MIREX challenge to evaluate lyrics-to-audio alignment, the most commonly used is the *Percentage of correct onsets* (PCO) ρ_τ^k , illustrated in [17], using a tolerance window for the perception of lyrics-to-audio alignment errors defined by a threshold τ [17].

$$\rho_\tau^k = \frac{1}{N_k} \sum_{word\ i} 1_{|\hat{t}_i - t_i| < \tau} \times 100 \quad (1)$$

where N_k is the number of words in the track k , t_i the ground truth start of the word timestamp of the lyrics unit and \hat{t}_i the predicted timestamp. It suggests that listeners tolerate errors falling within this window, and still perceive as synchronous lyrics and audio whose onsets are separated by this offset. A tolerance window for errors was fixed at 0.3s for the MIREX, albeit no psychology experiment was conducted to confer validity to this threshold.



Additionally, while spectacular progress has been made in the past years, the gap between state-of-the-art systems, as measured in the MIREX competition, has narrowed, with many systems achieving close to perfect PCO scores on the test sets. Therefore, it might now be important to make room for qualitative rather than quantitative metrics. In this work, we are interested in challenging the PCO metric from a user-centric perspective, focusing on how humans perceive asynchrony to derive stricter metrics for the task. To this aim, we expose the design of two perceptual experiments in Section 3 and their respective results in Section 4. We then propose a PCO adaptation in Section 5 and conclude in Section 6.

2. RELATED WORKS

Singing karaoke engages coordination of articulatory movements, music and language processing systems, as well as crossmodal integration of audio and visual stimuli. It is thus a rich context of perception involving complex stimuli. As a consequence, we briefly consider the research on all the domains outlined above to illustrate paradigms and hypotheses relevant to lyrics-to-audio alignment perception. When presented with a pair of audiovisual stimuli, individuals reported an asymmetric perception of asynchrony, with audio lagging preferred over visual lagging [18, 19]. This asymmetry has been correlated with faster transmission of the visual signal over the audio signal [18] or with the auditory dominance in temporal processing [20]. The latter hypothesis asserts that, when emitting a judgment of synchrony, audio would provide individuals a more accurate sensory information in the case of dynamic event such as music, and also a more stable internal representation of periodicity, contrary to the visual modality [20]. The listening experience is a continuous production of rhythmic expectancies [21]. In the case of sensorimotor synchronisation experiment, one effect induced by rhythmic expectancies is the anticipation of the stimuli in a sequence, also called Negative Mean Asynchrony (NMA). First reported by Dunlap [22], it states that the reaction to an audio stimuli tends to precede rather than follow the stimuli. Repp [23] discovered that individuals anticipate audio events up to 100ms ahead of time. Klemmer [24] revealed that the anticipation effect varies with the tempo of the rhythmic stimuli, usually measured in terms of InterOnset Interval (IOI) duration. He found that the reaction time of individuals when attempting to stay in phase with an isochronous stimulus, is a function of the IOI between stimuli. The reaction time was greater for shorter IOI, suggesting that individuals have less sensibility in slow tempo. These observations were further formalized as a function of local and global rhythmic context by McAuley [25]. Besides global rhythmic factors, the listening experience is punctuated by local variations. Metric events are periodic peaks of attention organized into nested hierarchies that coordinate attention to events on various time-scales, allowing for grouping and accentuation of notes [26]. Musical stresses are the cues to infer a general rhythmic pattern [26]. Among the signif-

icant factors of stress reported were the duration of syllables [26], loudness [27], alignment with beats [21] and sequence boundaries [21, 28].

Given previous studies, our theoretical hypothesis is based on two points. Firstly, we expect individuals to tolerate more audio lagging than lyrics lagging. Secondly, we expect perception of lyrics-to-audio synchrony to rely both on global and local rhythmic context.

3. METHOD

To investigate the perception of lyrics-to-audio alignment, we designed two psychological experiments inspired from the main application of this task, karaoke. We chose karaoke as it is a popular practice where the participants' rhythmical precision is important, requiring attention to the displayed lyrics as much as to the audio. The first experiment is designed to test the influence of global parameters on human perception of audio/displayed lyrics synchrony and to investigate its symmetrical properties. The second experiment intends to explore local factors influences. To run both experiments we developed a karaoke application prototype, whose displayed textual lyrics were intentionally misaligned with the background audio according to various, controlled conditions, thereby creating an audiovisual offset. The stimuli were presented to individuals who then annotated their perceived quality of alignment in different error scenarios. A snippet of this interface is displayed in Figure 1.

Both experiments were run online, through a web interface that was designed to be correctly displayed on both computer and phone screens, for a total duration of two weeks each, between January and April 2021. The first experiment was conducted only with Deezer employees while the second experiment was public and hence involving a larger and more diverse set of participants. Before engaging in karaoke, participants are asked to fill out a questionnaire allowing us to determine their level of musical expertise and familiarity with the practice of karaoke. We collect, with their consent, a range information of their age, declared gender and native language. We do not have control on their external environment when performing karaoke (external noise) or any other factor which might disturb the readability of the karaoke (low light, uncorrected vision problem). Nevertheless, the instructions of the experiment encourage the participants to use headphones and favor a quiet environment.

In both experiments the dependent variable measured is the perceived synchrony and the amount of offset between lyrics and audio is a within subjects factor. In order to prevent from order effect, the values of audiovisual offset are presented in random order. These two experiments are akin to the Simultaneity Judgment task (SJ) widely used in the literature for studying the synchrony perception of audiovisual stimuli [18, 19].

Your karaoke experience [X]

How well do you know this song?
 Not at all Moderately By heart

How well do you know the lyrics of this extract?
 Not at all Moderately By heart

In this extract the lyrics were :
 Ahead of the audio Lagging behind the audio
 Perfectly synchronized with the audio

Do you agree or disagree with the following statement:
 It was easy to sing karaoke on this extract.

Submit

Figure 1. Questionnaire used to evaluate lyrics-to-audio alignment.

3.1 Dataset

Since the measured effects should be valid irrespective of the song, we allow participants to choose their song for karaoke within a set of 80 songs from various genre (pop, rock, rap and metal) and language (English, French, German). We selected popular songs in the DALI dataset [16] with alignment done at word level. The first criterion for the choice of songs was their popularity, so that we can expect a large proportion of participants to be knowledgeable of their lyrics and melody. Other important point guiding our choice was the correct lyrics-to-audio alignment and the absence of syntactical problems. We manually controlled the alignment quality of this subset by visualizing their lyrics in the karaoke prototype and eliminated poorly aligned songs from our selection. To avoid a learning effect of the song, each song can be selected once for a trial and can only be listened to twice during a trial. Moreover, the order of the songs in the selection menu for karaoke is randomized for each trial.

3.2 Influence of global factors

3.2.1 Experiment design

In this experiment, each participant is asked to choose 14 songs from the dataset from which karaoke excerpts are presented. Each audio extract lasts 35 seconds and consists of a sequence of words within lyrical lines, highlighting each word subsequently according to their aligned onset times. A lyrics-to-audio alignment error is generated for each user-song pair randomly from a set of positive and negative offsets between the audio and the lyrics displayed on screen. The offset is fixed for the whole sequence, which means all words in the stimulus are shifted by the same amount. At the end of each trial, participants are asked to report whether they perceive an asynchrony between lyrics and audio with a ternary response ("lyrics ahead", "lyrics lagging", "synchronous"). This experiment has a repeated measure design, with lyrics-to-audio syn-

chrony perception as a dependent variable, and the lyrics-to-audio error offset as the independent variable having 14 modalities. It aims to measure an overall threshold of lyrics-to-audio synchrony perception and to study the influence of global rhythmic factors on this threshold, such as the tempo and word rate. If our theoretical hypothesis is confirmed, we expect to observe a greater proportion of "synchronous" responses for lyrics ahead than lyrics lagging, as well as a modulation of the perceptual threshold with the global rhythmic context (tempo, word rate).

3.2.2 Choice of offsets

In order to precisely define a threshold, we use a wide range of 14 offsets from $-1s$ to $1s$ with negative offsets corresponding to lyrics ahead and reversely positive offsets mean lyrics lagging behind audio. We intentionally keep this number as small as possible, since this value is equal to the number of annotated songs required for each participant. Meanwhile, we wish to highlight effects around the commonly used threshold of $0.3s$ and $-0.3s$. Thus we use smaller steps around these values. We also included larger offsets ($1s$, $0.75s$) as control values, to test that individuals systematically report those as asynchronous. In the same spirit, we expect the offset value 0 to trigger "synchronous" answers. The full experimental protocol was carefully tested beforehand with user testing sessions on six people. Based on these test results, we evaluated that completing the annotation required approximately 12 minutes per participant. Overall, the experiment involved 53 participants who completed the task.

3.3 Influence of local factors

In this second experiment, we make some changes in the karaoke interface. This time, we require each participant to choose one song from the dataset from which 10 audio excerpts are presented with different audiovisual offsets. Each sample is composed of three lyrical lines from the given song. The experience can be repeated multiple times with additional songs if desired. Each song takes around 3 to 5 minutes to annotate. Whilst in the first experiment the alignment errors were located on all the words of the sentence, in the second experiment, the position of the error may be located on the first, the last word of the sentence, or close to a beat. These choices are driven by some of the significant factors of stress described in Section 2 namely alignment with beats [21] and sequence boundaries [21,28]. We decided to discount the influence of long syllables [26] and loudness [27] for this study. In fact, long syllables and loud words are found to be overlapping respectively with the last word of the sentence and words closed to beats. The perceived synchrony is reported as a binary response ("yes", "no") with confidence on a 5-point Likert scale. This experiment intends to quantify the interaction of the error location in the sentence and the offset on the perceived alignment. It has a factorial design with the lyrics-to-audio offset and the position of the error as within subject factors. If our theoretical hypothesis is confirmed, we expect to observe a modulation of the percep-

tual threshold with the location of the error in the sequence.

Proximity of a word to a beat is defined as at a distance less than a *sixteenth note* from the beat, computed as $\frac{1}{16} = 15/\text{Beats Per Minute (BPM)}$. The tempo estimation relies on Anssi Klapuri's algorithm, which showed 80% accuracy with constant tempo during the International Society for Music Information Retrieval (ISMIR) 2004 tempo induction challenge [29]. Starting from the baseline threshold of synchrony perception established in the previous experiment, the second experiment focuses only on lyrics lagging with 3 offsets (0.25, 0.5, 0.75) and a control sample with no offset. We chose only positive offsets because of practical constraints. Indeed, applying a negative offset at the word level can (and does frequently) result in overlapping with previous words, at least for beat-aligned and end words. Filtering out cases of overlapping words resulted in an important selection bias toward very slow songs. To avoid that, we could apply linearly decreasing offsets to precedent words until no overlap remains, as a naive way to "catch-up" with the true annotation. Such behaviour is consistent with what is observed in errors made by lyrics-to-audio alignment systems, multiple errors on consecutive words being recurrent. The concern was that we would not control which first offsetted word the participant would be confronted with. We decided to not consider negative offsets in this experiment but the problem of overlapping words for positive offset remains. However, after applying linearly decreasing offsets to consecutive words until no overlap occurs, the first offsetted word to which each participant is confronted remains the word of interest. Ultimately, we collected 2458 annotations from 193 participants.

4. RESULTS

As we intend to compute an overall threshold of synchrony perception, we perform the analysis at the level of the aggregated results, considering all annotations from all users. We removed all the trials from participants who did not answer correctly to our control levels i.e. "non synchronous" at 1s and "synchronous" at 0s. This represented precisely 11% of answers for the first experiment. We conducted a similar cleaning phase for users of the second experiment using the control offset of 0 and removed 8% of answers.

4.1 Asymmetry of Lyrics-to-Audio Alignment Perception

Using the data collected in the first experiment, we compute an aggregated proportion of respondents who indicate that lyrics and audio are "synchronous", and display it as a function of the lyrics offset in Figure 2. We see that synchrony perception is typically asymmetric, positive offsets being more easily detected than negative ones. This was expected as it resonates with previous findings [18, 19]. Beyond aggregated data, we also looked at individual responses and found that the thresholds were indeed asymmetric for 72% of individuals.

To give perspective, we plot the window function that

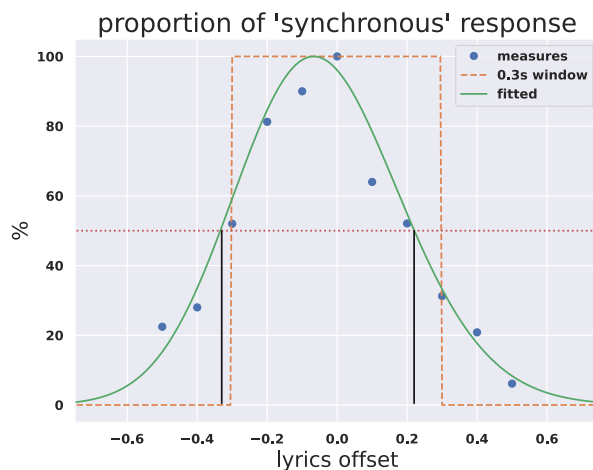


Figure 2. Aggregated results of *synchronous* judgment as a function of the lyrics/audio offset.

correspond to the PCO metric scoring as used in MIREX, with an absolute threshold value of 0.3s. We also fit a function akin to a scaled skew normal distribution function to the data points. Among several attempts with asymmetrical continuous functions, this was the best fit we obtained, although it does not respect the maximality at 0. Parameters of the fitted function are a skewness factor of 1.12, a location of -0.22 and a scale of 0.29, a multiplicative factor is also applied in order to have a value of 1. at the maximum. Using this function we can derive new perceptual thresholds for synchrony using a simple rule of 50% of respondents being able to detect the offset. For lyrics ahead and lyrics lagging we respectively identify the offsets -0.33 s and 0.22 s. Given the amount of noise in the data, we can reduce these to -0.3 s and 0.2 s and examine if the differences of perception are significant for these values. Indeed, pairwise tests revealed a significant difference of proportions of response "synchronous" on the levels -0.3 and 0.3 s ($\chi^2(1) = 4.26$, $p = .038$), while proportions on the levels -0.3 s and 0.2 are not statistically different ($\chi^2(1) = 0.04$, $p = .08$).

4.2 Sensitivity to global rhythmic context

In order to assess whether there is an influence of the global rhythmic context on lyrics-to-audio alignment perception, we compared the distribution of "synchronous" responses at each offset for two rhythmic factors: tempo and Words Per Second (WPS). We split our dataset of songs into two classes of tempo, defined as the upper and lower quartiles of the distribution of tempo, respectively fast (≥ 138 BPM) and slow (≤ 93 BPM). Although it is correlated with tempo, we also consider the average WPS rate of songs as a meaningful global factors. Again, we look at the first and last quartiles as Low (≤ 1.16 WPS) and respectively High WPS (≥ 1.2 WPS) classes. Figures 3 and 4 show the aggregated reported synchrony profiles for the negative offsets for the derived tempo and WPS classes. On both metrics, we observed no threshold discrepancy between the two classes for positive offsets.

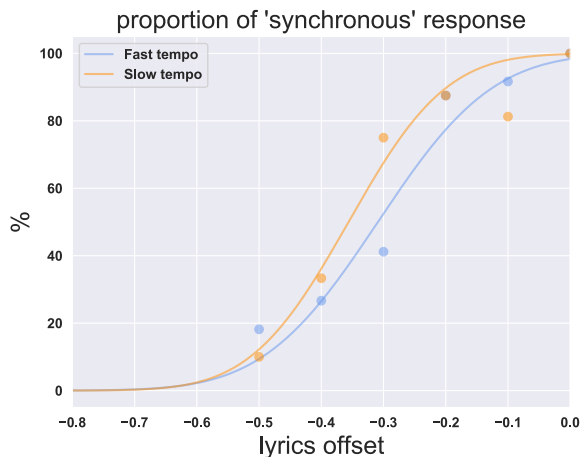


Figure 3. Proportion of response "synchronous" by tempo class.

To highlight the differences between classes, we fit a relatively simple sigmoid function to the data points. Among several candidates, a Gauss error function seemed most appropriate. Fitted functions are also displayed on Figures 3 and 4 and emphasize the different synchrony slopes. As before, we particularly consider the offset value intersecting with an average of 50% of "synchronous" responses as an indicator of participants' sensitivity to temporal asynchronies. Interestingly, the 50% threshold for the perception of synchrony is located at a larger offset (-0.36) for slow tempo than in fast tempo (-0.31). These results show that individuals report more frequently lyrics ahead as synchronous with slow tempo than with fast tempo. The lower sensitivity to lyrics-to-audio alignment errors in slow tempo is consistent with the results of [24]. Significance of these results are tested. The proportions of "synchronous" response at the offset $-0.3s$ show significant difference between songs with high and low tempo ($\chi^2(1) = 5.44$, and $p < .02$).

Analogously, we found out that subjects are more tolerant to lyrics ahead (audio lagging) in high word rate than in low word rate. The 50% threshold for the perception of synchrony is indeed located at a larger offset for high word rate (-0.39) than in low word rate (-0.28) (Figure 4). These results show that subjects are more tolerant to lyrics ahead (audio lagging) in high word rate than in low word rate. We again tested the significance of these results. The proportions of "synchronous" response at the offset $-0.3s$ are significantly different between songs with high and low WPS ($\chi^2(1) = 16.86$, and $p < .00004$).

4.3 Interaction between offset and word position

We designed the second experiment to distinguish perception of asynchrony as a function of the words position in the sentence. As explained in Section 3.3, we are only able to test for positive offsets. As insights from the previous experiment, we can assume that user sensibility is less affected by global factors for positive offsets. As a result, we expected it to be challenging for local factors too. For

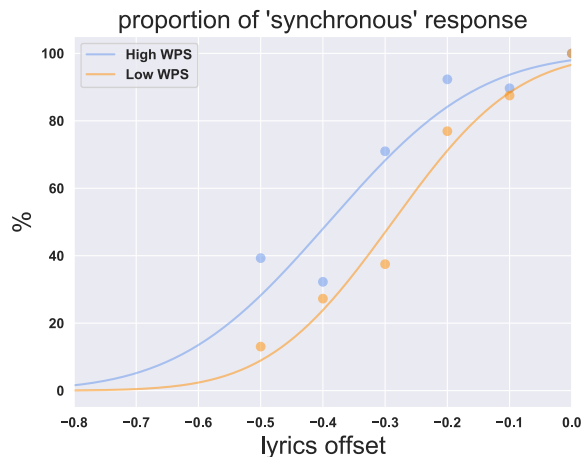


Figure 4. Proportion of response "synchronous" by WPS class.

this reason, we aimed at collecting a much larger set of annotation, with a reduced set of tested offsets.

Figure 5 presents an overview of the results. There is a fairly large amount of noise in the collected data points, and few clear differences between synchrony perceptions for the three classes of word positions. The noise is particularly clear from the displayed level of confidence of participants who were unable to detect the asynchrony even for large values of the offset, but still were quite confident about their choice (average around 3.8). Regarding the location of the alignment error within the sentence, Cochran's Q test did not indicate a notable difference among the proportions of synchrony responses reported for the three error positions, $\chi^2(2) = 5.77$, $p = .056$.

The only visible effect seems to be for words aligned on *beats*, for which the confidence in the "asynchronous" answer at the 0.25 level is markedly higher than for the *end* class. More precisely, a Wilcoxon signed-rank test revealed that lyrics-to-audio alignment comparing error located on the beat with those on the last word did elicit a statistically significant change in the reported confidence of perception of error in individuals at the 0.25 level ($Z = 2.756$, $p < 0.006$). Indeed, mean confidence rating was 4.1 for error on the beat and 3.5 for error on the last word of the sentence. Such phenomenon is not observed for the synchronous case.

5. DISCUSSION

5.1 General discussion

Building on psychological theory and previous studies, we had hypothesized that a perceptual evaluation of lyrics/audio alignment quality would be asymmetrical and depend on both global and local factors. Using a first experiment we did find strong evidence for asymmetry and, to some extent for global factors influence. Despite a much larger experimental setup which involved hundreds of participants, we were not able to exhibit a clear influence of the local factors we tested. This negative result could mean

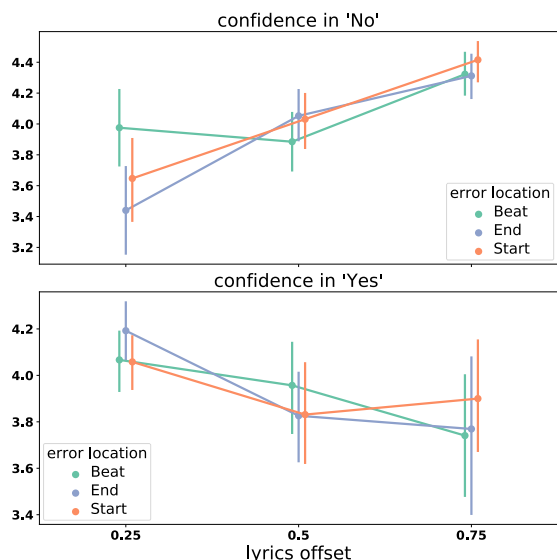


Figure 5. Plot of the reported answer (synchronous is "yes", asynchronous is "no") and confidence score (5 level Likert scale) in the perception of synchrony, by location of alignment error within the sentence.

that the local factors considered, i.e. the word position in the lyrical line are not the relevant ones. It is possible that words grammatical or semantic functions are more subject to human attention in a karaoke context. Indeed, the only significant phenomenon that we observed was on words located on *beats*, for which the asynchrony perception was more acute. Future work should investigate the relationship between rhythmic position and lyrical function of words and test new hypothesis of perceptual differences. Finally, although we did our best to build a realistic yet controlled experimental setup, we acknowledge that as a psychological experiment mostly conducted online, we can not completely rule out the possibility that the measurement noise was too high to allow us to detect signals on local factors.

There are arguably other factors that could influence this perception, notably at the human level. Indeed, familiarity with the song (e.g. previous knowledge of the lyrics and/or the music), but also participants’ facility with the languages, level of musical expertise and even karaoke practice could be important variables to consider. In the conducted experiment, we collected such information from participants. Although we did observe some interesting phenomenon, for the sake of clarity, we chose not to present additional results on these variables here and leave it to a follow-up study.

5.2 Implication for evaluation metrics

Here we would like to present a practical use of our results, as a perceptually motivated evaluation metric for lyrics to audio alignment tasks. Overall, we propose a generalization of the PCO metric in the following form:

$$\psi^k = \frac{1}{N_k} \sum_{word\ i} f(\hat{t}_i - t_i) \times 100 \quad (2)$$

	PCO	Asym-PCO	Perc-PCO
Gupta [13]	94.47 (1.52)	93.66 (1.59)	89.94 (1.71)
Vaglio [15]	91.85 (1.95)	90.82 (2.04)	86.79 (2.13)
Stoller [14]	87.02 (2.97)	85.23 (3.07)	79.93 (2.90)

Table 1. Averaged metrics over the Jamendo dataset songs. Standard errors are given in parenthesis.

where the function f can be seen as penalty weighting of the annotation offset and other notations are common with Equation 1. We then evaluated 3 state-of-the-art automatic lyrics-to-audio alignment models [14, 15, 30], on the 20 songs of the Jamendo dataset [14]. We have compared using the regular PCO ($f = 1_{[-0.3,0.3]}$), a slightly modified version still using a square window but taking into account the asymmetrical perceptual perception ($f = 1_{[-0.3,0.2]}$) and a Perceptual-PCO function that is the one fit from the data collected in our first experiment and depicted in Figure 2. This function can be seen as a smooth relaxation of the square window, taking into account the perceptive asymmetry of the error slopes.

Results are compiled in Table 1. Interestingly, there appears to be little difference between using the standard PCO window and a slightly shifted one. However, scores for the perceptual-PCO are much lower. This is despite the window support being larger (i.e. errors of more than 0.3s are not completely nullified). In our opinion, this new metric is better suited to capture the relative importance of alignment errors and weights them according to human perception. It can also help for comparing between alignment methods that achieve near perfect scores with the standard PCO. It is worth noticing that although we demonstrated it on the PCO, a similar weighting could be applied to other alignment metrics. A step further would be to parameterize the window function f on global song factors such as tempo and WPS. This would arguably require additional experiments with a larger, more diverse set of songs.

6. CONCLUSION

In this work, we challenged the objective evaluation of lyrics-to-audio alignment using hypothesis from psychological theory. We postulated three effects: asymmetry, influence of songs features and influence of words local positions. We were able to demonstrate the first two effects using a large scale online experiment, disguising the synchrony annotation task as a Karaoke experience. This framework proved less efficient for the third effect, despite our efforts to collect up to several thousands annotation points. We nonetheless proposed a readily usable weighting function to allow finer comparison between state-of-the-art alignment methods. Future work will investigate more diverse sets of factors, both on musical attributes and user features.

7. REFERENCES

- [1] A. Adadi and M. Berrada, “Peeking inside the black-box: a survey on explainable artificial intelligence (xai),” *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.
- [2] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proc. of the 40th annual meeting of the Association for Computational Linguistics (ACL)*, 2002, pp. 311–318.
- [3] L. Yin-Jyun, C. Ming-Tso, and C. Tai-Shih, “Singing voice correction using canonical time warping,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 396–400.
- [4] C. Pei-Chun, L. Keng-Sheng, and C. Homer, “Emotional accompaniment generation system based on harmonic progression,” *IEEE Trans. on Multimedia*, vol. 15, no. 7, pp. 1469–1479, 2013.
- [5] A. Huang and R. Wu, “Deep learning for music,” *arXiv preprint arXiv:1606.04930*, vol. abs/1606.04930, 2016.
- [6] A. Holzapfel and E. Benetos, “Automatic Music Transcription and Ethnomusicology: a User Study,” in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 678–684.
- [7] C. Raffel, B. Mcfee, E. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. Ellis, “mir_eval: A transparent implementation of common mir metrics,” in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 10 2014.
- [8] X. Hu and N. Kando, “User-centered Measures vs. System Effectiveness in Finding Similar Songs,” in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Oct. 2012, pp. 331–336.
- [9] A. Daniel, V. Emiya, and B. David, “Perceptually-based evaluation of the errors usually made when automatically transcribing music,” in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2008.
- [10] A. Ycart, L. Liu, E. Benetos, and M. Pearce, “Investigating the perceptual validity of evaluation metrics for automatic piano music transcription,” *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, vol. 3, pp. 68–81, 2020.
- [11] A. Ycart, L. Liu, E. Benetos, and M. T. Pearce, “Musical features for automatic music transcription evaluation,” *arXiv preprint arXiv:2004.07171*, 2020.
- [12] E. Vincent, “Improved perceptual metrics for the evaluation of audio source separation,” in *Int. Conf. on Latent Variable Analysis and Signal Separation (LVA/ICA)*. Berlin, Heidelberg: Springer-Verlag, 2012, p. 430–437.
- [13] C. Gupta, E. Yılmaz, and H. Li, “Automatic lyrics transcription in polyphonic music: Does background music help?” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [14] D. Stoller, S. Durand, and S. Ewert, “End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 181–185.
- [15] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. D’alché-Buc, “Multilingual lyrics-to-audio alignment,” in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020.
- [16] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, “Creating dali, a large dataset of synchronized audio, lyrics, and notes,” *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, 2020.
- [17] M. Mauch, H. Fujihara, and M. Goto, “Integrating additional chord information into hmm-based lyrics-to-audio alignment,” *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 20, no. 1, pp. 200–210, 2012.
- [18] R. L. J. van Eijk, A. Kohlrausch, J. F. Juola, and S. van de Par, “Audiovisual synchrony and temporal order judgments: effects of experimental method and stimulus type,” *Attention, Perception, & Psychophysics*, vol. 70, no. 6, p. 955–968, 2008.
- [19] A. Vatakis, B. Fuat, D. L. Massimiliano, and C. Ángel, *Timing and Time Perception: Procedures, Measures, & Applications*. Brill, 2018.
- [20] B. Repp and A. Penel, “Auditory dominance in temporal processing: New evidence from synchronization with simultaneous visual and auditory sequences,” *Journal of experimental psychology. Human perception and performance*, vol. 28, pp. 1085–99, 11 2002.
- [21] M. R. Jones and M. Boltz, “Dynamic attending and responses to time,” *Psychological Review*, pp. 459–491, 1989.
- [22] K. Dunlap, “Reaction to rhythmic stimuli with attempt to synchronize,” *Psychological Review*, vol. 17, pp. 399–416, 1910.
- [23] B. Repp, “Sensorimotor synchronization: A review of the tapping literature,” *Psychonomic bulletin & review*, vol. 12, pp. 969–92, 01 2006.
- [24] E. T. Klemmer, “Simple reaction time as a function of time uncertainty,” *Journal of experimental psychology*, vol. 54, no. 3, pp. 195–200, 1957.
- [25] J. D. McAuley and N. S. Miller, “Picking up the pace: Effects of global temporal context on sensitivity to the tempo of auditory sequences,” *Perception & Psychophysics*, vol. 69, pp. 709–718, 2007.

- [26] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music*. The MIT Press, 06 1996.
- [27] A. M. C. Sluijter, V. J. van Heuven, and J. J. A. Pacilly, “Spectral balance as a cue in the perception of linguistic stress,” *The Journal of the Acoustical Society of America (JASA)*, vol. 101, no. 1, pp. 503–513, 1997.
- [28] T. R. Knösche, C. Neuhaus, J. Haueisen, K. Alter, B. Maess, O. W. Witte, and A. D. Friederici, “Perception of phrase structure in music,” *Human Brain Mapping*, vol. 24, no. 4, pp. 259–273, 2005.
- [29] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, “An experimental comparison of audio tempo induction algorithms,” *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 14, pp. 1832–1844, 2006.
- [30] C. Gupta, H. Li, and Y. Wang, “Automatic pronunciation evaluation of singing,” in *Int. Speech Communication Association (INTERSPEECH)*, 09 2018, pp. 1507–1511.

SYNTHESIZER SOUND MATCHING WITH DIFFERENTIABLE DSP

Naotake Masuda Daisuke Saito

The University of Tokyo

{n_masuda, dsk_saito}@gavo.t.u-tokyo.ac.jp

ABSTRACT

While synthesizers have become commonplace in music production, many users find it difficult to control the parameters of a synthesizer to create the intended sound. In order to assist the user, the *sound matching* task aims to estimate synthesis parameters that produce a sound closest to the query sound. Recently, neural networks have been employed for this task. These neural networks are trained on paired data of synthesis parameters and the corresponding output sound, optimizing a loss of synthesis parameters. However, synthesis parameters are only indirectly correlated with the audio output. Another problem is that query made by the user usually consists of real-world sounds, different from the synthesizer output used during training. In this paper, we propose a novel approach to the problem of synthesizer sound matching by implementing a basic subtractive synthesizer using differentiable DSP modules. This synthesizer has interpretable controls and is similar to those used in music production. We can then train an estimator network by directly optimizing the spectral similarity of the synthesized output. Furthermore, we can train the network on real-world sounds whose ground-truth synthesis parameters are unavailable. We pre-train the network with parameter loss and fine-tune the model with spectral loss using real-world sounds. We show that the proposed method finds better matches compared to baseline models.

1. INTRODUCTION

Synthesizers have become an essential tool in modern music production, owing to their ability to produce a wide array of sounds. The user can directly interact with the parameters of the audio synthesis algorithm to discover interesting sounds. Despite the prevalence of synthesizers in modern music production, most music producers still find it difficult to control the parameters of a synthesizer. The relationships between a synthesis parameter and the perceptual qualities of the output sound is unclear, and complex synthesis algorithms create unexpected outputs. As such, producers often rely on *presets*, parameter settings crafted by sound designers. By using presets, producers can easily incorporate appealing sounds. However, finding

an appropriate preset can be difficult, and the sonic palette of the synthesizer is limited by the availability of presets.

Thus, there is great need for user assistance in the sound design process using a synthesizer. One way of assisting the user is automatic programming of the synthesizer to imitate a certain sound. We will refer to this task as *sound matching*. Given a query sound that the user wants to imitate, parameters for a certain synthesizer that produces the best match possible with the synthesizer is estimated. Recently, neural networks have been employed for the sound matching task [1–3]. They are trained to predict the ground-truth synthesis parameter from the synthesized sound, minimizing the error of estimated parameters.

However, the parameter loss maybe a suboptimal loss for optimization. In fact, the model that performs the best in terms of parameter loss does not always return the best match in terms of spectral features [3]. Since we are interested only in the audio match quality, it is better to optimize the network using a loss directly related to the audio. Unfortunately, conventional synthesizers do not allow for backpropagation of the gradients, leaving this problem unaddressed in previous works.

Another problem is that the models in previous works can only be trained on sounds created by the synthesizer (we will refer to them as *in-domain* sounds), as the best matching parameters for sounds not created by the synthesizer (*out-of-domain*) are unknown. In a sound matching application, the system should expect queries consisting of sounds not created by the same synthesizer, originating from acoustic instruments or different synthesizers. Thus, there is a gap in the domain between training and inference, which has been unaddressed by previous works.

In this paper, we address the two problems mentioned above by implementing a synthesizer with interpretable controls using differentiable DSP [4] modules. In our method, the estimator network is optimized in an end-to-end framework including the synthesizer. This allows us to utilize not only parameter loss but also spectral loss, which is more directly related to the audio output of the system. Also, since the ground-truth parameter values are unnecessary when optimizing for spectral loss, the model can be trained using out-of-domain sounds that better represent queries in real-life applications. The proposed model is pre-trained using parameter loss on in-domain data and fine-tuned with spectral loss on out-of-domain data. We show the effectiveness of our method in matching out-of-domain sounds through quantitative measures and subjective evaluations.



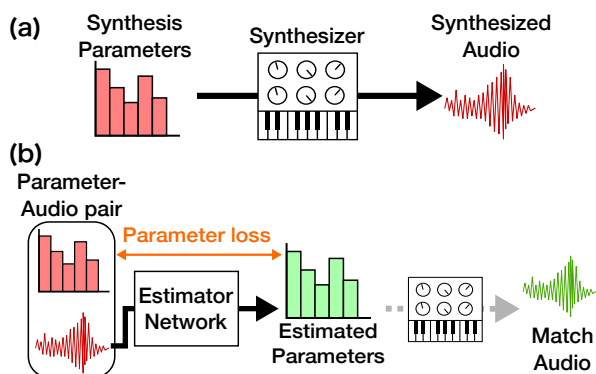


Figure 1. NN-based synthesizer sound matching. (a) A synthesizer renders audio according to synthesis parameters. (b) An estimator network is trained to estimate the parameters from the sound, optimizing the parameter loss.

2. RELATED WORKS

2.1 Synthesizer Sound Matching

Synthesizer sound matching is a task that aims to estimate the parameters of a synthesizer to produce a sound similar to the query sound. This supposes an application where the user has a sound that is similar to the desired sound, and queries the system for a parameter setting that produces a similar sound. The user can then adjust the sound further using the synthesizer. This is akin to the *query-by-example* approach in sound retrieval, where a sound that best matches the query sound is retrieved from the database [5]. In fact, sound matching has been realized by retrieval of presets from a database [6]. However, a large database of presets that sufficiently cover the capabilities of a synthesizer is not available for most synthesizers, as the distribution of presets is limited and often not free.

One of the earliest work in sound matching used genetic algorithm (GA) to match a target spectrum frame [7]. Subsequent works using GA challenged more complex tasks, using conventional synthesizer software to match the entire sound [8]. GA-based sound matching directly optimizes for the similarity of the query sound and the match sound in terms of audio features such as spectral distance. However, GA-based sound matching can take anywhere from 10 minutes to several hours to match a single sound, since fitness of each individual can only be evaluated by rendering audio. For example, a single run with population size of 200 for 200 generations would require 40,000 renders of the synthesizer.

Recently, supervised machine learning methods such as multiple linear regression [9] and neural networks (NNs) [1–3] have been used for sound matching. These methods view sound matching as a regression problem, where the synthesis parameters are estimated from audio features. This is illustrated in Figure 1. While NNs allow for fast estimation of synthesis parameters during inference, they optimize the parameter loss and not the actual match quality of the synthesized audio. This is because gradients can not be propagated through the conventional synthesizer.

To circumvent the same problem in the case of black-box audio effects, stochastic gradient approximation methods were applied [10]. However, the gradients obtained for the audio effect are only approximate.

It is also important to consider the actual applications of sound matching. It is expected that users will want to match sounds that were not originally made by the synthesizer. For example, use of vocal imitation as a query for sound matching has been proposed [6]. Perhaps a user will want to imitate acoustic instrument sounds using a synthesizer. While such sounds cannot be matched perfectly, synthesizers can imitate some of their qualities, leading to the discovery of unique sounds. Thus, out-of-domain sounds should be the focus of sound matching. For conventional neural network models trained on pairs of synthesis parameters and the corresponding audio output, such out-of-domain sounds are unseen during training.

2.2 Neural Audio Synthesis

Recently, neural networks have garnered attention as a new way to synthesize musical sounds. Since a typical neural network has millions of model parameters with no interpretability, it is impossible to directly interact with the parameters of a neural audio synthesizer as one would with the synthesis parameters of a conventional synthesizer. As such, neural networks must offer another way to control the synthesis. This is achieved through either model conditioning or learning a latent representation of musical sounds.

SING is a neural audio synthesizer that can be conditioned by the pitch, velocity, and instrument labels [11]. Embeddings for the instrument can be learned to adjust the timbre more flexibly [12]. Alternatively, an autoencoder can be used to learn the latent representation of musical sounds. A standard autoencoder with feedforward layers was used to reconstruct spectral frames [13]. A WaveNet autoencoder can be used to model raw audio of musical sounds [14]. Autoencoder models encode the audio into a compact representation and decode it to reconstruct the audio. By modifying this representation, the output sound can be controlled.

Compared to conventional synthesizers, neural audio synthesizers can potentially create more realistic sounds and offer a novel way to control musical sounds. However, they do not provide full control to the user over the synthesis process, and their use in practical music production has been limited so far.

2.3 Differentiable DSP

While neural audio synthesizers aim to generate raw audio using only neural networks, differentiable digital signal processing (DDSP) aims to integrate conventional signal processing elements with deep learning [4]. The parameters of a differentiable audio synthesis model are estimated by a neural network in an end-to-end manner. In the original DDSP paper, a differentiable version of an additive synthesis model called the harmonics-plus-noise model is used to generate audio. This is a variant of the sinusoids-plus-noise model [15]. While the harmonics-plus-noise

model can be considered as a synthesizer, it is much more complicated than conventional synthesizers, as the amplitude of each harmonic and the full frequency response of the filter must be specified. The harmonics-plus-noise synthesizer allows for accurate reconstruction of real instrument sounds, but the synthesis parameters are far too many to allow for direct interaction.

The idea of DDSP has inspired a handful of works. Adversarial loss was used with a hierarchical generator network to improve the quality of the output [16]. Pitch detection of musical signals was accomplished by using differentiable DSP in a self-supervised framework [17]. New differentiable DSP modules have been proposed as well. An infinite impulse response (IIR) filter was implemented using differentiable DSP and its parameters were trained to emulate a guitar pedal [18]. Similarly, differentiable bi-quad filters were used for parametric equalizer matching, where optimizing spectral loss was shown to be superior to parameter loss [19]. Our work expands this idea to the synthesizer sound matching problem.

Parallels can be drawn between differentiable DSP and differentiable rendering. Differentiable rendering aims to integrate rendering of 3D objects into a deep learning framework [20]. 3D attributes of an object were estimated from a 2D image in an end-to-end framework [21]. This network was first trained by a 3D attribute prediction loss using ground-truth labels, and a projection loss using a renderer was introduced afterwards. This is similar to our training procedure, in which the network is pre-trained by synthesis parameter estimation loss, and spectral loss using the differentiable synthesizer is introduced afterwards.

3. PROPOSED METHOD

3.1 Overview

A diagram of the proposed method is shown in Figure 2. Melspectrogram frames calculated from the audio are fed into a neural network to estimate the synthesis parameters frame-by-frame. The in-domain dataset consists of synthesis parameters and the synthesized sound. This is generated by random sampling of synthesis parameters. For in-domain sounds, the parameter estimation loss is calculated between the estimated and the ground-truth synthesis parameters, in a similar manner to conventional NN-based synthesizer sound matching. Finally, a differentiable synthesizer is used to render the audio from the synthesis parameters. This allows for end-to-end training using spectral loss, for both in-domain and out-of-domain sounds. By using out-of-domain sounds for training, it is expected that our proposed method will be better able to generalize to actual queries consisting of real-world sounds.

3.2 Differentiable Synthesizer

An *additive-subtractive* synthesizer that approximates a classical subtractive synthesizer using additive synthesis was implemented in PyTorch. This design is inspired by popular additive-subtractive synthesizer software such as *Harmor* by Image-Line or *Razor* by Native Instruments.

This synthesizer features two oscillators with varying pitch and amplitude. The waveform of each oscillator can be interpolated between a sawtooth wave and a square wave. Each oscillator is implemented in an additive way, meaning that sine oscillators with different frequencies are added up to create a waveform with richer harmonic. More specifically, sawtooth waveform and square waveform with fundamental frequency f can be decomposed into sine waves as follows:

$$x_{sawtooth}(t) = \frac{2}{\pi} \sum_{k=1}^{\infty} \frac{\sin(k \cdot 2\pi ft)}{k}, \tag{1}$$

$$x_{square}(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin\{(2k - 1)2\pi ft\}}{2k - 1}. \tag{2}$$

The output of two oscillators are mixed and fed into a resonant low-pass filter. This filter can alter the timbre by attenuating the harmonics above the cutoff frequency and accentuating the harmonics around the cutoff frequency according to its resonance parameter. While previous works has proposed a differentiable implementation of a resonant IIR filter [18], IIR inherently involves recurrent computation which is computationally expensive. Thus, we approximate a resonant filter by applying the frequency response of the filter as a multiplier to the amplitudes of the harmonics. Ultimately, the parameters of this synthesizer are as follows: amplitude, frequency and saw/square wave mix of each oscillator and the cutoff frequency and resonance of the filter.

The parameters of the synthesizer can change over time to create movement in a sound. Conventional synthesizers use envelope generators and low-frequency oscillators (LFOs) to control the modulation of some important synthesis parameters. In our experiments, the amplitudes of each oscillator and the cutoff frequency of the filter were estimated frame-by-frame, and other parameters were set to a single value which was estimated from the last output of the GRU.

Our method aims to assist the user in controlling a practical synthesizer, so we implemented a differentiable synthesizer with familiar controls such as cutoff frequency. While this synthesizer has fewer parameters compared to the harmonics-plus-noise model in the original DDSP, we find that parameter estimation is more difficult for this synthesizer than the harmonics-plus-noise model. We suppose that this is due to the indirectness of the relationship between the synthesis parameters and the output spectrum. A single synthesis parameter in the harmonics-plus-noise model roughly corresponds to a single spectrogram time-frequency component of the output, especially when it is conditioned by the fundamental frequency. On the other hand, a synthesis parameter in the proposed synthesizer model can affect many components, and its effects are dependent on each other. Furthermore, our synthesizer is intentionally limited in terms of the sounds it can create. The estimator must utilize the synthesizer to roughly imitate features of the target sound, which imposes a unique challenge.

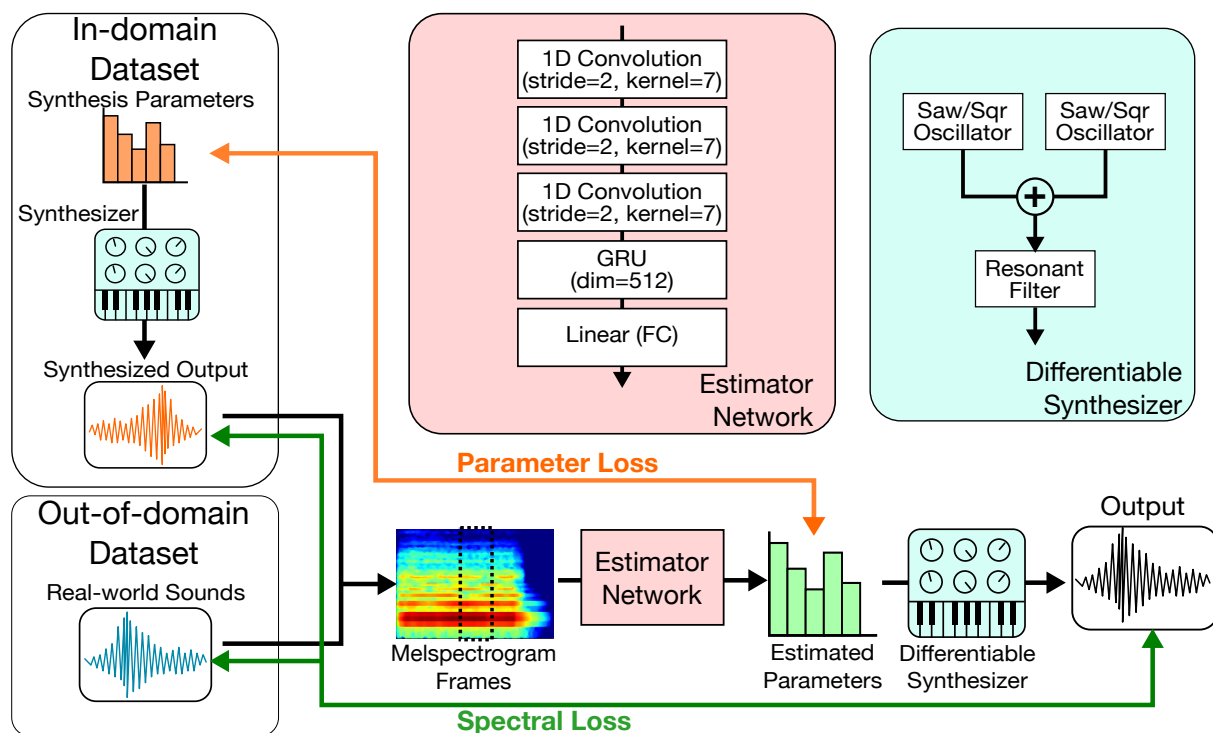


Figure 2. The architecture of the proposed model. Two different losses can be calculated in this framework: the parameter loss and the spectral loss. Parameter loss can be calculated only for in-domain data, whose ground-truth synthesis parameters are available.

3.3 Training

The estimator network can be trained using both the parameter loss and the spectral loss. The parameter loss is defined as the L1 loss between the estimated parameter and the ground-truth synthesis parameter. The spectral loss is a multi-scale spectrogram loss [4], which is defined as the sum of L1 loss of spectrograms and log-spectrograms in multiple resolutions. In the experiments, we use FFT sizes of (64, 128, 256, 512, 1024, 2048). Frames were overlapped at 75% with the next frame.

During preliminary experiments, we found that training the model with only spectral loss was ineffective. We suppose that this is due to the indirectness of the relationship between the synthesis parameters and the output spectrum. We found that pre-training the model by parameter loss and fine-tuning the model using spectral loss was the most effective. More specifically, our training procedure can be split into three steps. First, the network is trained by parameter loss on the in-domain dataset. Next, spectral loss is gradually introduced and eventually replaces the parameter loss completely. Finally, the model is trained using the out-of-domain dataset.

This final step can be considered as a form of *domain adaptation*. Specifically, unsupervised domain adaptation aims to transfer the knowledge of labeled source domain to a target domain with no labels [22]. While ground-truth parameter values are unavailable for the out-of-domain sounds, we can transfer the knowledge of models trained using in-domain sounds to out-of-domain sounds by switching to the spectral loss.

4. EXPERIMENT SETUP

4.1 Training Procedure

The proposed method aims to improve the quality of sound matching by use of spectral loss and adaptation to out-of-domain data. To examine the effect of each, the performance of models trained using three different training procedures are compared.

- *Parameter-loss only model* (hereinafter, denoted as *P-loss*). This model is trained using only parameter loss for 400 epochs. This is in line with conventional NN-based sound matching methods and serves as the baseline of our experiment.
- *In-domain spectral loss model* (*Synth*). This model is pre-trained using parameter loss for 50 epochs. For the next 150 epochs, a spectral loss is gradually introduced by increasing the weighting of the spectral loss linearly and decreasing that of the parameter loss. Finally, the model is trained for 200 epochs using only the spectral loss on the in-domain dataset.
- *Out-of-domain spectral loss model* (*Real*). This model is trained in the same way as the *Synth* model for the first 200 epochs. Then, the model is trained for 200 epochs using the spectral loss on the out-of-domain dataset.

The learning rate is decreased with an exponential decay rate of 0.99 every epoch (16000 iterations). To match

the scale of the parameter loss and spectral loss, the L1 parameter loss is multiplied by a factor of 10 during training. The network was trained with a batch size of 64.

4.2 Estimator Network

The estimator network is trained to predict the synthesis parameter at each time step from the melspectrogram of input audio. Melspectrogram frames with 128 bands were extracted from the input waveform with an FFT size of 1024 samples and a hop size of 256 samples. Each frame is fed into 3 layers of 1D convolution with batch normalization to obtain a high-level representation of spectral features. Then, the output is fed into a gated recurrent unit (GRU) layer. Finally, the output of GRU is fed into a linear layer. Since all synthesis parameters are normalized to be within (0, 1), sigmoid nonlinearity is applied to the network output.

4.3 Dataset

4.3.1 In-domain

The in-domain dataset is generated by randomly sampling synthesis parameter settings and rendering them with the same synthesizer used in the model. The value of a static parameter is uniformly randomized. The temporal evolution of a dynamic parameter is modelled by an attack-decay-sustain-release (ADSR) envelope generator used in most conventional synthesizers. The parameters of this envelope generator are attack time, decay time, sustain level, release time and the peak/floor levels. These envelope parameters are uniformly randomized for each dynamic parameter. Gaussian noise is added to the output of this envelope generator to model the fluctuation present in real-world sounds. The note-off point triggering the release stage of the envelope is at 3 seconds, and the audio was recorded for 4 seconds. Parameter settings that resulted in silence were removed from the dataset. 20,000 sound-parameter pairs were generated, and partitioned into an 80-10-10 train-validation-test split.

4.3.2 Out-of-domain

For the out-of-domain sounds, the NSynth dataset [14] was used. This dataset includes acoustic and synthetic musical sounds from sample libraries. They were played with MIDI notes in various pitch lasting 3 seconds and recorded for 4 seconds at sampling rate of 16kHz. 20,000 sounds were randomly selected from the full dataset and partitioned into an 80-10-10 train-validation-test split.

5. RESULTS

We perform objective and subjective evaluation of the sound matching results and discuss our findings. Audio examples and source code are available on the accompanying webpage¹.

¹<https://hyakuchiki.github.io/DiffSynthISMIR/>

Models	In-domain			Out-of-domain	
	Param	LSD	Multi	LSD	Multi
<i>P-loss</i>	0.065	16.14	4.72	19.60	8.84
<i>Synth</i>	0.083	14.38	3.37	19.13	5.79
<i>Real</i>	0.177	15.35	3.87	17.27	3.90

Table 1. Objective measures of sound matching (Param: L1 parameter loss, LSD: log-spectral distortion, Multi: multi-scale spectrogram loss). Smaller values indicate better performance.

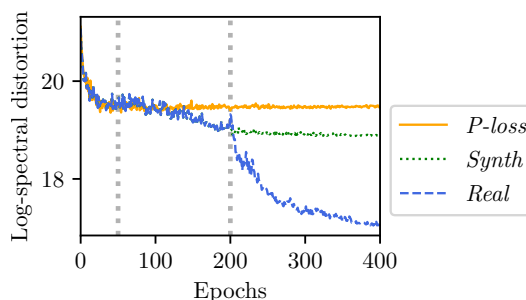


Figure 3. Spectral loss on out-of-domain sounds during training. The gray lines at 50th epoch and 200th epoch indicate the change in the weighting of the loss.

5.1 Quantitative Results

As a quantitative measure of the quality of sound match, we use log-spectral distortion (*LSD*) and the multi-scale spectral loss (*Multi*). We also calculate the L1 parameter loss (*Param*) for in-domain sounds. The results are shown in Table 1. The *P-loss* model achieved the best performance in terms of parameter loss, but performed poorly compared to other models in terms of spectral measures. This suggests that parameter loss is an inadequate criterion for match quality. The *Synth* model performed the best for matching the spectra of in-domain sounds, but the *Real* model was superior for out-of-domain data. This result shows that the fine-tuning was effective in transferring the knowledge learned from in-domain sounds to out-of-domain sounds. Since the out-of-domain data better represents query sounds in real-life applications, the *Real* model is the most promising for sound matching.

To examine the effects of the training procedures, we monitored the LSD for the out-of-domain validation set during training. This is shown in Figure 3. We can see that introducing the spectral loss from the 50th epoch caused a gradual decrease in LSD for the models *Synth* and *Real*. After the 200th epoch, the *Real* model was trained using out-of-domain data. From the sharp drop in LSD after this point, we can see the effectiveness of using out-of-domain sounds. The *P-loss* model was ineffective in improving spectral loss beyond a certain point.

5.2 Subjective Evaluation

For subjective evaluation of the match quality, paired comparison was conducted with reference stimuli via a crowd

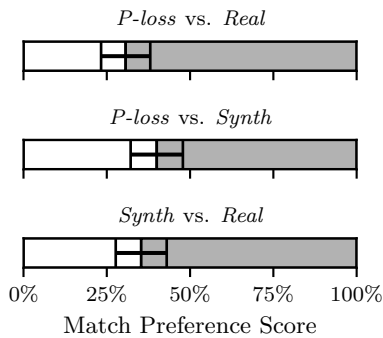


Figure 4. Results of subjective evaluation of the match quality. The three models were compared in a round-robin manner. Error bars denote 95% confidence intervals.

sourcing system. In total, 25 listeners from various backgrounds answered 18 questions. In each question of the test, listeners were exposed to the original target sound and the matches created by two models. The listeners answered which match sounded more similar to the target sound. The order in which the two matches were presented was randomized. The target sounds were randomly chosen from the test set of the out-of-domain data, as they better represent queries in real-life applications of sound matching than in-domain sounds.

Results of the preference test is shown in Figure 4. First, the *P-loss* model was compared with the *Real* model. The audio outputs of the *Real* model was preferred more frequently than the *P-loss* model, indicating that the proposed method of using spectral loss and using out-of-domain sounds during training was effective in producing perceptually better matches, compared to the conventional method of optimizing only parameter loss. Second, the models *P-loss* and *Synth* were compared. The *Synth* model performed better than the *P-loss* model, indicating that the use of spectral loss was effective in producing perceptually better matches. Finally, the models *Real* and *Synth* were compared. The *Real* model performed better than the *Synth* model. This highlights the importance of fine-tuning the estimator model with data that more closely resembles the query.

We show examples of sound matching in Figure 5. We can note that while all models perform comparably well on most in-domain sounds, the *P-loss* model tended to fail in reproducing features such as pitch and spectral envelope of the the out-of-domain sounds. The first out-of-domain sound in Figure 5 is a brass sound with rich harmonics, but only the *Real* model successfully produced timbre resembling a brass sound. For the second out-of-domain sound, the *P-loss* and *Synth* models failed to estimate the pitch, resulting in a sound with lower pitch.

6. CONCLUSIONS AND FUTURE DIRECTIONS

We presented a novel method of synthesizer sound matching by implementing the synthesizer using differentiable

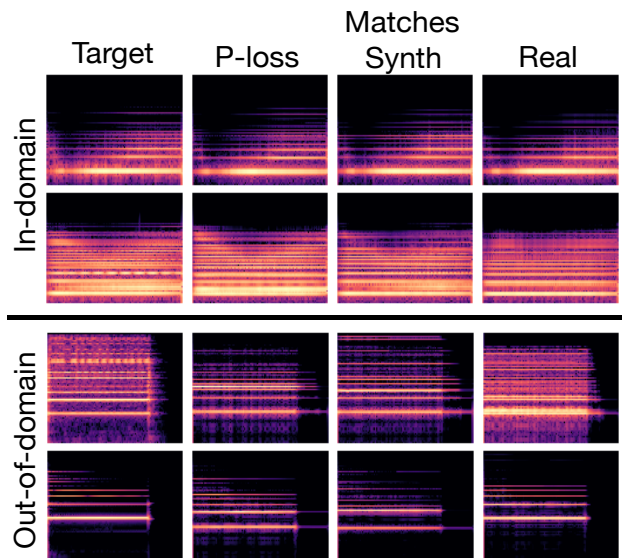


Figure 5. Examples of sound matching results. The three models were used to match the same target sounds taken from the in-domain and out-of-domain test set. We display the spectrogram (frequency is log-scaled, 0-8000Hz) of the audio output.

DSP. The proposed method is able to directly optimize spectral loss in an end-to-end manner. Furthermore, the model is able to utilize real-world sounds during training. By pre-training the model with parameter loss on the synthetic data and fine-tuning on real-world sounds with spectral loss, we showed that the proposed method can provide perceptually better matches to real-world sounds compared to baseline models.

While dynamic parameters were estimated frame-by-frame in our experiments, typical synthesizers use ADSR and LFO modules to model the dynamics of parameters. Such modules are required for intuitive control of the dynamics and playing notes with different length, but the use of such modules has been left unaddressed by our work. One solution is to estimate the envelope parameters from the frame-wise synthesis parameters [23]. Another solution is to implement such modules in a differentiable manner and use them during training. Preliminary experiments using differentiable envelope modules yielded promising results, although the match is less accurate for real-world sounds due to the simplification of the dynamics.

Another direction is to experiment with different synthesis techniques and audio effects. Preliminary experiments on using spectral loss to estimate the parameters of an FM synthesizer was shown to be less successful. This may be due to the fact that FM synthesis creates many in-harmonic partials resulting in a complex spectrum. Recent research suggests that deep audio embeddings may be a better distance metric than multi-scale spectral loss for complex synthesizer sounds [24]. Such alternative criteria for the match quality is worth considering not only for improving the estimation quality, but also for providing unique matches that capture a certain feature of the query.

7. REFERENCES

- [1] O. Barkan, D. Tsiris, N. Koenigstein, and O. Katz, “InverSynth: Deep estimation of synthesizer parameter configurations from audio signals,” *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 2385–2396, 2019.
- [2] M. J. Yee-King, L. Fedden, and M. D’Inverno, “Automatic programming of VST sound synthesizers using deep networks and other techniques,” *IEEE Trans. on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, 2018.
- [3] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, “Universal audio synthesizer control with normalizing flows,” in *Proc. of the 22nd Int. Conf. on Digital Audio Effects*, 2019.
- [4] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, “DDSP: Differentiable Digital Signal Processing,” in *Proc. of the Int. Conf. on Learning Representations*, 2020.
- [5] P. Esling and C. Agon, “Multiobjective time series matching for audio classification and retrieval,” *IEEE Trans. on Audio, Speech and Language Processing*, vol. 21, no. 10, pp. 2057–2072, oct 2013.
- [6] M. Cartwright and B. Pardo, “SynthAssist: Querying an audio synthesizer by vocal imitation,” in *Proc. of the Int. Conf. on New Interfaces For Musical Expression*, 2014, pp. 363–366.
- [7] A. Horner, J. Beauchamp, and L. Haken, “Machine Tongues XVI: Genetic algorithms and their application to FM matching synthesis,” *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, mar 1993.
- [8] K. Tatar, M. Macret, and P. Pasquier, “Automatic synthesizer preset generation with PresetGen,” *Journal of New Music Research*, vol. 45, no. 2, pp. 124–144, 2016.
- [9] K. Itoyama and H. G. Okuno, “Parameter estimation of virtual musical instrument synthesizers,” in *Proc. of the 40th Int. Computer Music Conf.*, 2014, pp. 1426–1431.
- [10] M. A. Martinez Ramirez, O. Wang, P. Smaragdis, and N. J. Bryan, “Differentiable signal processing with black-box audio effects,” in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2021, pp. 66–70.
- [11] A. Défossez, N. Zeghidour, N. Usunier, L. Bottou, and F. Bach, “SING: Symbol-to-instrument neural generator,” in *Advances in Neural Information Processing Systems*, 2018, pp. 9041–9051.
- [12] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, “Neural music synthesis for flexible timbre control,” in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 2019, pp. 176–180.
- [13] A. M. Sarroff and M. Casey, “Musical audio synthesis using autoencoding neural nets,” *Proc. of the 40th Int. Computer Music Conf., ICMC*, pp. 1411–1417, 2014.
- [14] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, “Neural audio synthesis of musical notes with WaveNet autoencoders,” in *Proc. of the 34th Int. Conf. on Machine Learning*, 2017, pp. 1068–1077.
- [15] X. Serra, “Musical sound modeling with sinusoids plus noise,” in *Musical Signal Processing*, C. Roads, S. Pope, A. Picialli, and G. D. Poli, Eds., 1997, pp. 91–122.
- [16] M. Michelashvili and L. Wolf, “Hierarchical timbre-painting and articulation generation,” in *Proc. of the Int. Society for Music Information Retrieval Conf.*, 2020, pp. 916–922.
- [17] J. Engel, R. Swavely, A. Roberts, L. Hantrakul, and C. Hawthorne, “Self-supervised pitch detection by inverse audio synthesis,” in *Workshop on Self-Supervision in Audio and Speech at the 37th Int. Conf. on Machine Learning*, 2020.
- [18] B. Kuznetsov, J. D. Parker, and F. Esqueda, “Differentiable IIR filters for machine learning applications,” in *Proc. of the 23rd Int. Conf. on Digital Audio Effects*, 2020, pp. 297–303.
- [19] S. Nercessian, “Neural parametric equalizer matching using differentiable biquads,” in *Proc. of the 23rd Int. Conf. on Digital Audio Effects*, 2020, pp. 265–272.
- [20] H. Kato, Y. Ushiku, and T. Harada, “Neural 3D mesh renderer,” in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018, pp. 3907–3916.
- [21] S. Yao, T. M. H. Hsu, J. Y. Zhu, J. Wu, A. Torralba, W. T. Freeman, and J. B. Tenenbaum, “3D-aware scene manipulation via inverse graphics,” in *Advances in Neural Information Processing Systems*, 2018, pp. 1887–1898.
- [22] G. Wilson and D. J. Cook, “A survey of unsupervised deep domain adaptation,” *ACM Trans. on Intelligent Systems and Technology*, vol. 11, no. 5, Jul. 2020.
- [23] K. Jensen, “Envelope model of isolated musical sounds,” in *Proc. of 2nd COST G-6 Workshop on Digital Audio Effects*, 1999.
- [24] J. Turian, J. Shier, G. Tzanetakis, K. McNally, and M. Henry, “One billion audio sounds from GPU-enabled modular synthesis,” *arXiv preprint arXiv:2104.12922*, 2021.

A MODULAR SYSTEM FOR THE HARMONIC ANALYSIS OF MUSICAL SCORES USING A LARGE VOCABULARY

Andrew McLeod

EPFL

andrew.mcleod@epfl.ch

Martin Rohrmeier

EPFL

martin.rohrmeier@epfl.ch

ABSTRACT

The harmonic analysis of a musical composition is a fundamental step towards understanding its structure. Central to this analysis is the labeling of segments of a piece with chord symbols and local key information. In this work, we propose a modular system for performing such a harmonic analysis, incorporating spelled pitches (i.e., not treating enharmonically equivalent pitches as identical) and using a very large vocabulary of 1540 chords (each with a root, type, and inversion) and 70 keys (with a tonic and mode), leading to a full harmonic characterization similar to Roman numeral analysis. Our system’s modular design allows each of its components to model an aspect of harmony at an appropriate level of granularity, and also aids in both flexibility and interpretability. We show that our system improves upon a state-of-the-art model for the task, both on a previously available corpus consisting mostly of pieces from the Classical and Romantic eras of Western music, as well as on a much larger corpus spanning a wider range from the 16th through the 20th centuries.

1. INTRODUCTION

The analysis of the harmonic content of a musical composition or performance is fundamental to the field of MIR. It can take many forms, depending on the input format and desired output representation and specificity, but typically involves two basic steps: the segmentation of a musical input, and the labeling of each segment with a harmonic symbol from some vocabulary. The vocabulary is typically either a key or a chord symbol, but more recent work has begun to investigate a joint approach, identifying both.

A musical key can be defined by its tonic pitch and a mode. In this work, like most previous work, we only consider major and minor mode, but many others exist in practice (see [1] for a recent discussion). Most work on key detection, be it from audio [2–4] or some symbolic format [5–8], only classifies each piece as having a single key (i.e., its global key), disregarding any modulation to different local keys that might occur. One reason for this

is simply the lack of labelled datasets with specific local key information, and the expertise required to produce (and evaluate) such annotations. More recently, however, some systems have addressed the problem of local key changes, incorporating a segmentation step into the key detection pipeline [9, 10], or producing a continuous distribution over local keys throughout the duration of a piece [11].

Chord detection (also called chord transcription) and chord sequence prediction are very widely researched tasks in MIR, though the vocabulary used can vary dramatically. At its most basic, each chord in a vocabulary has a root pitch and a chord type (major triad, minor triad, etc.). Much existing work only has a limited chord vocabulary of size 24 (12 semitone pitch classes and either major or minor triads) [12, 13], often due again to the difficulty of creating datasets with more specific chord types (and the difficulty of inducing spelled pitches from MIDI or audio input). More recently, some work has included common additional triad types, such as diminished and augmented triads [14], as well as various 7th chords, and further extensions, like suspended or 9th chords [15, 16]. Additionally, each chord can have an inversion—describing which of its pitches is its bass note—the inclusion of which is also becoming more common [17].

In this work, we take as input a musical score, and label each segment with both a key and a chord symbol, including a very large vocabulary of 12 different chord types and inversions for each. As in standard Roman Numeral Analysis (RNA; [18]), this allows each chord’s root pitch to be interpreted relative to the corresponding key’s tonic pitch (including any local key modulations). In fact, our system’s output is nearly equivalent to a full RNA, lacking only altered chordal tones such as suspensions, and pedal tones, which we intend to include in future work. We also use spelled pitches (where an $A\sharp$ is a different pitch from a $B\flat$), which is still uncommon in existing work.

A few prior models have been proposed for the joint labelling of keys (including modulation) and chords. Statistical models have been used for the purpose [19–21], though they use an enharmonic MIDI pitch representation (where $A\sharp$ and $B\flat$ are equivalent), a small vocabulary of chord types, and no inversions. Structurally, however, these models have somewhat inspired the design of our system (though they use Markov models instead of neural networks), explicitly modelling chord and key changes in a sequential fashion. More recently, deep learning models have also been proposed which use a large vocabulary



of chord types as well as inversions. In [22], a transformer model is used, though it takes a piano-roll representation as input, and therefore uses the more reduced MIDI pitch representation rather than spelled pitch. This model is extended in [23] using the same input format, but now using spelled pitch output, though still using a reduced set of pitch classes compared to ours. Finally, in [24], a convolutional network is proposed for the task. This model uses the same pitch representation as ours, and roughly the same chord vocabulary, so we use it for comparison.

These existing deep learning models consider sequential information either at the frame level (an eighth or 16th note) [22, 24], or by grouping frames together into larger blocks [23]. Our modular design is such that each component models one specific aspect of the full harmonic analysis task at the appropriate level, receiving only input that is relevant to that aspect, and at the appropriate step length. For example, the component which models chord progressions sees only chord symbols as input, once per chord, while the component that performs the chord segmentation sees individual notes as input. Our hypothesis is that this design allows each component to better capture the patterns relevant to the task at hand.

The modular design also gives a practical, benefit over the single-model end-to-end design that is common in existing work. Its output is highly interpretable, which has helped significantly in its development (we note a few specific instances of this in the paper). In particular, it enabled the following development process: (1) locate mislabeled segments in its output; (2) examine the outputs of each component for that segment, comparing them to our intuition about the analysis (this is easy because each component corresponds to a well-defined aspect of the analysis); and (3) integrate additional features into the corresponding component, depending on our analysis.

2. PROPOSED MODEL

2.1 Vocabulary

We use a large vocabulary of chords and keys as they appear in scores, taking on a full characterization as used in music theory [18]. Chord roots may be any pitch A–G, double-flat to double-sharp (35 total), and we include 12 chord types: major, minor, augmented (each as a triad, or with a major or a minor 7th), and diminished (as a triad, or with a minor or diminished 7th). Each chord can be in any inversion (3 for triads, 4 for tetrads), for a total of 1540 possible chords. Keys may be major or minor with the same pitch range, for a total of 70 possible keys. Our model does not output applied chords (e.g., secondary dominants like V/V) directly. Rather, we treat them as brief, potentially recursively embedded, key changes as in [25].

2.2 Overview

Our system is composed of 6 modules, each with a well-defined input and output (see Figure 1). The system’s input is a sequence of notes N ordered temporally by onset position, where notes with equal onset position are ordered by

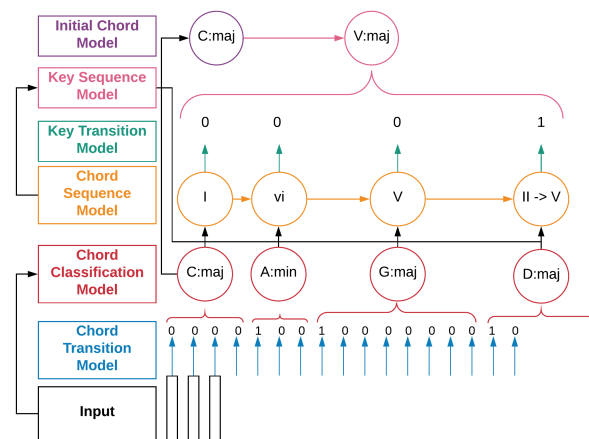


Figure 1. Overview of our fully integrated system. Each component depends on the component directly below it, and additional dependencies are indicated by arrows.

increasing pitch. The i th note in this sequence is denoted by n_i , and a note’s onset position is denoted by $on(n_i)$. Specific implementation details (e.g., the precise encoding of each note) are explained in Section 2.3.

The *Chord Transition Model* (CTM) takes this sequence N and predicts whether each note is the first (temporally) of a new chord. We denote ct_i as the i th output of the CTM. Of these outputs, the first (ct_0) is set to 1, and any ct_i where $on(n_i) = on(n_{i-1})$ are set to 0. Thus, chord transitions correspond with vertical slices in the musical score.

The *Chord Classification Model* (CCM) takes a sub-sequence of notes from N , and outputs a distribution over all 1540 possible absolute chord symbols for it. We denote the sub-sequence from the i th to the j th note as $n_{i..j}$.

The *Chord Sequence Model* (CSM) takes a sequence of relative chord symbols (whose roots are represented as an interval above the tonic, not as an absolute pitch class), and outputs a distribution over the next relative chord in the sequence. We denote the i th relative chord symbol in this sequence as c_rel_i . At inference time, it’s output always assumes that the key will not change on the next chord, and is unused in the case of a key change.

The *Key Transition Model* (KTM) takes as input a sequence of relative chord symbols starting from the beginning of the piece ($c_rel_{0..i}$, $i > 1$, denoting the first $i + 1$ relative chord symbols), and predicts whether chord c_rel_i is in a different key than c_rel_{i-1} . At inference time, c_rel_i is initially represented relative to the key tonic of c_rel_{i-1} . Then, in the case of a key change, the hidden state of the KTM (if there is one) is reverted to its previous state, and given c_rel_i in the new key.

The *Key Sequence Model* (KSM) takes as input the same sequence of relative chord symbols as the KTM ($c_rel_{0..i}$, $i > 1$), and outputs a probability distribution over a new key for c_rel_i , including both the mode (major or minor) and a tonic pitch class (represented as an interval over the previous key’s tonic pitch class). It’s output is only used when there is a key change and ignored otherwise. Similarly to the KTM, at inference time, c_rel_i is initially rep-

resented relative to the key tonic of $c_{rel_{i-1}}$. Then, after the key change, the hidden state of the KSM (if there is one) is reverted to its previous state, and given the updated c_{rel_i} in the new key.

Finally, the *Initial Chord Model* (ICM) outputs a prior probability distribution over the first relative chord of a piece c_{rel_0} given a mode, and is used in combination with the first absolute chord symbol to generate a distribution over possible first absolute keys for the annotation. For example, the ICM may determine that the c_{rel_0} in major mode has a 0.4 probability of being a I chord. It is used as a replacement for the CSM for the first chord.

2.3 Implementation

Due to the modular design of our system, the precise implementation of each model (including the representations used therein) is very flexible. For example, although the CTM is well-defined to take as input a sequence of notes in a particular order, the precise way in which each note is represented is left as an implementation detail, and can easily be changed without affecting the other models. Such details, as well as the design of each model individually, are described in this section.

As input to the CTM, each note n_i is represented as the concatenation of one-hot vectors (or scalars) encoding certain musical features of each note: the note's pitch class A–G, double-flat to double-sharp (one-hot of length 35); the note's octave 0–11, where C4 is in octave 6 to allow for negative octaves (one-hot of length 12); the note's normalized MIDI pitch height (0–1), where C-1 (MIDI note 0) = 0 and G9 (MIDI note 127) = 1 (scalar); the metrical levels of the note's onset and offset positions (downbeat, beat, sub-beat, or other; two one-hots of length 4); the duration of the note, where a whole note has duration 1 (scalar); and the duration from the note's onset to that of the previous note and that of the following note, again measured in whole notes (two scalars).

Our CTM is a neural network composed of a single feed-forward layer followed by a Bi-LSTM (each using ReLU activation). At each step, the LSTM's outputs are concatenated together and fed into two additional feed-forward layers (the first with ReLU activation and the last with sigmoid activation to produce a probability value).

As input to the CCM, each note vector is embedded into its musical context in two ways. First, we append to this sub-sequence the vectors of the two notes on each side of this sub-sequence (or vectors of zeros if this goes beyond the range of the piece). Second, we append to each note vector more musical features related to the note's context: onset and offset position relative to the chord window on a linear scale where the beginning of the chord is at position 0 and the end of the chord is at position 1 (2 scalars); duration as a proportion of the chord's duration (scalar); relative normalized pitch, where instead of C-1 being 0 and G9 being 1, the minimum pitch of a note in the window is 0 and the maximum pitch is 1 (scalar); and relative octave, where the octave of the lowest note in the window is subtracted from each note's octave before embedding (one-hot

of length 12). The relative pitch and octave are examples of where our system's interpretability helped in its design: we noticed that the CCM was struggling to output the correct inversions when a passage doesn't contain any notes in a low octave (it had learned to depend on bass notes for this), and therefore added the relative features, which helped in that regard.

Its output is a distribution over all 1540 absolute chord symbols. It would be possible to treat each aspect of a chord symbol (root, type, and inversion) as a separate feature of each chord, and have the CCM output one distribution over each, as has been done in previous work (e.g., [24, 26]). However, while this approach makes sense in terms of reducing the size of a model, it doesn't make sense conceptually: there may be a situation in which the model sees a C in the bass, and thinks the chord is either a C major triad in root position or an A minor 7th chord in 1st inversion. In cases such as this, it is important that every feature of a chord is considered holistically as a unit, rather than potentially classifying the chord as a C minor 7th chord in 1st inversion.

Our CCM is a neural network composed of a single feed-forward layer followed by a Bi-LSTM (each using a ReLU activation). The outputs from the last LSTM state in each direction are concatenated together and fed into a single feed-forward layer with softmax activation.

The CSM, KTM, and KSM all take the same input, where each relative chord is encoded as a vector of concatenated musical features: the root pitch class and bass note pitch class, each represented as the interval above the key tonic on the line of fifths (-14–14; two one-hots of length 29); the chord type (one-hot of length 12); the inversion (one-hot of length 4); the metrical levels of the chord's onset and offset positions (same as for the note encoding; two one-hots of length 4); the duration of the chord in whole notes (scalar); and a flag indicating the current key's mode (1 for major; 0 for minor). At every chord at which there is a key change, we also append a key change vector, which encodes the following: the tonic of the new key, represented as the interval above the previous key's tonic on the line of fifths (-14–14; one-hot of length 29); the mode (major or minor; one-hot of length 2); and a flag indicating that this is a key change (binary 1). For each chord at which the key does not change, this key change vector is filled with all 0s.

The CSM's output is a distribution over chord symbols consisting of a root (-14–14 on the line of fifths from the key tonic), chord type, and inversion (1276 chords in total). Some of these chords may correspond with a valid absolute chord symbol (e.g., one with relative root -14 in the key of B $\flat\flat$). These are simply ignored. Likewise, some valid chords are not covered by the CSM's range (e.g., a B \sharp chord in the key of B $\flat\flat$, which is exceedingly unlikely [27]). The prior for these chords is 0. Similarly, the KSM's output is a distribution over key symbols consisting of a tonic (-14–14 on the line of fifths from the previous key tonic) and a mode (58 keys in total). Invalid (e.g., 14 fifths below B $\flat\flat$ major) and uncovered (e.g., 20 fifths

above B \flat major) key changes are treated in the same way.

Each of these models has a single feed-forward layer, followed by an LSTM whose last output is sent through another feed-forward layer (all with ReLU activations). A final feed-forward layer is then used, with softmax activation in the CSM and KSM, and sigmoid in the KTM.

Since there can be many reasonable choices for the next chord in a sequence, we noticed when inspecting the system’s outputs that the CSM’s distribution was more flat than desired. Therefore, we compare two additional versions of it: an inversion invariant CSM-I, which outputs a distribution over relative roots and chord types (348 chords in total), in which case probabilities are shared between different inversions of the same chord; and a triad-reduced CSM-T, which includes the CSM-I’s inversion invariance, and further shares weights between chords built upon the same triad (e.g., a V7, Vmaj7, and V all share outputs).

The ICM is a much simpler model than the rest, and we simply count the proportion of each chord—grouped by chord type, inversion, and relative root—as the first chord of a piece in our training data. We then apply additive smoothing to this estimate, adding $\frac{1}{1540}$ (where 1540 is the number of chords in our vocabulary) to each count.

2.4 Inference

Given the interactions between the modules of our system, the inference procedure required to use it to label a score is not trivial. However, since each component can be treated as a black box, once such a process is defined, we can flexibly use different modules interchangeably. This section gives an overview of the inference process, which, at its core, is the process of finding the most probable labeling of the musical score according to our system. We explore the search space iteratively using beam search decoding.

First, we run the CTM on the full input sequence N , generating ct_i for each note n_i . Next, we find all valid chord windows for the piece. For a chord window from $n_{i\dots j}$ to be valid, six conditions must be satisfied: (1) $ct_i \geq ct_{min}$, where ct_{min} is a minimum threshold for a chord change; (2) $ct_{j+1} \geq ct_{min}$ (unless $j = |N| - 1$); (3) $ct_k \leq ct_{max}$ for all $i < k \leq j$ (ct_{max} is a maximum threshold for allowing a note to *not* be a chord change); (4) the duration of the resulting chord window (i.e., $on(n_{j+1}) - on(n_i)$) must be less than a maximum value C_dur_{max} ; (5) a valid chord window exists which ends at n_{i-1} (unless $i = 0$); and (6) a valid chord window exists which begins at n_{j+1} (unless $j = |N| - 1$). (1)–(3) together ensure that the CTM is not ignored; (4) ensures that the resulting chord labeling is well-formed (extremely long passages with no chord changes are quite rare); and (5) and (6) ensure that each chord window can be part of a complete labeling of the musical score.

Having found all possible chord windows, the search process involves finding the most probable complete and labeled path through the score. A complete path C is a sequence of consecutive chord windows $c_0\dots|C|-1$, where $c_m = n_{i_m\dots j_m}$ ($i_0 = 0$, $j_{|C|-1} = |N| - 1$, and $\forall m, i_m = j_{m-1} + 1$). A labeled path is a complete path where each

chord window c_m therein is labeled by assigning it a chord ($Ch(c_m)$) and a key ($K(c_m)$), with the constraint that no chords are repeated (i.e., $\forall m, Ch(c_m) \neq Ch(c_{m+1})$).

One exception to the above constraints is the *merge* rule, which allows the CCM to override the CTM’s ct_{min} threshold in some cases. It is legal for two consecutive chord windows c_m and c_{m+1} to be merged if: (1) the resulting chord window’s duration ($on(n_{j_{m+1}+1}) - on(n_{i_m})$) is still less than C_dur_{max} ; and (2) $Ch(c_m) = Ch(c_{m+1})$ (which only occurs if the CCM assigns each a high probability). The merge step can be repeated as many times as possible as long as the two constraints are still met. This rule is another example of the interpretability of our system helping in its development: we noticed that the CTM was over-segmenting the input, and the thresholds were difficult to tune. However, the CCM’s outputs were relatively accurate. Thus, we created the merge rule, which allows us to tune the CTM thresholds to over-transition, letting the more accurate CCM merge windows later.

Given a complete labeled path through a score, its probability is the product of the probabilities of each chord window c_m , where an individual chord window’s probability is calculated as a product of its CTM probability $P_{ct}(c_m)$, its CCM probability $P_{cc}(c_m)$, its KTM probability $P_{kt}(c_m)$, and its sequence probability $P_{seq}(c_m)$, as shown in Eqn. 1. The exponent $|c_m|$ is used so that each path’s probability is a product of an equal number of probabilities, no matter the number of chord windows (without this exponent, a path with fewer, longer windows would be preferred).

$$P(c_m) = P_{ct}(c_m)(P_{cc}(c_m)P_{kt}(c_m)P_{seq}(c_m))^{|c_m|} \quad (1)$$

A window’s CTM probability is calculated as in Eqn. 2 (ct_{j_m+1} is ignored if $m = |C| - 1$).

$$P_{ct}(c_m) = ct_{i_m} ct_{j_m+1} \prod_{k=i_m+1}^{j_m} (1 - ct_k) \quad (2)$$

A window’s CCM probability is the CCM prior for the range $n_{i_m\dots j_m}$ for the assigned chord $Ch(c_m)$. A window’s KTM probability is likewise taken directly from the KTM: it is either the KTM’s output at that window (if $K(c_m) \neq K(c_{m-1})$), or one minus the KTM’s output (in all other cases). The first KTM probability ($P_{kt}(c_0)$) is always 1, since the key will never change on the first chord.

A window’s sequence probability (the probability of a chord and key given the previous) is taken from the ICM ($P_{ic}(c_m)$) for $m = 0$, the KSM ($P_{ks}(c_m)$) if there is a key change, or the CSM ($P_{cs}(c_m)$) otherwise, as shown in Eqn. 3. Initially, the system predicted far too many key changes, and inspecting its output, made the cause clear: since the KSM outputs a distribution over 70 keys, while the ICM and CSM output distributions over more than 1000 chords, the KSM’s outputs will naturally be greater. Thus, we introduced a parameter α , ensuring that the values output by each model lie in a similar range.

$$P_{seq}(c_m) = \begin{cases} P_{ic}(c_m) & \text{if } m = 0 \\ P_{ks}(c_m)^\alpha & \text{if } K(c_m) \neq K(c_{m-1}) \\ P_{cs}(c_m) & \text{otherwise} \end{cases} \quad (3)$$

3. EXPERIMENTS

3.1 Setup

We use two different corpora: (1) an internal corpus of harmonic annotations, including the publicly available Mozart [28] and ABC [29] corpora, and additional pieces which are private, but pending future release; and (2) functional-harmony (F-H) [24], consisting of a combination of prior existing corpora: TAVERN [30], BPS-FH [22], and Roman Text [31]. The internal corpus contains a much wider range of composers: 742 pieces from the 16th to the 20th centuries, with J.S. Bach, Couperin, Grieg, Beethoven, Schütz, Mozart, Corelli, Chopin, Kozeluh, Monteverdi, Mendelssohn, and Schubert all having more than 20 pieces, and 99 by other composers. F-H is smaller: 201 pieces in total, with 119 by Beethoven, 29 by Schubert, 24 by J.S. Bach, and 19 by other composers. We treat the two corpora separately, and in each, randomly take 80% for training, and 10% each for testing and validation.

For evaluation, we use a type of Chord Symbol Recall (CSR) [32]: for each piece, the proportion of time during which the estimated label matches the ground truth label. Given our large vocabulary, we apply CSR at various levels of specificity, similar to [24]¹. We report CSR with regards to the chord root, chord root+type, chord root+type+inversion (the full chord), key (ignoring the chord), and full (including chord and key). We intend to investigate more advanced metrics (e.g., where some label substitutions are penalized less than others) in future work.

All components of our system were trained with *Adam* [33] and a learning rate of 0.001. A scheduler cut the learning rate in half when the validation loss didn't improve for 10 epochs, and training was stopped when the validation loss failed to improve for 20 epochs. When training the CCM, we transposed each input and target by $-7-7$ fifths (a similar data augmentation technique was used by [24]).

For this study, we did not perform a large grid search over deep model structures (e.g., more feed-forward layers). Rather, we grid searched over layer sizes of 64, 128, and 256 for each of the feed-forward and LSTM layers of each model. The system-wide inference parameters were set using a grid search on the validation set of each corpus, resulting in ct_{max} and ct_{min} of 0.3 and 0.4 for the F-H corpus, and 0.35 and 0.55 for the internal corpus; $C_{dur_{max}}$ of 10 for both corpora (this parameter has little effect); and α of 30 for the F-H corpus and 50 for internal corpus. α has by far the largest effect of any parameter, where low values result in much more frequent key changes. We leave an investigation of the performance of each module for future work, concentrating here on overall performance.

3.2 Results

We compare versions of our system using the standard CSM, the inversion invariant CSM-I, and the triad-reduced CSM-T. As a baseline, we use the pre-trained model of [24], computing new results on our internal corpus, and

¹ In the metrics reported in [24], applied roots are not considered key changes as they are here, but instead included in the “Degree” metric.

	Model	Root	+Type	+Inv.	Key	Full
Internal	[24]	57.0	47.7	37.6	64.9	29.0
	CSM	76.6	68.8	62.1	66.9	44.7
	CSM-I	76.5	68.7	62.0	69.0	46.3
	CSM-T	77.6	70.0	62.8	70.2	46.9
F-H	[24] ²	—	—	—	—	42.8
	CSM	73.3	65.4	55.6	60.8	40.5
	CSM-I	75.0	66.8	56.9	67.0	44.6
	CSM-T	75.4	67.8	58.1	69.4	45.9

Table 1. The results of our system compared to the model of [24] on our internal corpus of annotations (top), and the functional-harmony meta-corpus used in [24] (bottom).

using the full CSR from [24] for the F-H corpus (the component-wise metrics are calculated differently). The results can be found in Table 1. Here, we can see that our modular system outperforms the baseline on both the internal corpus and, more notably, on the F-H corpus.

There seems to be a systematic difference between the annotations of the two corpora, given the relatively poor performance of the pre-trained baseline [24] on the internal corpus. We would expect this to be due to the wider range of styles, but while this does appear to play some role, the baseline actually performs *worse* (22.9 overall) on a subset of the internal corpus’ test set including only Beethoven pieces. We plan to more thoroughly investigate differences between the annotations of the two corpora in future work.

Interestingly, the baseline performs relatively well on key detection (even on our internal corpus), which points to one downside of our modular approach: the key depends on outputs from the other components, adding noise to the process, which isn’t a factor for the end-to-end baseline.

For our system, we see that the three versions perform similarly on the internal corpus, but the two invariant CSMs outperform the standard one—significantly on the F-H corpus. Interestingly, this difference mainly shows in a more accurate key labelling, rather than chord. Inspecting the results, we see that the standard CSM usually assigns a low probability to the more rare chords like minor 7th chords or inverted chords. This makes the model prefer key changes in these cases, which avoid using the CSM’s output distribution. The CSM-I and CSM-T avoid this problem due to their invariances. In future work, we intend to investigate alternate reductions where, for example, dominant 7th chords (which are quite common) are not reduced to major in the CSM-T.

Figure 3 shows the CSM-I’s chord classification performance on the internal corpus as a confusion matrix. It performs best on major triads, minor triads, and dominant 7th chords, which comprise 44%, 22%, and 18% of the corpus, respectively. For the less common chord types, when the root is correct, the system often misclassifies the chord

² The values in [24] are calculated slightly differently: applied chords are included in Root (rather than Key here), and Type and Inversion refer to only those features of a chord (in [24]), rather than in combination with those to their left. “Full” is comparable.

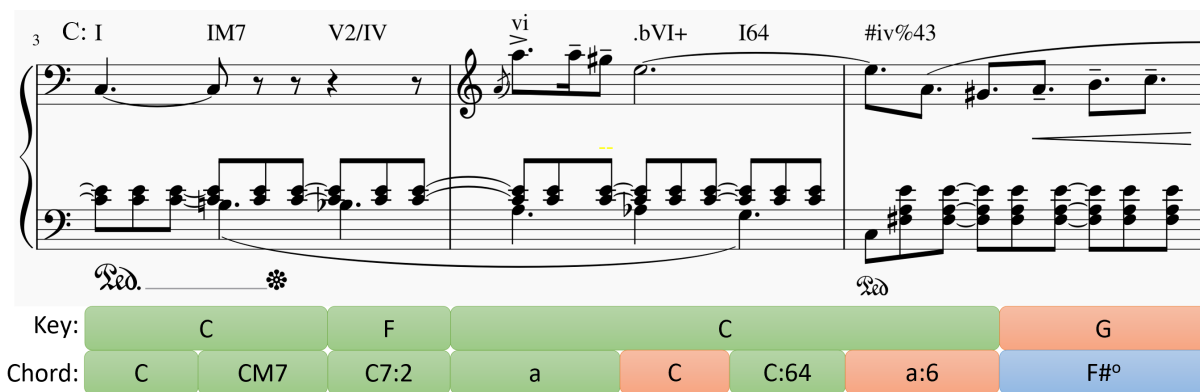


Figure 2. The CSM-I system’s output (below) for bars 3–5 of Grieg’s Notturmo, Op. 54, No. 4, given the annotations (above). Green indicates a correct label, red is incorrect, and blue is partially correct. Inversions are indicated by the figured bass after each colon.

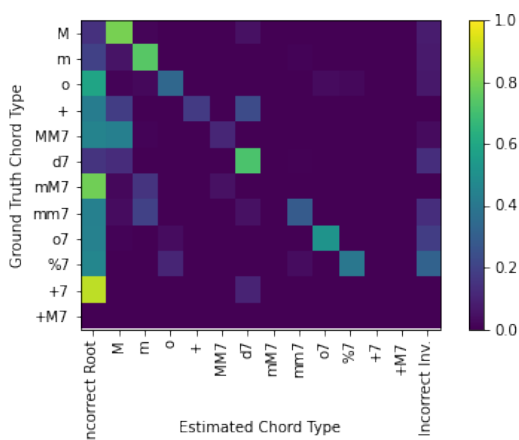


Figure 3. A confusion matrix of the CSM-I system’s chord classifications on the internal corpus. “Incorrect Inv.” indicates the proportion of otherwise correct labels with an incorrect inversion.

type as the corresponding triad (e.g., MM7 chords are classified as major triads). In the rightmost column, we see that if the system gets the correct root and type, the inversion is usually correct, with the largest proportion of errors occurring for diminished and half-diminished 7th chords.

Figure 4 shows CSR split by mode and chord type, where it performs slightly better in minor keys, particularly for minor triads and diminished triads and 7th chords. Overall, it achieves 44.6 CSR in minor keys but only 40.9 in major. Chord inversions also have a large effect on performance, with our system classifying 71.5% of root position chords correctly, but only 54.4%, 38.3%, and 40.5% of 1st, 2nd, and 3rd inversion chords correctly, respectively. This makes sense, because root position chords make up 38% of the corpus, while 1st, 2nd, and 3rd inversion chords comprise 25.4%, 8.8%, and 3.7% respectively.

An example output of the CSM-I is shown in Figure 2, for bars 3–5 of Grieg’s Notturmo, Op. 54, No. 4. The annotations (in C major) are shown above the staff, and outputs are below it. The outputs are correct in bar 3, finding the applied dominant V2/IV as a key change to F. In bar 4,

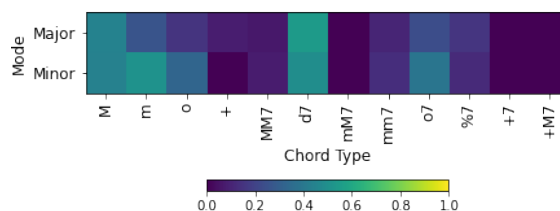


Figure 4. The CSM-I system’s full CSR (including key and chord) per mode and chord type on the internal corpus.

it incorrectly classifies the augmented A^b triad (a^bVI , uncommon prior to the late Romantic era) as a C major triad, ignoring the A^b . It over-segments bar 5, finding an inverted A minor triad (ignoring the F^\sharp) followed by an F^\sharp diminished triad (since it has missed the C from the downbeat), and modulates to G major (essentially, it has classified this chord as a $\sharp vii^o/V$, which is possible, but incorrect here).

4. CONCLUSION

We have presented a modular system for the harmonic analysis of musical scores. We showed how the system’s modularity, in addition to the theoretical benefit of modelling each aspect at the appropriate level, makes its output interpretable, describing where this property helped in its design. All code and models are available online.³

In future work, we will leverage the system’s modularity further. We intend to apply the system to MIDI and audio input, in which case we would only need to retrain the CTM and CCM with new data, while the other components can remain the same, or be supplemented with additional training data from the MIDI and audio files. We also intend to design a human-in-the-loop annotation tool using this system, where expert annotators can first get the system’s output, change some labels as they see fit, and then re-run the search process, constraining the system to a path that includes the manually corrected labels. This process could be faster than the current fully manual approach to harmonic annotation.

³<http://github.com/apmcleod/harmonic-inference>

5. ACKNOWLEDGMENTS

Research supported through the Swiss National Science Foundation within the project “Distant Listening – The Development of Harmony over Three Centuries (1700–2000)” (Grant no. 182811). This project was conducted at the Latour Chair in Digital and Cognitive Musicology, generously funded by Mr. Claude Latour.

6. REFERENCES

- [1] D. Harasim, F. C. Moss, M. Ramirez, and M. Rohrmeier, “Exploring the foundations of tonality: statistical cognitive modeling of modes in the history of Western classical music,” *Humanities and Social Sciences Communications*, vol. 8, no. 1, pp. 1–11, dec 2021.
- [2] Á. Faraldo, S. Jordà, and P. Herrera, “A Multi-Profile Method for Key Estimation in EDM,” in *AES International Conference on Semantic Audio*. Audio Engineering Society, jun 2017.
- [3] G. Bernardes, M. E. P. Davies, and C. Guedes, “Automatic musical key estimation with adaptive mode bias,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, mar 2017, pp. 316–320.
- [4] F. Korzeniowski and G. Widmer, “Genre-agnostic key classification with convolutional neural networks,” in *ISMIR*, 2018.
- [5] H. C. Longuet-Higgins and M. Steedman, “On Interpreting Bach,” *Machine Intelligence*, vol. 6, pp. 221–241, 1971.
- [6] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 2001.
- [7] D. Temperley, “What’s Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered,” *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.
- [8] D. Temperley and E. W. Marvin, “Pitch-Class Distribution and the Identification of Key,” *Music Perception: An Interdisciplinary Journal*, vol. 25, no. 3, pp. 193–212, mar 2008.
- [9] C. Weiss, H. Schreiber, and M. Mueller, “Local Key Estimation in Music Recordings: A Case Study Across Songs, Versions, and Annotators,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, oct 2020.
- [10] L. Feisthauer, L. Bigo, M. Giraud, and F. Levé, “Estimating keys and modulations in musical pieces,” in *17th Sound and Music Computing Conference*, 2020.
- [11] R. Lieck and M. Rohrmeier, “Modelling hierarchical key structure with pitch scapes,” in *ISMIR*, 2020.
- [12] H. Papadopoulos and G. Peeters, “Joint Estimation of Chords and Downbeats From an Audio Signal,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 1, pp. 138–152, jan 2011.
- [13] F. Korzeniowski and G. Widmer, “Improved chord recognition by combining duration and harmonic language models,” in *ISMIR*, 2018.
- [14] Y. Wu, T. Carsault, E. Nakamura, and K. Yoshii, “Semi-supervised Neural Chord Estimation Based on a Variational Autoencoder with Latent Chord Labels and Features,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 1–1, 2020.
- [15] T. Rocher, M. Robine, P. Hanna, and R. Strandh, “Dynamic chord analysis for symbolic music,” in *International Computer Music Conference*, 2009, pp. 41–48.
- [16] T. Carsault, A. McLeod, P. Esling, J. Nika, E. Nakamura, and K. Yoshii, “Multi-Step Chord Sequence Prediction Based on Aggregated Multi-Scale Encoder-Decoder Networks,” in *IEEE International Workshop on Machine Learning for Signal Processing, MLSP*, vol. 2019-October, 2019.
- [17] B. McFee and J. P. Bello, “Structured training for large-vocabulary chord recognition,” in *ISMIR*, 2017.
- [18] E. Aldwell, C. Schachter, and A. Cadwallader, *Harmony and voice leading*. Cengage Learning, 2018.
- [19] C. Raphael and J. Stoddard, “Functional harmonic analysis using probabilistic models,” *Computer Music Journal*, vol. 28, no. 3, pp. 45–52, 2004.
- [20] D. Temperley, “A Unified Probabilistic Model for Polyphonic Music Analysis,” *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, mar 2009.
- [21] C. Aitken, T. O’Donnell, and M. A. Rohrmeier, “A Maximum Likelihood Model for the Harmonic Analysis of Symbolic Music,” in *Proceedings of the Sound and Music Computing Conference (SMC)*, Cyprus, 2018.
- [22] T. P. Chen and L. Su, “Functional harmony recognition of symbolic music data with multi-task recurrent neural networks,” in *ISMIR*, 2018, pp. 90–97.
- [23] T.-P. Chen and L. Su, “Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.
- [24] G. Micchi, M. Gotham, and M. Giraud, “Not all roads lead to rome: Pitch representation and model architecture for automatic harmonic analysis,” *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 42–54, 2020.

- [25] M. Rohrmeier, “The syntax of jazz harmony: Diatonic tonality, phrase structure, and form,” *Music Theory and Analysis (MTA)*, vol. 7, no. 1, pp. 1–63, 2020.
- [26] T.-P. Chen and L. Su, “Harmony Transformer: Incorporating Chord Segmentation into Harmony Recognition,” in *ISMIR*, nov 2019.
- [27] F. Moss, “Transitions of tonality: a model-based corpus study,” Ph.D. dissertation, 2019.
- [28] J. Hentschel, M. Neuwirth, and M. Rohrmeier, “The Annotated Mozart Sonatas: Score, Harmony, and Cadence,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 1–14, 2021.
- [29] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, “The annotated beethoven corpus (ABC): A dataset of harmonic analyses of all beethoven string quartets,” *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018.
- [30] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, “Theme and variation encodings with roman numerals (tavern): A new data set for symbolic music analysis,” in *ISMIR*, 2015.
- [31] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, “The romantext format: A flexible and standard method for representing roman numeral analyses,” in *ISMIR*, 2019.
- [32] C. Harte, “Towards automatic extraction of harmony information from music signals,” Ph.D. dissertation, Queen Mary University of London, 2010.
- [33] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

A DEEP LEARNING METHOD FOR ENFORCING COHERENCE IN AUTOMATIC CHORD RECOGNITION

Gianluca Micchi

Katerina Kosta

Gabriele Medeot

Pierre Chanquion

ByteDance

{name.surname}@bytedance.net

ABSTRACT

Deep learning approaches to automatic chord recognition of symbolic music have improved the state of the art, but they still face a common problem: how to deal with a vast chord vocabulary. The naive approach of writing one output class for each possible chord is hindered by the combinatorial explosion of the output size (~ 10 million classes). We can reduce this complexity by several orders of magnitude by treating each label (e.g. key or chord quality) independently. However, this has been shown to lead to incoherent output labels. To solve this issue we introduce a modified Neural Autoregressive Distribution Estimation (NADE) as the last layer of a Convolutional Recurrent Neural Network. The NADE layer ensures that labels related to the same chord are dependently predicted, and therefore, enforce coherence. The experiments showcase the advantage of the new model both in chord symbol prediction and functional harmonic analysis compared to the model that does not include NADE as well as state-of-the-art models.

1. INTRODUCTION

Harmony, together with counterpoint and form, is traditionally considered to be one of the three main parts of musical composition in Western classical tradition [1]. This tradition is based on what is known as the *tonal system*, that is, a “theoretical formulation of certain psychological or physiological constraints upon the perception of sounds and their combination” [2][p.206]. Their musical effect can be summarised in a few rules that are followed by most Western music (and also some non-Western music) [3].

Nevertheless, harmonic interpretation of music is complex due to its ambiguity. The same audio content can acquire different perceptual significance depending on its context: As a simple example, the chord symbols $A\sharp$ major and $B\flat$ major are acoustically indistinguishable but used in different contexts, hence the different spelling. Therefore, it is necessary to study the chord not as single entities but as

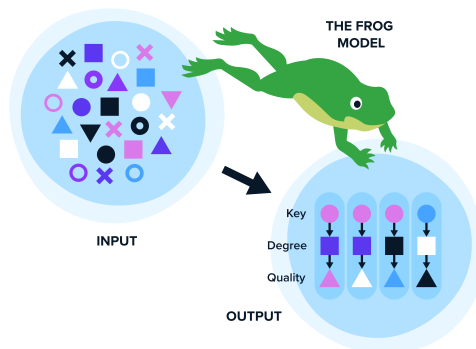


Figure 1: The proposed model, **frog**, analyses an input score in symbolic format and outputs the functional harmonic analysis through autoregressive predictions of its elementary components at each time step.

a progression. This is usually done with the help of the Roman numeral notation (RN), which describes every chord in relation to the local key. RNs provide insights into harmony theory by exposing its invariances and symmetries. They highlight the function of each chord inside the progression and, for this reason, Automatic Chord Recognition (ACR) with RNs is also known as functional harmonic analysis.

The problem of harmony has a long academic history, but remains central to modeling and understanding most music, including modern pop; indeed, harmony is one of the main categories in which submissions to the 2020 AI Song Contest were judged. [4]. Therefore, it is natural that computational analysis of harmony has attracted so much attention in the MIR community.

Previous work. There is a relatively vast body of work on ACR from audio signals (see [5] for a literature review on the topic). All these methods address chord symbol recognition instead of functional harmonic analysis. From the very first article published on the subject, the idea that has dominated the field is to interpret the audio signal using chroma features [6]. This amounts to identifying and annotating the pitch content of the audio signal at each given time frame. Given how close the resulting audio representation resembles a symbolic music score, it is a bit puzzling to see how little attention symbolic ACR has received.

There are only a few works, to our knowledge, that explicitly perform ACR on symbolic scores. Kröger et al. [7] collect a few early approaches, and a more recent one is



© Gianluca Micchi, Katerina Kosta, Gabriele Medeot, Pierre Chanquion. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Gianluca Micchi, Katerina Kosta, Gabriele Medeot, Pierre Chanquion, “A Deep Learning Method for Enforcing Coherence in Automatic Chord Recognition”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

presented in [8]. However the interest on the topic has increased in more recent years, probably driven also by the growing attention to symbolic music for music generation [4, 9–13]. Symbolic music makes a perfect entry point for the harder task of functional harmonic analysis because, with respect to audio data, it offers a much more direct representation of the musical content. There are two popular MIR approaches to functional harmonic analysis: One uses generative grammars [14, 15], the other a deep learning based data-driven approach that can learn rules more flexibly [16–19].

Known issues in ACR. One difficult issue that prevents naive rule-based approaches from being successful is the identification of non-chord tones: Music often contains notes that are not part of the core harmony and designing a system that knows when to ignore these anomalies is a complex task to define algorithmically [20]. Specifically for the task of functional harmonic analysis, there is also the problem of automatic key recognition, a subject that is well known in the community [21–24] but still largely unsolved, partly due to ambiguities in its definition [24].

Finally, an important issue is that the number of possible output classes is very large. Even considering only the most common chords, the possibilities easily exceed 100,000. This is because there is a combinatorial explosion due to the presence of several elementary labels associated with each possible chord. In the case of functional harmonic analysis, these labels are key, tonicisation, degree, quality, and inversion. For chord symbol prediction, instead, they are root, quality, and inversion. To reduce the dimensionality of the output space, it has been proposed to predict each label independently [16–18]. However, this approach often leads to incoherent output labels. For example, for a harmony that can be interpreted as either A minor or C major, a system without coherence could equally well output A *major* or C *minor*.

Our proposal. We propose a chord recognition algorithm (CRA) —which we nickname **frog** after the sound they make in Italian, *cra-cra*— that analyses chords through separate but coherent predictions of their elementary labels (see Fig. 1). This is achieved through the addition of a Neural Autoregressive Distribution Estimator (NADE) [25–27] to a CRNN architecture. At each time step, we provide the NADE with a fixed ordering of the chord’s elementary labels from which it sequentially samples, conditioning each sampling probability on the outcome of the previous labels. We release the code open-source and make it available at [28].

We evaluate the output of **frog** against two state-of-the-art models on a large dataset of functional harmonic analyses, containing over 300 scores that span over two centuries, from Monteverdi to Brahms. Due to the similarity in data representation, mentioned above, we believe our system to be extensible to the audio domain [17, 19].

2. DATA REPRESENTATION AND CORPUS

In RN notation, each chord is defined by its relation with the tonic of the local key. The basic components of RNs



Figure 2: Example RN analysis of the Prelude in C from JS Bach’s Well-Tempered Clavier, book first.

are key, degree of the scale on which the chord is built (expressed in Roman numerals), quality of the chord (i.e., the type of triad plus any possible extension), and inversion (i.e., which of the notes is the lowest). For example, from the RN analysis in Fig.2 we see the annotation V65 at the third measure. In the key of C (see first measure), this corresponds to a G (fifth degree of the scale) dominant seventh chord in first inversion (numerals 65).¹

Sometimes, chords are borrowed from other keys for a very short period of time and introduce some colouring in the harmonic progression. For example a D7 chord contains an F♯. Whenever we find such a chord in the key of C resolving to a G chord we identify it as a dominant chord borrowed from the neighbouring key of G and encode it as V7/V. Those borrowed chords are known as tonicised chords, and the tonicisation defines the relation between the local key and the temporary tonic [24]. The boundaries of this relation are sometimes blurry. As Kostka and Payne put it, "The line between modulation and tonicization is not clearly defined in tonal music, nor is it meant to be" [30].

The tonicisation completes the set of elementary labels that we use to describe a chord in the RN notation.

RN encoding. The simplest data encoding for RN requires 24 keys, 7 degrees, 7 tonicisations, and 4 inversions per each quality of chord. In our analyses we use 10 chord qualities: 4 triads (major, minor, diminished, and augmented), 5 sevenths (major, dominant, minor, half-diminished, and diminished), and augmented sixth.

When predicting all these labels at once, their sizes multiply to make a total of 47k possible output classes. If one wants to add pitch spelling, support for alterations both in degree and in tonicisation, and a direct prediction of the root, the total number of combinations climbs up to 22 millions [18]. Also, while the ten qualities cover most of the cases in classical music, making up for 99.98% of the dataset we consider, they don’t even come close to describing the wealth of extensions and colourings that are commonly used in jazz music [33]. In short, it is not desirable to deal directly with such a combinatorially explosive situation. Making individual predictions for each of the elementary labels that form the chord and then combining them together, instead, results in a summation of their output sizes, rather than a multiplication, making the problem tractable again.

Chord symbols. From the RN notation it is possible to derive chord symbols. Those are defined only by root, quality, and inversion. For example, a V65 in C major in

¹ The details of RN notation are complex and out of the scope of this paper. See [29] for an introduction to the syntax.

Dataset	Composer	Content	Crotchets	Annotations
Roman Text [29] [31]	C Monteverdi	48 Madrigals	15,040	5,828
	JS Bach	24 Preludes	3,168	2,107
	FJ Haydn	24 String Quartets Movements	9,113	4,815
	Various	156 Romantic Songs	22,254	11,851
	Various	4 Additional Compositions	2,649	1,165
BPS-FH [17]	Lv Beethoven	32 Sonata Movements	23,554	8,615
TAVERN [32]	WA Mozart	10 theme and variations	7,833	3,887
	Lv Beethoven	17 theme and variations	12,840	6,836
Total		315 scores	96,450	45,104

Table 1: The datasets included in our training / validation data. TAVERN has two alternative annotations: We have only included the anonymous annotator A.

RN notation would be written in chord symbols, following the same encoding as *mir_eval* [34, 35], as *G:7/3*. All information about local key and tonicisation is lost.

Datasets. In recent years, several datasets of RN annotations have been published. Up to our knowledge, they have all been collected and converted to the “rntxt” data format [29] inside the GitHub repository in [36]. We report the size and content of the corpora we used in Table 1.² We provide the dataset parsing code, including fixes on known issues, in [28].

3. NADE FOR HARMONIC ANALYSIS

Given a musical context, that is, the portion of score between t_0 and t_1 , let us focus on the prediction of a single chord at time $t \in [t_0, t_1]$. As we have seen, a chord can be separated in several labels. This means that we can represent the output class of the chord as a variable in a multi-dimensional space. If those dimensions were all independent, one could project the distribution on each axis and independently estimate each projection of the distribution. This is the case, for example, of a rectangular uniform distribution in a 2D space, which can be written as a product of two independent uniform distributions: $p(x, y) = p_x(x)p_y(y)$. But if the distribution is more complex this is no longer true. What one can always do without loss of generality is to determine an ordering of the dimensions and estimate their value sequentially, *conditioning each dimension given the result of all the preceding ones*. This approach is at the heart of the Neural Autoregressive Distribution Estimator, or NADE [25–27].

Introduction to the NADE. The NADE is composed of two parts: a visible layer –which is made of as many neurons as there are dimensions in the distribution that we want to encode– and a hidden layer. At each step, the content of the hidden layer is used to determine the value of the next neuron of the visible layer. The output sampled from the newly-updated neuron is then reinjected into the hidden layer to inform the decision on the next step. The equations are the following [27]:

$$p(x_d | \mathbf{x}_{<d}) = \text{sigmoid}(\mathbf{V}_d \cdot \mathbf{h}_d + b_d), \quad (1)$$

$$\mathbf{h}_d = \text{sigmoid}(\mathbf{W}_{<d} \cdot \mathbf{x}_{<d} + \mathbf{c}), \quad (2)$$

where x_d is the output at dimension d , $\mathbf{x}_{<d}$ is the vector of all the outputs before d , \mathbf{V} and \mathbf{W} are respectively the

² Due to the restrictive licence under which it is released, we decided not to include the ABC dataset [37] into our training data.

tensor of hidden-to-visible and visible-to-hidden weights, \mathbf{b} is the vector of biases in the visible layer and \mathbf{c} in the hidden layer. Eqs. 1 and 2 are to be applied iteratively for all neurons in the visible layer.

Application of NADE to chords. NADE has been applied to music generation [10, 38], with the visible layer representing a frame of the piano roll. In the case of harmonic analysis, the visible layer represents instead a chord annotation. We separate the annotation along six dimensions (see Sec. 2) and organise them in the following order: key, tonicisation, degree, quality, inversion, and root.³ Every element in this list is conditioned on all the elements that appear before it.

This situation is slightly different from the one in the original NADE formulation since the output is no longer a collection of binary units, but of categorical units with a variable number of classes. The same mechanism of NADE still works, but we have to make a few modifications to it: First, a softmax layer is applied instead of a sigmoid to Eq. 1. Then, to adapt to this change in the output size, the weight tensors \mathbf{V}_d , which was understood to be unidimensional and of size n_h in the original work, is instead two-dimensional and of size (n_d, n_h) . Similarly, the shape of $\mathbf{W}_{<d}$ is $(n_h, \sum_{i<d} n_i)$ instead of $(n_h, d-1)$.

It is worth emphasising that, in this case, the NADE is used to autoregressively model the distribution of the output *on* the different dimensions of the chord and *at a specific instant of time* t .

The final frog model. So far, we only explained how to introduce correlation in the outputs but not how to derive those outputs from the inputs. Inspired by [38], we do so with the help of the biases, that we define as

$$\mathbf{b} = \text{sigmoid}(\boldsymbol{\theta}_v \cdot \mathbf{f}(\mathbf{x}) + \boldsymbol{\beta}_v), \quad (3)$$

$$\mathbf{c} = \text{sigmoid}(\boldsymbol{\theta}_h \cdot \mathbf{f}(\mathbf{x}) + \boldsymbol{\beta}_h).$$

Here $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are the weights and biases of a dense layer connecting an arbitrary function of the inputs \mathbf{f} with the NADE biases.

The function \mathbf{f} that we choose is a CRNN since it has already been proven to work well in this domain

³ The ordering of the output labels has been suggested by music theory, and namely by the fact that the degree of the Roman numeral directly depends on the key. For example, the same chord of G major could have degree *I* in the key of G or *IV* in D. Keys are a much broader structure than single chords and they change more slowly, therefore we decided to prioritise them in order to avoid an excessive amount of modulations in the output. Similarly, the degree depends on the tonicisation, since this latter expresses the temporary key from which the music borrows.

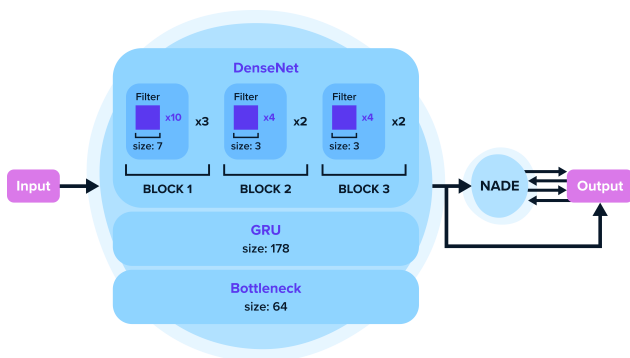


Figure 3: Schematic representation of the model structure.

[16, 18]. In particular, we follow Micchi et al. [18] and use DenseNet [39] for the convolutional part and a bi-directional GRU [40] for the recurrent part, which takes care of modelling the autoregressive part of the calculations in the time domain. A fully connected layer is introduced as a bottleneck in between the GRU and the NADE. We represent the final model, **frog**, in Fig. 3.

The hyper-parameters of **frog** have been selected with the help of hyperopt [41]. In particular, the DenseNet is made of three blocks: The first block has three convolutional layers, each made of 10 filters of size 7; the other two blocks are identical, each with 2 layers made of 4 filters of size 3. The GRU has 178 hidden neurons and is trained with a dropout of 0.2, while the bottleneck layer is made of 64 neurons. Finally, the hidden layer of the NADE is made of 350 neurons. The training is done with an ADAM optimiser with a learning rate of 0.003.

4. EXPERIMENTAL SETUP

We train our models on the task of functional harmonic analysis on symbolic scores. The input is a symbolic file (such as MusicXML, MIDI and ****kern**) and the output is an aligned harmonic analysis. We tested **frog** against two state-of-the-art models: the original CRNN architecture [18] that we used as a basis for our model and the improved Harmony Transformer model (HT*) [17, 19]. Our proposed model, **frog**, has in total 389k trainable weights, while the HT* has 750k and the original CRNN architecture only 83k. The size difference between **frog** and CRNN is partially due to **frog** including NADE (93k weights), as well as the increase of the GRU’s size to 251k weights for **frog**. A CRNN model with the same GRU hyper-parameters as **frog**, only marginally improved the outcome compared to the original CRNN model, so the former was disregarded.

All the trainings use early stopping and typically require less than 20 epochs. The entire training of **frog** lasts for a little more than 2 hours on a recent laptop (no GPU needed). The loss function is the sum of all the categorical cross entropies applied separately to each output. Each individual collection in the dataset is split 80/20 between training and validation data.

4.1 Data encoding

Pitch. For CRNN and **frog**, we have implemented two different representations of the input data: “pitch class+bass” and “pitch spelling+bass”. Pitch class+bass contains 24 elements, 12 indicating all the active pitch classes (multi-hot encoded) and 12 indicating the lowest active pitch class—the bass (one-hot encoded). If pitch class+bass is used, the output labels root and key are also encoded using only pitch classes, therefore having respectively size 12 and 24 (the keys can be major or minor).

Pitch spelling+bass, instead, contains 35 elements, that is, the seven notes times five alterations (double flats, flats, diatonic, sharps, double sharps). When pitch spelling+bass is used, the output label root has shape 35 and keys 36—this is obtained keeping the 18 keys between C \flat and A \sharp in the circle of fifths⁴ in two modes, major and minor.

Meter. We tested whether or not the addition of metrical information has a positive impact on the outcome. In models that are trained with metrical information (tagged with “w/ meter” label in Section 5), the input includes two additional vectors. The first one-dimensional vector is 1 whenever a new measure begins and 0 otherwise, the second one-dimensional vector is 1 at the onset of a new beat and 0 otherwise.

Time. The input data is quantised in time frames of the length of a demisemiquaver (1/32nd note). Due to the presence of pooling layers in the convolutional part, the output resolution is reduced and corresponds to the quaver (1/8th note).

4.2 Comparison with HT* inputs and outputs

HT* has a slightly different approach. In the original paper, the authors present two separate HT* models. In both cases, the input is encoded in MIDI numbers following a piano roll representation and additionally contains information of the tonal centroids [42].

The first model is trained for functional harmonic analysis and has two outputs: the key (24 categories = 12 pitch classes \times 2 modes) and the RN annotations (5,040 categories = 9 tonicisations \times 14 degrees \times 10 qualities \times 4 inversions). We use these RN predictions to derive the root of the chord and therefore its chord symbol representation.

The other model is trained only for chord symbol recognition and has a single output with 25 possible categories: major and minor triads (possibly with extensions) for all the 12 pitch classes and a last category for all remaining chords. We decided not to include the latter model in our experiments because the output vocabulary is too small to be fairly compared with the other models. Such a variant would be comparable to the models we train only in case it contained the same roots, qualities, and inversions as the others, for a total of 480 output classes. Moreover, such chord symbol-oriented HT* can not produce predictions for functional harmonic analysis because of the absence of key, tonicisation, and degree.

⁴ We do not keep all keys from F $\flat\flat$ to B $\sharp\sharp$ because most of those keys are never used in practice and also because they would require triple flats and sharps to encode all the diatonic pitches.

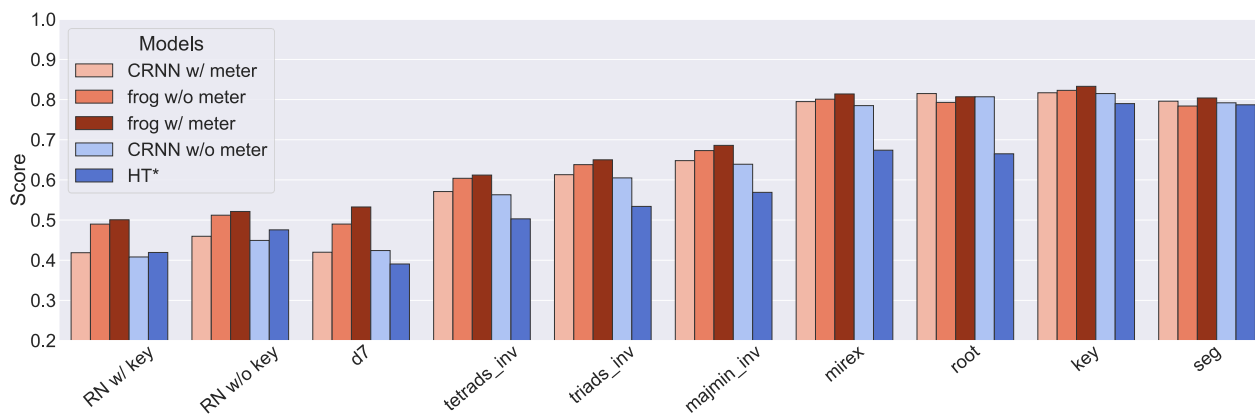


Figure 4: The score of all tested models on selected metrics. The models in shades of red (the first three) are our contributions, the blue ones (last two) are from the state-of-the-art. From left to right: the first three metrics report the ratio of frames with correct predictions to the total number of frames for the written tasks. The remaining seven report the results derived from the specific *mir_eval* tasks.

5. RESULTS

Evaluating the quality of a functional harmonic analysis is an extremely challenging task. First, there could be different analysis of the same music that are equally acceptable [18, 32] — this is a complex issue that might require a complete rethinking of our training strategies and we do not address it in this paper. Second, not all errors are equally important: one could argue that correctly identifying the inversion is less important than the root or the quality of the chord. To address this second issue, we report the scores on several metrics and let the readers decide which one is the most important for their task.

5.1 Analysing the metrics

Remarkably, **frog** shows better results when compared to the previous state-of-the-art models (CRNN w/o meter and HT*) in almost every metric considered (see Fig. 4). Notice that, to provide a better comparison with the HT* model, we report the results of the *pitch class + bass* input data representation (see Sec. 4).

Accuracy on Roman numerals. The most complete metric that we show is the accuracy on RNs. (see Fig. 4, first two metrics from the left). We present two versions: in the first (“RN w/o key”), the prediction is considered correct if and only if tonicisation, degree, quality, and inversion are all correct — this corresponds to the direct RN output of the HT* model (see Sec. 4). For this task, **frog** reports a 52.1% accuracy against the 47.6% that we obtained for HT* and 44.9% for CRNN (w/o meter, understood from now on). The new XL-CRNN achieves 47.1%.⁵

The second case (“RN w/ key”) requires also a correct prediction of the key. Here, **frog** still gives a correct prediction in 50.1% of cases against 41.9% that we obtained for HT* and 40.8% for CRNN (43.1% on XL-CRNN). The

⁵ The HT* results we report show a significantly higher accuracy than the 41.7% reported in [19]. We assume that this is due to the larger size of the dataset we train all three models on.

absolute margin of improvement of **frog** on the best competing state-of-the-art algorithms goes from 4.5% on RN w/o key to 8.2% on the more complex task of RN w/ key.

The case of the diminished sevenths. Diminished seventh are a special chord in music theory because they divide the octave in 4 equal intervals. Therefore, these highly symmetrical chords are often used during modulations. This makes them very easy preys to problems of misclassification due to the lack of coherence. In addition, they are sporadic chords, making up 4.3% of our dataset, which makes correct predictions both difficult and important. The accuracy with **frog** makes a big leap from 39.1% of the HT* model and 42.4% of CRNN to 53.3%, showing a better than average result on these chords (See Fig. 4, metric “d7”).

The scores on *mir_eval* metrics. We then report a selection of the metrics included in the package *mir_eval* [35] (see Fig. 4, last seven metrics to the right).

The first conclusion we can draw from these results is that the HT*, which chooses its output among a large vocabulary of more than 5000 output classes, has the lowest accuracy of all systems. The more powerful variant of the ACR-oriented version of HT* that we mentioned in Sec. 4 would however probably obtain higher scores than this general-purpose HT* on these metrics.

The second conclusion is that all models perform almost equally on segmentation. The segmentation is reported as the minimum of the score on over-segmentation and under-segmentation and for all models the minimum score is given by the over-segmentation. This could be due either to an intrinsic limitation that is common to all architectures and that needs yet to be discovered; it could be also due to the fact that human annotators might prefer a more synthetic analysis: for example, some notes could be interpreted as passing tones by humans and considered instead as structural part of the chord by the algorithm.

The root coherence. As we saw in Sec. 2, it is possible to derive the root of a chord from its key, tonicisation,

and degree. The CRNN and **frog** models predict an additional redundant output, the root, with the assumption that it helps the systems learn faster. Comparing the root derived from the RN with the one directly estimated by the model we can obtain a measure of the internal coherence of the output labels. CRNN has a root coherence of 78.9%, compared to **frog** which has a root coherence of 99.0%.

Adding metrical information. We notice that the introduction of metrical information (cf. Section 4) has a positive but quite small impact on the results in all metrics.

5.2 Analysing the confusion matrix

In Fig. 5 we report the confusion matrix for the key obtained with **frog** when trained with pitch spelling. The keys are arranged according to the circle of fifths (F-C-G-D...) with the major keys preceding the minor keys, i.e., the top-left quadrant shows the major/major correlation while the bottom-right the minor/minor correlation. The values reported are the occurrences of each pair ground-truth/prediction and are presented in a logarithmic scale to enhance the different scales in prediction errors.

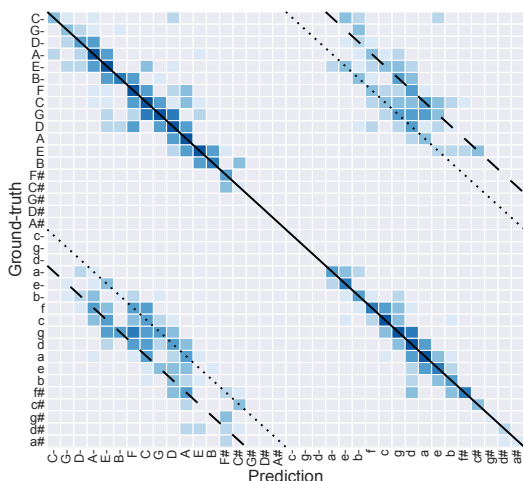


Figure 5: Confusion matrix for **frog** on label *key*.

The *mir_eval* key metric that we reported in Fig. 4 assigns 1 point to all keys that are correctly predicted, 0.5 points to dominant/sub-dominant predictions (**G/F** instead of **C**), 0.3 to relative (**a** instead of **C** or vice versa), and 0.2 to parallel (**c** instead of **C** or vice versa). Those cases are the ones reported on the five diagonals super-imposed to the plot: The main diagonal, in solid line, shows the correctly predicted keys. Dominant predictions are immediately to its right and sub-dominant to its left. Dashed lines show the relative predictions, while dotted lines show the parallel predictions. Some rows and columns of the confusion matrix are empty: These are the keys that are supported by the model but never used in the test dataset.

5.3 Using a key oracle

In a separate but related experiment, we have allowed **frog** to access a key oracle. We did that by reading the key from the test data and setting it as the first output of the visible

layer of the NADE. Then, we sampled the remaining labels autoregressively in the given order, as usual.

We measured the impact of this key oracle on the results. Without a dedicated retraining, a multi-output model with no coherence between the different labels, such as the HT* or the CRNN, would report unvaried accuracies for all elementary labels except key. This entails that the accuracy for the RN w/ key prediction be equivalent to the one for RN w/o key. However, this is not what happens with **frog**: The degree accuracy goes from 72.6% to 80.3% and the tonicisation from 91.4% to 94.0%.⁶ As a result, the accuracy on RN w/ key jumps to 60.3%, much higher than the 52.1% we would expect in absence of coherence.

6. CONCLUSIONS AND PERSPECTIVES

We report an advancement in the field of automatic chord recognition and especially functional harmonic analysis for symbolic music. The improvements are mostly due to the use of a modified version of the NADE algorithm, which allows us to separate the complex and large vocabulary of all possible output classes into a set of elementary labels (such as key, degree, and quality of the chords) while retaining strong coherence between them. This effectively reduces the size of the output classes by several orders of magnitude and at the same time offers better results, as we showed in Sec. 5.

A consequence of the reduction in complexity of the output labels is the increased flexibility that this model gives to the users, as changes to the chord labels do not dramatically alter the size of the model nor the complexity of the task. For example, one could easily introduce a larger amount of chord colourings, which makes **frog** a better candidate for analysing music such as jazz.

A lot still remains to explore on the details of this approach. We kept the six output labels that had already been presented in previous articles [16–18], where they had been used in absence of the coherence-enforcing NADE layer. Now, we expect to be able to improve the quality of the output even further by tweaking these six labels. For example, one could study the key by separating tonic and mode (major / minor); and the degrees could be separated on two axis: the position on the scale and the alteration.

Concerning the input representation, there are at least three strains of research that caught our interest: relative music representation [43, 44], use of Tonnetz for better convolutions in pitch spaces [45], and better representations for the metrical strength [46].

Another aspect to test is the introduction of Orderless-NADE [26]. The OrderlessNADE effectively trains one separate model for all the possible orderings and then averages the results obtained. This approach could improve the quality of the model both directly (because it demonstrates to be intrinsically superior to our ordered model) and indirectly (because it allows us to find a better ordering than the one we proposed).

⁶ The remaining three labels —inversion, quality, and root— are not impacted, which is consistent with the fact that they are independent from the key from a music theory point of view.

7. ACKNOWLEDGEMENTS

We thank Raluca Semencescu for creating the main illustrations in the paper and Jordan B. L. Smith for his extensive paper review before submission.

8. REFERENCES

- [1] A. Schoenberg, *Harmonielehre*. Universal Edition, 1922.
- [2] N. Cook *et al.*, *Music, imagination, and culture*. Oxford University Press, 1990.
- [3] D. Tymoczko, *A geometry of music: Harmony and counterpoint in the extended common practice*. Oxford University Press, 2010.
- [4] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinulescu, and C. Cai, “Human-AI Co-creation in Songwriting,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference, ISMIR 2020*, 2020.
- [5] J. Pauwels, K. O’Hanlon, E. Gómez, and M. B. Sandler, “20 Years of Automatic Chord Recognition From Audio,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, nov 2019, pp. 54–63. [Online]. Available: <https://qmul.ac.uk/xmlui/handle/123456789/62088>
- [6] T. Fujishima, “Real-time chord recognition of musical sound: A system using common lisp music,” pp. 464–467, 1999.
- [7] P. Kröger, A. Passos, M. Sampaio, and G. De Cidra, “Rameau: A system for automatic harmonic analysis,” in *In Proceedings of ICMC*, 2008.
- [8] K. Masada and R. C. Bunescu, “Chord recognition in symbolic music: A segmental crf model, segment-level features, and comparative evaluations on classical and popular music,” *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 1–13, 2019.
- [9] G. Hadjeres, F. Pachet, and F. Nielsen, “Deepbach: a steerable model for bach chorales generation,” in *International Conference on Machine Learning*, 2017, pp. 1362–1371.
- [10] C. Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by convolution,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*, 2017, pp. 211–218. [Online]. Available: <https://coconets.github.io>
- [11] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinulescu, and D. Eck, “Music transformer,” *arXiv preprint arXiv:1809.04281*, 2018.
- [12] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, “A hierarchical latent vector model for learning long-term structure in music,” in *International Conference on Machine Learning*, 2018, pp. 4364–4373.
- [13] M. Dinulescu, J. Engel, and A. Roberts, “Midime: Personalizing a musicvae model with user data,” in *Workshop on Machine Learning for Creativity and Design, NeurIPS*, 2019.
- [14] M. Rohrmeier, “Towards a generative syntax of tonal harmony,” *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, 2011.
- [15] D. Harasim, M. Rohrmeier, and T. J. O’Donnell, “A generalized parsing framework for generative models of harmonic syntax.” in *ISMIR*, 2018, pp. 152–159.
- [16] T.-P. Chen and L. Su, “Functional harmony recognition of symbolic music data with multi-task recurrent neural networks,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 90–97. [Online]. Available: <https://github.com/>
- [17] T. P. Chen and L. Su, “Harmony transformer: Incorporating chord segmentation into harmony recognition,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, 2019, pp. 259–267. [Online]. Available: https://aclweb.org/aclwiki/POS_
- [18] G. Micchi, M. Gotham, and M. Giraud, “Not All Roads Lead to Rome: Pitch Representation and Model Architecture for Automatic Harmonic Analysis,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 42–54, may 2020. [Online]. Available: <http://transactions.ismir.net/articles/10.5334/tismir.45/>
- [19] T.-P. Chen and L. Su, “Attend to Chords: Improving Harmonic Analysis of Symbolic Music Using Transformer-Based Models,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 1–13, feb 2021. [Online]. Available: <http://transactions.ismir.net/articles/10.5334/tismir.65/>
- [20] Y. Ju, N. Condit-Schultz, C. Arthur, and I. Fujinaga, “Non-chord tone identification using deep neural networks,” in *Proceedings of the 4th International Workshop on Digital Libraries for Musicology*, 2017, pp. 13–16.
- [21] D. Temperley, “What’s key for key? the krumhansl-schmuckler key-finding algorithm reconsidered,” *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.
- [22] S. T. Madsen, G. Widmer, and J. Kepler, “Key-finding with interval profiles,” in *In Proceedings of ICMC*, 2007.

- [23] N. Nápoles López, C. Arthur, and I. Fujinaga, “Key-finding based on a hidden markov model and key profiles,” in *6th International Conference on Digital Libraries for Musicology*, 2019, pp. 33–37.
- [24] N. Nápoles López, L. Feisthauer, F. Levé, and I. Fujinaga, “On local keys, modulations, and tonicizations,” in *Digital Libraries for Musicology (DLfM 2020)*, 2020.
- [25] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” *Journal of Machine Learning Research*, vol. 15, pp. 38–39, 2011.
- [26] B. Uria, I. Murray, and H. Larochelle, “A deep and tractable density estimator,” *31st International Conference on Machine Learning, ICML 2014*, vol. 1, pp. 719–727, 10 2014. [Online]. Available: <http://arxiv.org/abs/1310.1757>
- [27] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, “Neural autoregressive distribution estimation,” *Journal of Machine Learning Research*, vol. 17, pp. 1–37, 2016.
- [28] G. Micchi, accessed 2021-07-29. [Online]. Available: <https://gitlab.com/algomus.fr/functional-harmony>
- [29] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, “The romantext format: A flexible and standard method for representing roman numeral analyses,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, 2019, pp. 123–129.
- [30] S. Kostka and D. Payne, *Tonal harmony*. McGraw-Hill Higher Education, 2013.
- [31] N. Nápoles López, “Automatic harmonic analysis of classical string quartets from symbolic score,” Master’s thesis, Universitat Pompeu Fabra, 2017.
- [32] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, “Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, 2015, pp. 728–734.
- [33] T.-P. Chen, S. Fukayama, M. Goto, and L. Su, “Chord jazzification: Learning jazz interpretations of chord symbols,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference, ISMIR 2020*, 2020.
- [34] C. Harte, M. B. Sandler, S. A. Abdallah, and E. Gómez, “Symbolic representation of musical chords: A proposed syntax for text annotations.” in *ISMIR*, vol. 5, 2005, pp. 66–71.
- [35] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, “mir_eval: A Transparent Implementation of Common MIR Metrics,” in *Proc. of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 367–372.
- [36] M. Gotham, accessed 2021-07-29. [Online]. Available: <https://github.com/MarkGotham/When-in-Rome>
- [37] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, “The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets,” *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fdigh.2018.00016>
- [38] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription,” in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 2, 6 2012, pp. 1159–1166. [Online]. Available: <http://arxiv.org/abs/1206.6392>
- [39] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, 2017, pp. 2261–2269. [Online]. Available: <https://github.com/liuzhuang13/DenseNet>.
- [40] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: <https://www.aclweb.org/anthology/D14-1179>
- [41] J. Bergstra, D. Yamins, and D. D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in *30th International Conference on Machine Learning, ICML 2013*, vol. 28, 2013, pp. 115–123.
- [42] C. Harte, M. Sandler, and M. Gasser, “Detecting harmonic change in musical audio,” in *Proceedings of the ACM International Multimedia Conference and Exhibition*, 2006, pp. 21–26.
- [43] S. Lattner, M. Grachten, and G. Widmer, “Learning transposition-invariant interval features from symbolic music and audio,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 661–667. [Online]. Available: <https://github.com/SonyCSLParis/cgae-invar>
- [44] —, “A predictive model for music based on learned interval representations,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 26–33.

- [45] R. Abecidan, M. Giraud, and G. Micchi, “Towards custom dilated convolutions on pitch spaces,” in *International Society for Music Information Retrieval Conference (ISMIR 2020), Late-Breaking Demo Session*, 2020.
- [46] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdallah, M. Selvi, E. Newton-Rex, and K. Webster, “StructureNet: Inducing structure in generated melodies.” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018, pp. 725–731.

MODELING BEAT UNCERTAINTY AS A 2D DISTRIBUTION OF PERIOD AND PHASE: A MIR TASK PROPOSAL

Martin A. Miguel^{1,2} and Diego Fernández Slezak^{1,2}

¹Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales.

Departamento de Computación. Buenos Aires, Argentina.

²CONICET-Universidad de Buenos Aires. Instituto de Investigación en Ciencias de la Computación (ICC). Buenos Aires, Argentina.

mmiguel@dc.uba.ar

ABSTRACT

This work proposes modeling the beat percept as a 2d probability distribution and its inference from musical stimulus as a new MIR task. We present a methodology for collecting a 2d beat distribution of period and phase from free beat-tapping data from multiple participants. The methodology allows capturing beat-tapping variability both within (e.g.: mid-track beat change) and between annotators (e.g.: participants tap at different phases). The data analysis methodology was tested with simulated beat tracks assessing robustness to tapping variability, mid-tapping beat change and disagreement between annotators. It was also tested on experimental tapping data where the entropy of the estimated beat distributions correlated with tapping difficulty reported by the participants. For the MIR task, we propose using optimal transport as an evaluation criterion for models that estimate the beat distribution from musical stimuli. This criterion provides better scores to beat estimations closer in phase or period to distributions obtained from data. Finally, we present baseline models for the task of estimating the beat distribution. The methodology is presented with aims to enhance the exploration of ambiguity in the beat percept. For example, it exposes if beat uncertainty is related to a pulse that is hard to produce or conflicting interpretations of the beat.

1. INTRODUCTION

The beat is a cognitive percept fundamental for the human musical experience. It is often conceived as an isochronous pattern expressed by tapping with a hand or foot. Mentally, the location and duration of musical events are defined with respect to it. It is also subjective as different listeners may select different tapping speeds or locations when producing the beat [1–3]. Also, how easily the beat percept is perceived – its pulse clarity – may vary according to the musical stimulus [4].

The uncertainty on whether there is a perceived beat and which is finally perceived is also part of the music experience. For example, pulse clarity has been related with higher valence [5] and arousal [6]. Pulse clarity has also been shown to correlate with the degree and variability of movement [7, 8]. Overall, uncertainty on the musical structure is considered relevant for the analysis of emotion in music. Theories on the mechanisms through which music produces affective responses point towards unfulfilled expectations. We listen to music and predict how it will develop. If such predictions are defied, an affective response emerges [9–11]. Predictions on the musical structure include how its events organize in time, which is arranged based on the periodicity of the beat. On top of the beat, other hierarchical structures are conceived, such as the downbeat and the meter. These, in turn, organize the length and location of repetitions and sections of a song [12, 13]. With this in mind, both the frequency and the location of the beat are of relevance.

Moreover, the certainty of our predictions is important as well. It is considered that prediction error is what causes affective response [14]. If no structure is predicted with high probability, no outcome is considered a prediction error [15]. In order to build models to analyze expectations with respect to the timing of musical events, we need to model which beat percepts emerge in listeners, if there are conflicting interpretations, and what is the certainty about them.

Ambiguity in the interpretation of the beat has been analyzed experimentally only by looking into how listeners select different tapping speeds. [16] analyzed subjective tempo on various musical stimuli, concluding that, on top of the music’s base tempo, its structure and accents can also influence the selected tapping tempo. [1] observed that listeners may have different strategies to select a tapping speed. Some can comfortably tap at the fastest consistent pulse of the music, while others had a tendency towards a slower subdivided pulse.

From a Music Information Retrieval perspective, the tempo estimation task aims to estimate the most likely tapping speeds. For example, in the MIREX tempo estimation challenge, algorithms must provide the two most salient tempos together with their saliency [17] (for a review see [18–20]). Another related MIR task is beat track-



© M. Miguel and D. Fernández Slezak. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: M. Miguel and D. Fernández Slezak, “Modeling beat uncertainty as a 2D distribution of period and phase: a MIR task proposal”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

ing. The task aims to estimate, from a musical surface, the time points where a listener would tap. Several beat tracking models have been introduced, many within the beat tracking task in MIREX (for a review, see [21, 22]). These models find the best beat track, but rarely produce information on other possible interpretations.

Considering these limitations where only tempo is analyzed or only one possible beat is estimated, this work proposes modeling the beat as a 2D probability distribution of tempo and phase. The probability modeling allows capturing both tempo and location of the beat, whether more than one interpretation is likely, and overall beat certainty. Furthermore, we propose to construct the inference of the beat distribution as a MIR task. This constitutes an extension of the tempo estimation task, as it yields more information about the possible interpretations of the beat and their saliences. The extended information is relevant to understand which metrical interpretations are likely for a listener and how certain she might be, which is deemed important for the analysis of affect in music. We present the pipeline to construct this task, including how to obtain these distributions from the listener’s tapping data and a proposal evaluation metric for models estimating the distributions from musical stimuli.

In the next sections, we present the formalisms of the proposed task. First, we define the 2d distribution and present the algorithm for obtaining it from tapping data. Next, we assert that the definition and the algorithm capture relevant beat situations from simulated tapping data. The algorithm is also applied on experimental tapping data where participants were required to tap to a self-selected beat on rhythms of varying complexity. We describe the experiment and show how different beat uncertainty scenarios are present from the data and that a more spread distribution correlates with reported tapping difficulty. Finally, we present an evaluation metric for MIR models producing these distributions from musical stimuli. We also provide 3 baseline models and their scores on the experimental dataset.

2. BEAT AS A 2D DISTRIBUTION

We propose modeling the beat percept as a 2d probability distribution of period (or tempo) δ and phase ρ . Such distribution allows illustrating the beat variability commonly expressed in beat-tapping data. We consider δ to be in some time unit in a bounded range reasonable for beat perception. A proposed range is 250 ms (240 bpm) to 1800 ms (33 bpm) [1]. We consider $\rho \in [0, 1]$ as a relative location within the period. The methodology estimates a discretized probability distribution $p(\delta, \rho | \text{stimulus})$ from the beat-tapping data of multiple participants while listening to the stimulus. For example, in the simulated data (section 2.1.1), we discretize the support every 25 ms for the period δ and 0.05 for the phase ρ . Examples of the 2d distribution for different rhythmic stimuli are presented in Figure 1.

Let us consider the tapping data as a set of tap times series $\{t_i^j\}$, with j indicating the participant ($j = 1, \dots, J$), and i indicating the tap index ($i = 1, \dots, N_j$), where J is

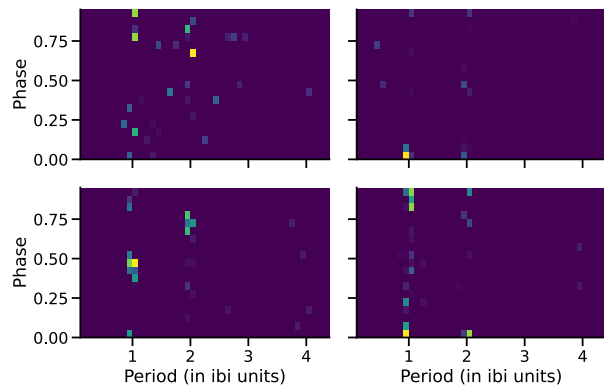


Figure 1: Example beat distributions from experimental tapping data for various rhythmic stimuli. The distributions depict different beat perception situations, such as high and low variability (top left and right, respectively), one period with multiple phases (bottom left) or one main phase but multiple periods (bottom right).

the number of participants and N_j the number of taps by participant j . Our aim is to calculate the evidence for each beat bin present in the data. The process is illustrated in Figure 2. First, taps are split into segments $\{s_i^{j,k}\}_{i=1}^{S_i^{j,k}}$ of similar inter-tap intervals (Figure 2a). From each segment, a tactus, with inter-beat interval (IBI) δ and phase relative to the beginning of the stimuli ρ , is estimated with a linear regression. The period and phase are obtained using only the tapping data, without requiring a true beat level or a reference beat sequence. Next, the segment is assigned to a bin in the beat distribution (Figure 2b). To quantify the evidence provided by a segment, the stimulus is segmented into time frames which are assigned to overlapping segments (Figure 2a). The final distribution histogram is generated by adding the frame counts assigned to each beat bin by each segment and then normalizing (Figure 2c).

Taps are segmented by iterating them in order. The first two tap times constitute the first segment. Then, the segment is extended by inspecting if the distance between the last tap in the segment and the following tap time (Δ_*) is similar to the average inter-tap interval of the segment ($\bar{\Delta}_{s_i^{j,k}}$) within a threshold Δ_{th} . The criterion is expressed in equation 1. If the equation holds, the segment is extended with the new tap time. Otherwise, a new segment is initiated with the next two tap times. Δ_{th} is defined as 0.175 based on the Continuity metric used as evaluation in the beat tracking task [23].

$$\frac{|\Delta_* - \bar{\Delta}_{s_i^{j,k}}|}{\bar{\Delta}_{s_i^{j,k}}} \leq \Delta_{th} \quad (1)$$

With each tap segment $\{s_i^{j,k}\}$ defined, we estimate which beat is being produced during the segment. To do so, we perform a linear regression with parameters α and β minimizing the mean squared difference between $s_x^{j,k}$ and $(\alpha + \beta \times (x - 1))$ (with $x = 1, \dots, S_i^{j,k}$). The beat selected for the segment corresponds to the beat bin of the discretized distribution containing $(\delta = \beta, \rho =$

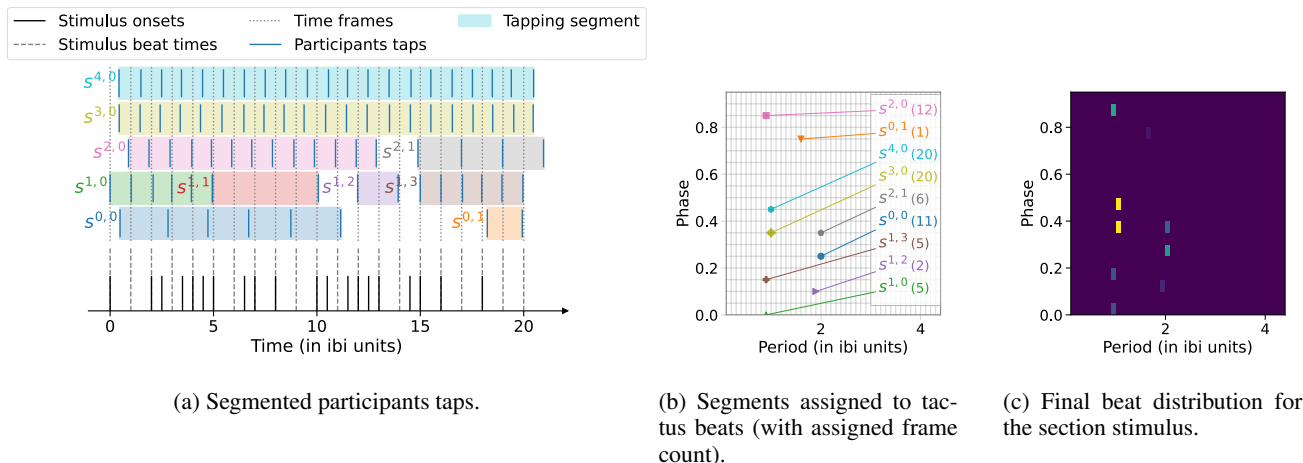


Figure 2: Illustration of the process to obtain a beat distribution from tapping data. (a) presents tapping data from 5 participants (one per row) on a section of a stimulus. Participants taps are clustered in segments $\{s_i^{j,k}\}$ of similar inter-tap intervals. From each segment, a tactus (ρ, δ) is obtained with a linear regression. The stimulus is divided into time frames, which are assigned to the overlapping tapping segment (if any). (b) the tactus of each segment is mapped onto the distribution support. Segments contribute to their tactus bins according to the number of frames assigned to the segment (in parenthesis). (c) shows the distribution generated from the process.

$(\alpha \bmod \beta) / \beta$. The estimated values correspond to the inter-tap interval and location with respect to the beginning of the rhythm. The segmentation process allows having more tapping information to inform the parameter estimation in spite of tapping variability.

Finally, the duration of the stimulus is split into time-frames. Each time-frame is assigned the beat bin of the segment that overlaps with it. If no segment overlaps with the frame, the frame is ignored. The beat distribution $p(\delta, \rho | \text{stimulus})$ is defined as the normalized histogram of frame counts for each beat. Figure 1 presents examples of this procedure applied to data from a tapping experiment (the experiment and the adaptations to the procedure are described in section 2.1.2).

2.1 Evaluation

2.1.1 Simulated data

We evaluated the methodology by producing simulated tap time series from a selected beat and then calculating the beat distribution with the procedure presented in the previous section. First, we asserted that distributions obtained were robust to tapping variability. Second, considering a listener may change the beat they are tapping, we examined whether the probability of two different beats is proportional to the time each beat is produced. Third, considering data from various listeners, we assessed that the probability of two beats is proportional to the number of listeners tapping to it. For the experiments we used beat bins with period $\delta \in [250ms, 1800ms]$ with $25ms$ increments, and phase $\rho \in [0, 1]$ with steps of 0.05.

For the first evaluation, we created simulated beat-tapping series with tapping variability and evaluated whether the beat distribution obtained had most of the probability mass in the expected beat. To produce each

simulated tapping series, a beat was defined by first randomly selecting a beat bin and then drawing a specific period δ and phase ρ values within the bin. With the selected period and phase, tap times were generated starting at $\rho \times \delta$ and then adding δ time for each successive tap until reaching 30 seconds. Tapping variability was controlled with parameter σ and was incorporated into the tap series by adding Gaussian noise $N(0, \sigma \times \delta)$ to each tap time. We tested 20 tapping variability magnitudes, with $\sigma \in [0, 0.08]$. Tapping variability has been reported to range from 2% to 4%, depending on inter-beat interval duration and musical training of the listener [24, 25]. It may be as high as 5% for very slow tempos (below 60 bpm) [26]. We selected 100 random beats and used each to produce 20 tapping series, one for each value of σ .

Figure 3 presents the average probability for the original beat bin at each σ . Since the variability in the tapping times might yield a different beat bin, we evaluated whether the probability mass was captured by neighboring phase and period bins. We see the probability of the originally selected beat bin decays with tapping variability, but up to 5% it is mostly captured by the immediate neighboring phase bin. The probability of the original bin together with the neighboring period bins is practically the same as the original bin, meaning that probability mass is rarely transferred to a different tempo.

The methodology is expected to work in free-tapping situations where the listener may stop and even change the beat she is tapping during the musical excerpt. The segmentation and framing procedure is used to capture these changes. We evaluate whether this behavior is captured by simulating tapping series where the beat is changed mid-tapping. Each series is generated by selecting two random period and phase values as before, selecting the proportion of time the tapping series will produce each period

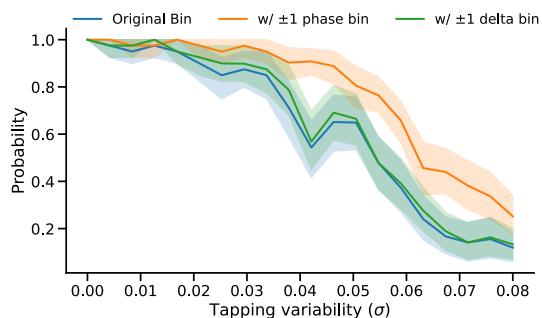


Figure 3: Evaluation on whether a beat distribution obtained from simulated tapping data assigns the probability to the original beat bin, with respect to tapping variability (σ). Most probability mass is given to the original bin, although it decays with variability. Up to 5% variability, most of the probability is assigned to either the original bin or its neighboring phase bins. Little probability is transferred to nearby period. Values are presented as mean probability for 100 simulations per sigma value, along with the 90% confidence interval.

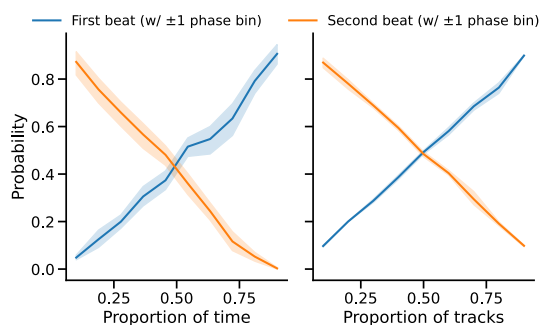


Figure 4: Probability of the first and second selected beats with respect to the proportion of time (left) or series (right) each beat is produced. Probability for both beats are shown to be linearly related to the amount of time or series expressing this beat. Values are presented as mean probability for 50 simulations per proportion value, together with 90% confidence intervals.

and phase, and then generating the tapping for the first and second beat for the defined proportion of the time. From these tapping series, the beat distribution is obtained and the probability assigned to the two selected beat bins is observed. Here, tapping series are generated with 3% tapping variability and are also 30 seconds long.

Similarly, the methodology is designed to capture the proportion in which different beats are selected by a set of listeners. In the last simulation, we produced a set of tapping series where a proportion was generated based on one beat and the rest on a second one. For each simulation, the beat distribution is obtained by counting the number of frames assigned to each beat bin from the complete set of tapping series. Then, the probability of each beat, with respect to the number of simulated participants producing that beat, is observed. Tapping series were also generated with 3% tapping variability and 30-second duration.

In Figure 4 we present the results of the simulations. In both cases, the probability of the first selected beat decays linearly with the proportion of time or series the beat is produced. Equally, as the proportion of time or series expresses the second beat, its probability increases.

2.1.2 Experimental data

Finally, the proposed methodology was used to obtain 2d beat distributions from free-tapping data. In Figure 1 we present examples of the beat distribution for four stimuli. Data was collected on an on-site experiment where participants were asked to tap on a sensing surface to a self-selected beat while listening to rhythmic stimuli of varying rhythmic complexity [27]. The experiment was designed to capture subjective beat. Participants were instructed to tap to any self-selected beat and were allowed to stop or change their tapping throughout the stimulus. After each trial, participants rated how difficult the tapping task was with values between 1 (easy) and 5 (hard). Stimuli consisted of repeating rhythms produced using identical click sounds. A total of 33 rhythmic passages were presented to each participant. To use rhythms with validated complexity, 11 of the rhythms were taken from [3] and 7 from [28]. 5 were isochronous beats at 150, 200, 250, 500, 800 ms inter-beat intervals. 10 new patterns were created from four beat patterns (in contrast with 8 from [3] and [28]) to have participants familiarize with the task. 7 of these were always presented at the beginning of the experiment. All other stimuli presentations were randomized. With the exception of the isochronous stimuli, pattern-based stimuli were presented varying the notated inter-beat interval between 450 and 550 ms. IBIs were pseudo-randomized avoiding using the same one in two consecutive trials. Each stimulus consisted of repeating the rhythmic pattern to last a minimum of 24 seconds (and up to 31 seconds). From 35 total participants, 30 remained after filtering participants that were deemed to not understand the concept of beat. They were selected as participants who replicated the stimulus instead of defining a beat in more than three trials. 6 participants were female, and 24 are male. The overall average age was 28.27 (sd = 7.94) and overall mean musical training was 5.43 years (sd = 4.62).

From the data collected, we gathered a training subset to be used in the exploration performed in this work. 15 participants were selected to uniformly represent the range of training years. The rhythmic stimuli subset contains all the isochronous excerpts, 5 from [3], 3 from [28] and 4 from the newly proposed rhythms. Rhythms subsets were selected to uniformly represent the reported tapping difficulty range. In the experimental data, tap times are normalized to the inter-beat interval used during the experiment. This allows comparing presentations of the same stimulus at different IBIs. For the analysis in Figure 1, the beat distribution was obtained with period $\delta \in [0.1, 4.5]$ with increments of 0.01. The figure shows how inferred beats concentrate on specific areas of the distribution. For example, we can observe that the tapping period is most often the stimulus' inter-beat interval and only in some rhythms

double-period tapping is also present. The distributions also show that the tapping phase is commonly near 0 (or 1), indicating synchrony with the stimulus' beat. This is not always the case, as in the lower left subfigure, anti-phase tapping (phase nearby 0.5) is more prevalent, indicating that participants considered this beat more likely than the one originally defined in the stimulus.

We also assessed whether the spread of the probability mass was related to the tapping difficulty reported by the participants. We estimated tapping difficulty from the distribution by calculating its entropy. Reported tapping difficulty for each stimulus was obtained as the mean of the per-participant z-standardized difficulty scores. Spearman rank's correlation was calculated on the training subset, ignoring the isochronous stimuli as they were not intended to express rhythmic complexity. The correlation yielded $r = 0.88$ and $p < 0.001$.

3. EVALUATION METRIC

We propose estimating the probability of different beats being perceived as the pulse of a musical stimulus as a new MIR task. An empirical discrete beat distribution (considered as period and phase) is obtained from tapping data produced by listeners. A model for this task would be required to produce estimates of the probability of each beat from the musical stimulus. To evaluate the model, both distributions must be compared. We propose using the Earth Mover's distance (or Wasserstein distance) for this comparison [29]. This distance evaluates how much probability mass must be moved in order to convert one distribution into the other. It considers two distributions with probability mass nearby to be less distant than if their mass is further apart. In contrast to the more commonly used Kullback-Leiber divergence, the Earth Mover's distance does not require both distributions to have non-zero mass on the entire support. It also takes into account the topology of the support as it allows defining a distance between the bins. We consider the distance between beat bins (δ_i, ρ_i) and (δ_j, ρ_j) in the distribution's support as the Manhattan distance between the bins. We propose adding a multiplier M_δ to the distance between periods to penalize the difference in the period more than in the phase. Here, we use $M_\delta = 5$. We also modify the distance calculation to allow a phase value close to 1 to be also close to 0, given the circularity of the phase [2, 30]. We present the used topology in equation 2.

$$d((\delta_i, \rho_i), (\delta_j, \rho_j)) = |\delta_j - \delta_i| \times M_\delta + \min(|\rho_j - \rho_i|, 1 - |\rho_j - \rho_i|) \quad (2)$$

To test the proposed distance we generated pairs of distributions from increasingly distanced phase bins. Figure 5 presents the mean Earth Mover's distance and Kullback-Leiber divergence calculations for 20 simulations at each possible phase distance. We observe that the proposed distance increases linearly with the distance in phase between the distributions and then decreases when the distance in

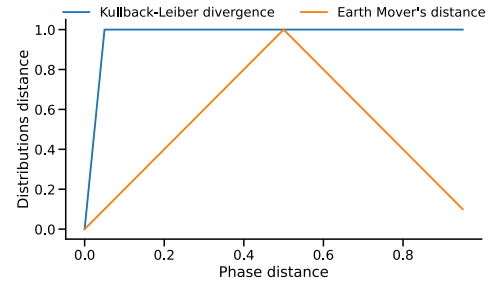


Figure 5: Distance metrics for two distributions only shifted in phase. The Earth Mover's distance is proportional to the shift in phase and responds to the circularity of phase. When the shift is greater than 0.5, the distance decreases.

phase is reduced by the corresponding circularity. Contrastingly, the Kullback-Leiber divergence fails to capture the proximity of the distributions.

3.1 Baseline models

We now present three reference models to provide an overview of the expected distance values for the task, as well as exemplify a first approach. The models are designed to take as an input a series of onset times, equivalent to the stimuli used in the experiment. The models are also expected to provide a discretized beat distribution, considering the support described in section 2.1.2. We describe the models and present their scores on the training subset.

The first reference model is the uniform distribution on any beat, i.e.: $p(\delta, \rho | \text{stimulus}) \propto 1$. Although a uniform distribution might not have the largest distance to the target distribution, we will use its distance to express the scores of the models with respect to it. In case the estimated distribution is more distant than a uniform distribution, the score will be higher than 1. The second reference model is fixed on phase 0 and only provides non-zero probability for periods $\delta_{1..4} = \{0.5, 1, 2, 4\}$. The model, named *Phase Zero*, is expressed in equation 3. The probability D_i provided to each period was calibrated by fitting a Gamma distribution to the distribution of tempos selected by the participants in the isochronous stimuli of the training subset.

$$p(\delta, \rho | \text{stimulus}) \propto \begin{cases} D_i & \text{if } \delta = \delta_i \text{ and } \rho = 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

The third reference model, named *Beat*, assigns probability proportional to a beat fitness score multiplied with a prior distribution on the period. The fitness score projects the given period and phase throughout the length of the stimulus and evaluates whether the projected beat times coincide with musical onsets. Coincidence is measured as the density of a Gaussian window centered on the expected beat time with variance $\sigma_w = 0.1 \times \delta$ [31]. To avoid favoring faster or slower tempos, the number of correctly predicted beat times is multiplied by the number of correctly predicted onset times [22]. The model is described

Model	Distance	Rel. Dist.
Beat	6.740872	0.398315
Phase Zero	7.307728	0.460842
Uniform	17.895050	1.000000

Table 1: Evaluation scores on training subset for reference and baseline models. Distance is the mean Earth Mover’s distance of each estimated to each empirical beat distribution. Relative Distance is the mean distance, relative to the distance of the uniform distribution, per stimulus. Lower values mean a closer estimation to the target distribution.

in equation 4. The prior for the period is given by the same Gamma described above. The distribution provided by this model is later filtered by turning to zero all beat bins where probability is below the 95th percentile, as only the most salient beats are expressed by a listener’s tapping.

$$p(\delta, \rho \mid \text{stimulus}) \propto \text{score}(\delta, \rho, \text{stimulus}) \times \text{prior}(\delta) \quad (4)$$

$$\text{score}(\delta, \rho, s) = \frac{[\sum_j \max_b W((\gamma_b - p_j)/(0.1\delta))]^2}{|p| \times |\gamma|}$$

with γ_b the onset times and p_j the projected beat series.

In Table 1 we present the average Earth Mover’s distance for each model’s beat distribution to the tapping data. The table also presents the mean relative distance when compared with the uniform model for each stimulus. In this evaluation, the isochronous stimuli are excluded from the evaluation since they were used to calibrate the period prior. The table shows how assigning most of the distribution to phase zero reduces the relative distance from the uniform distribution by half. The *Beat* model adds a 6% improvement. In Figure 6 we present a sample of estimated and empirical beat distributions from the training dataset for the *Beat* model. We present the two closest and two most distant estimations in the dataset.

4. DISCUSSION

We propose a new MIR task consisting of estimating the probability distribution of which beats are perceived by listeners for a musical stimulus. For this task, the beat is modeled as a 2d distribution of period and phase. The proposal includes a methodology for obtaining the distribution from tapping data of listeners and an evaluation metric for comparing estimated and empirical distributions.

The entire pipeline behavior was assessed on simulated tapping data. It was also tested on experimental data from listeners tapping to a self-selected beat while exposed to rhythms of different complexity. We also propose a set of reference models that estimate the beat distribution from the stimulus.

Modeling the beat considering period and phase extends previous analyses of beat ambiguity that mainly focused on tempo. The phase adds another dimension, as some beat

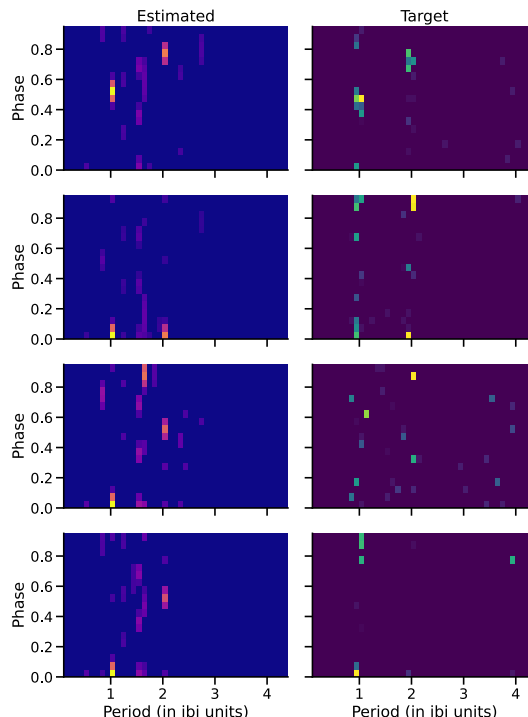


Figure 6: Sample of the two best (top rows) and two worst (bottom rows) beat distribution estimations by the *beat* model (left column).

ambiguity comes from where to tap, instead of at which speed. The modeling also allows a more detailed analysis of the concept of pulse clarity. For example, low pulse clarity may be due to multiple competing beat interpretations or to a single beat that is hard to follow. In the 2d distribution, the first scenario would be portrayed as multiple separated bins with equal probability and the second one as having all probability mass concentrated, but with no clear mode. Furthermore, the pipeline can be applied to tapping data from single or multiple listeners, exhibiting individual beat ambiguity or group disagreement.

This proposal can be further developed into modeling the distribution of beat series. In the beat tracking task, models are required to produce one beat track and therefore cannot capture situations where annotators disagree because more than one beat is reasonable. Another limitation to this approach is that it does not allow modeling non-isochronous beats. This would require a richer description of the beat which may not be as simple to visualize.

Finally, the focus on both dimensions of the beat, period and phase, is required for models of the meter. The added focus on uncertainty can be carried onto meter, yielding uncertainty on the whole rhythmic interpretation. This, in turn, can be used for the analysis of expectation in music as a mechanism driving affective responses. Most recent theories on this mechanism assign a key role to prediction error, which takes into account the certainty with which predictions of future events are made [11, 14]. Estimating the certainty of different beat estimations constitutes an initial step in this direction.

5. REFERENCES

- [1] P. A. Martens, “The Ambiguous Tactus: Tempo, Subdivision Benefit, And Three Listener Strategies,” *Music Perception*, vol. 28, no. 5, pp. 433–448, 06 2011. [Online]. Available: <https://doi.org/10.1525/mp.2011.28.5.433>
- [2] E. W. Large, J. A. Herrera, and M. J. Velasco, “Neural networks for beat perception in musical rhythm,” *Frontiers in Systems Neuroscience*, vol. 9, p. 159, 2015. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fnsys.2015.00159>
- [3] W. T. Fitch and A. J. Rosenfeld, “Perception and Production of Syncopated Rhythms,” *Music Perception*, vol. 25, no. 1, pp. 43–58, 09 2007. [Online]. Available: <https://doi.org/10.1525/mp.2007.25.1.43>
- [4] O. Lartillot, P. Toiviainen, and T. Eerola, “A matlab toolbox for music information retrieval,” in *Data Analysis, Machine Learning and Applications*, C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 261–268.
- [5] W. Trost, S. Frühholz, T. Cochrane, Y. Cojan, and P. Vuilleumier, “Temporal dynamics of musical emotions examined through intersubject synchrony of brain activity,” *Social Cognitive and Affective Neuroscience*, vol. 10, no. 12, pp. 1705–1721, 05 2015. [Online]. Available: <https://doi.org/10.1093/scan/nsv060>
- [6] G. Luck, P. Toiviainen, J. Erkkilä, O. Lartillot, K. Riikkilä, A. Mäkelä, K. Pyhälä, H. Raine, L. Varkila, and J. Värri, “Modelling the relationships between emotional responses to, and musical content of, music therapy improvisations,” *Psychology of Music*, vol. 36, no. 1, pp. 25–45, 2008. [Online]. Available: <https://doi.org/10.1177/0305735607079714>
- [7] V. E. Gonzalez-Sanchez, A. Zelechowska, and A. R. Jensenius, “Correspondences between music and involuntary human micromotion during standstill,” *Frontiers in Psychology*, vol. 9, p. 1382, 2018. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fpsyg.2018.01382>
- [8] B. Burger, M. R. Thompson, G. Luck, S. Saarikallio, and P. Toiviainen, “Music Moves Us: Beat-Related Musical Features Influence Regularity of Music-Induced Movement,” no. July, 2012, pp. 183–187. [Online]. Available: http://icmpc-escom2012.web.auth.gr/sites/default/files/papers/183_Proc.pdf
- [9] L. B. Meyer, *Emotion and meaning in music*. Chicago University Press, 1956.
- [10] D. B. Huron, *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.
- [11] P. Vuust and M. A. G. Witek, “Rhythmic complexity and predictive coding: a novel approach to modeling rhythm and meter perception in music,” *Frontiers in Psychology*, vol. 5, p. 1111, 2014. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fpsyg.2014.01111>
- [12] D. Temperley, “Computational models of music cognition,” *The psychology of music*, pp. 327–368, 2012.
- [13] C. Palmer and C. L. Krumhansl, “Mental representations for musical meter,” *Journal of Experimental Psychology: Human Perception and Performance*, vol. 16, no. 4, p. 728, 1990.
- [14] R. J. Zatorre, “Why do we love music?” in *Cerebrum: the Dana forum on brain science*, vol. 2018. Dana Foundation, 2018.
- [15] P. Vuust, M. J. Dietz, M. Witek, and M. L. Kringelbach, “Now you hear it: a predictive coding model for understanding rhythmic incongruity,” *Annals of the New York Academy of Sciences*, vol. 1423, no. 1, pp. 19–29, 2018. [Online]. Available: <https://nyaspubs.onlinelibrary.wiley.com/doi/abs/10.1111/nyas.13622>
- [16] D. Moelants and M. McKinney, “Tempo perception and musical content: What makes a piece fast, slow or temporally ambiguous?” in *Proceedings of the 8th International Conference on Music Perception and Cognition*, 2004, pp. 558–562.
- [17] M. McKinney and D. Moelants, “Deviations from the resonance theory of tempo induction,” in *Conference on Interdisciplinary Musicology*, R. Parncutt, A. Kessler, and F. Zimmer, Eds. Department of Musicology, University of Graz, 2004, pp. 124–125.
- [18] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, “An experimental comparison of audio tempo induction algorithms,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.
- [19] H. Schreiber, F. Zalkow, and M. Müller, “Modeling and estimating local tempo: A case study on chopin’s mazurkas,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR), Montreal, Quebec, Canada*, 2020.
- [20] S. Böck, F. Krebs, and G. Widmer, “Accurate tempo estimation based on recurrent neural networks and resonating comb filters,” in *ISMIR*, 2015, pp. 625–631.
- [21] S. Böck, F. Krebs, and G. Widmer, “Joint beat and downbeat tracking with recurrent neural networks,” in *ISMIR*, 2016, pp. 255–261.
- [22] M. A. Miguel, M. Sigman, and D. Fernandez Slezak, “From beat tracking to beat expectation: Cognitive-based beat tracking for capturing pulse clarity through time,” *PLOS ONE*, vol. 15, no. 11, pp. 1–22, 11 2020. [Online]. Available: <https://doi.org/10.1371/journal.pone.0242207>

- [23] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2006.
- [24] S. Fujii, M. Hirashima, K. Kudo, T. Ohtsuki, Y. Nakamura, and S. Oda, "Synchronization error of drum kit playing with a metronome at different tempi by professional drummers," *Music Perception: An Interdisciplinary Journal*, vol. 28, no. 5, pp. 491–503, 2011.
- [25] B. H. Repp and Y.-H. Su, "Sensorimotor synchronization: a review of recent research (2006–2012)," *Psychonomic bulletin & review*, vol. 20, no. 3, pp. 403–452, 2013.
- [26] B. H. Repp and R. Doggett, "Tapping to a very slow beat: a comparison of musicians and nonmusicians," *Music Perception*, vol. 24, no. 4, pp. 367–376, 2007.
- [27] M. A. Miguel, P. Riera, and D. Fernández Slezak, "A simple and cheap setup for timing tapping responses synchronized to auditory stimuli," *Behavior Research Methods (in press)*, 2021. [Online]. Available: <https://doi.org/10.3758/s13428-021-01653-y>
- [28] D.-J. Povel and P. Essens, "Perception of temporal patterns," *Music Perception: An Interdisciplinary Journal*, vol. 2, no. 4, pp. 411–440, 1985.
- [29] G. Peyré and M. Cuturi, "Computational optimal transport," 2020.
- [30] N. I. Fisher, *Statistical Analysis of Circular Data*. Cambridge University Press, 1993.
- [31] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing, "On tempo tracking: Tempogram representation and kalman filtering," *Journal of New Music Research*, vol. 29, no. 4, pp. 259–273, 2000.

A CASE STUDY OF DEEP ENCULTURATION AND SENSORIMOTOR SYNCHRONIZATION TO REAL MUSIC

Olof Misgeld^{1,2}

Torbjörn Gulz^{1,2}

Andre Holzapfel²

Jūra Miniotaitė²

¹ KMH Royal College of Music, Sweden

² KTH Royal Institute of Technology, Sweden

misgeld@kth.se, gulz@kth.se

ABSTRACT

Synchronization of movement to music is a behavioural capacity that separates humans from most other species. Whereas such movements have been studied using a wide range of methods, only few studies have investigated synchronisation to real music stimuli in a cross-culturally comparative setting. The present study employs beat tracking evaluation metrics and accent histograms to analyze the differences in the ways participants from two cultural groups synchronize their tapping with either familiar or unfamiliar music stimuli. Instead of choosing two apparently remote cultural groups, we selected two groups of musicians that share cultural backgrounds, but that differ regarding the music style they specialize in. The employed method to record tapping responses in audio format facilitates a fine-grained analysis of metrical accents that emerge from the responses. The identified differences between groups are related to the metrical structures inherent to the two musical styles, such as non-isochronicity of the beat, and differences between the groups document the influence of the deep enculturation of participants to their style of expertise. Besides these findings, our study sheds light on a conceptual weakness of a common beat tracking evaluation metric, when applied to human tapping instead of machine generated beat estimations.

1. INTRODUCTION

Feeling the beat is universal to music experience and production, and rhythmical patterns are determinant for musical genres across the globe. *Enculturation* concerns the influence of the surrounding cultural environment on the development of individuals' perception, cognition and behavior. *Sensorimotor synchronization* (SMS) regulates how humans synchronize their behavior to time ordered stimuli in various sensory modalities [1]. The enculturation in musical contexts has been explored in auditory SMS studies on the perception and reproduction of rhythms in

different cultural contexts [2]. However, most SMS studies have used generated stimuli, and only a small number of studies – including cross-cultural studies – have had humans tapping along with real music [3]. In general, research on enculturation challenges the dominance of Western music in music perception and cognition research [4,5].

Previous studies in rhythm perception have been based on subjects with highly different cultural and ethnic backgrounds [2–9]. The present study compares musicians within a similar cultural group: music students at the same higher music education institution, with specialization in either one of the two genres. Hence, the present study adds to previous research on the role of deep enculturation in rhythm perception by examining sensorimotor synchronization of musicians in two different musical genres.

We ask whether deep enculturation as practitioners leads to higher agreement in sensorimotor synchronization to music stimuli among the group of musicians practicing the particular style. Furthermore, we explore how emerging differences between groups of musicians are tied to genre-specific musical parameters. To this end, we record subjects from two groups of musicians tapping the beat to music examples from two genres. Our analysis employs a combination of computational measures and qualitative analysis in order to shed light on genre-specific interpretations of musical meter. We apply beat tracking evaluation metrics in order to estimate the degree of agreement between the tapping responses in the two groups of musicians. In addition to the computational agreement estimate, recording the tapping in audio format enables us to capture both the time instances of responses and their dynamic emphasis. Based on this information, we explore the relation between tapping responses and metrical structure based on histograms of the recorded tapping responses.

The genres included in this study are jazz music and Scandinavian traditional folk music, both known for their intricate rhythmical structures. The choice of these two genres was further motivated by the fact that they are both taught at the Royal College of Music in Stockholm. This results in an environment to recruit participants who share a common cultural background, but differ mainly in terms of the musical style in which they have particular expertise.

In Scandinavian folk music, some triple meter dance music forms include styles with non-isochronous, asymmetric beat patterns [10, 11]. Although these patterns are



well-documented in contemporary Nordic folk music and dance practice, behavioral responses to this music have not yet been approached in the context of SMS studies. In jazz, on the one hand, a large part of the repertoire consists of music with a definite relation to the beat, which is controlled by quarter notes played in the bass and the ride cymbal [12]. On the other hand, in some jazz music the main beat is not connected to a specific instrument. Identifying the beat and subdivisions can therefore be challenging for a listener. Based on these genre characteristics we expect varying agreements between the two groups of musicians in three aspects: the dynamic emphasis of the beats in a measure, concerning non-isochronous asymmetric beat patterns in Scandinavian folk music, and on the main metrical level in jazz music.

The remainder of the paper is structured as follows: in Section 2 we refer to relevant research on sensorimotor synchronization, enculturation, mutual agreement metrics, and rhythm and meter perception within the two genres. Section 3 describes the experimental setup and methods for data collection and analysis, including the process of correcting the mutual agreement metrics for a detected tempo bias caused by human tapping motor noise. In Section 4 we present the results of our analysis, which are further discussed in Section 5.

2. BACKGROUND-RELATED WORK

2.1 Sensorimotor synchronization and tapping studies

One of the most common experimental setups to explore SMS is by means of tapping studies, where the main goal is to examine subjects' ability to coordinate hand or finger movement to rhythm-stimuli. These stimuli usually consist of relatively simple rhythms synthesized using click sounds, and subjects are instructed to synchronize with the stimuli as accurately as possible. As summarized by [1, 13], synchronisation is characterized by a negative asynchrony with a variability of the standard deviation of the asynchronies (SD_{asy}) depending on the intervals in stimuli and tapping responses.

In discourse about music the concepts of time and timing are used in many ways, frequently with a judgemental connotation. One way to quantitatively examine timing is to perform tapping experiments, and there has been relatively much research performed that indicates higher tapping accuracy for professional musicians than for non-musicians. For instance, magnetic resonance experiments showed how professional pianists had a faster and different learning process in complicated tapping attempts [14]. For pianists, however, tapping can be regarded as similar to everyday practice at the instrument. It has been shown by [15] that being in their proper environment with their instruments helps musicians to perform with a significantly lower synchronization error when playing the drum set than in previous tapping experiments.

Whereas the vast majority of tapping studies has been conducted with simple rhythmic stimuli, tapping data obtained when using musical stimuli can provide addi-

tional information. Palmer and Krumhansl describe how more experienced listeners/practitioners use subdivisions to identify meter and beats with more confidence [16]. London et al. [17] move a step further and question the traditional Western way of identifying beats based on melodic and rhythmic accents and argue that the rhythmic organization in certain music styles can be based on contra-metricity: a significant portion of note onsets tend to be non-congruent with the metrical framework. Here, they base their arguments on research on drum ensembles in Mali [18] and Turkish modal art music [19]. Hence, tapping studies with real music stimuli may provide valuable information when the informants are musicians with in-depth knowledge of music structure.

2.2 Enculturation and meter perception

Cross-cultural studies have provided evidence for the effects of enculturation on various aspects of music perception and cognition, and several studies have compared the perception of rhythmical and metrical categories between distant cultural groups [4, 5]. Experimental studies [3, 6, 7, 20] indicate that, for rhythm perception, familiarity is a more important factor than complexity based on integer ratios. The perception of rhythm is known to be primed from the metrical context [21], and practitioners of non-Western musical cultures have been shown to accurately represent the complex asynchronous rhythmic patterns common in their respective genre [2]. Studies have explored rhythm perception and enculturation at early ages [22], and [9] have reported effects from passive, short-term exposure on children's perception of non-isochronous meter. Drawing on these findings, recent works have presented a probabilistic model simulating enculturation in meter perception, with predictive coding inferred from previous rhythmic exposure [23].

2.3 Scandinavian traditional folk music

Polska, springlek, pols and springar are Swedish and Norwegian triple meter music and dance forms that in regional sub-forms include asymmetric beat [10, 11]. In addition to the asymmetry on the beat level, sub-beat rhythm patterns are often non-isochronous and un-even [24], so that asymmetric beat lengths cannot be attributed from simply adding equal shorter pulse units [25, 26]. Rather, it has been suggested that beat asymmetry should be understood as uneven subdivisions of an isochronous *common slow pulse* at measure level, and in relation to the periodicity of dance movements [25]. Furthermore, variations in asymmetric beat lengths occur within performances, corresponding with distinct melodic rhythm gestalt patterns [27]. The close connection with dance is reflected in how musicians dynamically articulate beat cycles [27]. As a consequence, understanding the metrical structure requires some familiarity with these specific musical forms. As this music is often performed solo, on violin or fiddle, with plenty of ornamentation and complex bowing patterns, researchers [28] have pointed to the challenge of finding sound events that precisely correspond to the experienced

musical beat. The Norwegian folk musician and researcher Groven reported tapping on a morse transmitter for measuring uneven beat ratios in springar/polska music already in the 1930-ties [29], more recent studies have applied tapping [30], sound graph analysis [11, 24, 30], and motion capture [31, 32] for analyzing asymmetric beat patterns in Scandinavian folk music. However, at this point we are not aware of tapping studies comparing musicians from different genres encountering these music forms.

2.4 Jazz music

Jazz music developed rapidly during the 20th century and now consists of many different sub-genres from early jazz, such as Dixieland and swing, to modern jazz such as modal jazz and free jazz [33]. Common to the different specializations is that they all contain several complex parts in terms of melody, harmony, and rhythms. The rhythmic complexity becomes particularly evident in the more modern jazz, where nontraditional meters and intricate percussive subdivisions are used [12]. The complexity includes different overlay techniques like the placement of 2- or 4-cycle patterns over 3/4 meter and 3/4 or 3/2 patterns over 4/4 meter. The musicians have to agree on where the – sometimes hard to discern – beat is while playing. Additional difficulty for an inexperienced listener to perceive the beat may emerge when the beat is not directly marked by any instrument. Studies in cognitive neuroscience explored the pre-attentive brain response to various musical parameter changes and reported high measures for jazz musicians compared to classical and rock/pop musicians [34]. Other studies suggest listening, but above all, performing music in specific genres generates a clearer meter perception [7]. Many jazz tunes are also performed in a high tempo (up to 400 BPM), which also challenges the musician’s technique and the listener’s beat perception.

2.5 Mutual agreement

Most of the evaluation in SMS experiments is based on statistical analysis of the asynchronies between participant responses and stimulus onsets (see Section 1 in [1] for an overview). Such an analysis would require the compilation of a reference beat for the music stimuli, which is a problematic procedure since not all beats coincide with note onsets. In the absence of a reference annotation, previous work [35] has suggested to employ beat tracking evaluation metrics to estimate the mutual agreement between beat estimates obtained from beat tracking algorithms. This approach has been used to analyze the agreement between human beat tapping sequences in the context of exploring music collections [36]. Among various employed metrics, information gain [37] was found to produce reliable estimates for such mutual agreement.

3. METHOD

3.1 Apparatus

Music stimuli were presented through studio-monitoring headphones (Beyerdynamic DT 700 PRO headset). The

experimental setup applied first by [38] was used to record participant tapping. It consists of a sensor made from soft material, with a microphone attached to the surface of the sensor. A Focusrite 6i6 USB sound card was used to simultaneously record the microphone output and a split of the headphone signal in two channels of a stereo wav file.

3.2 Stimuli

20 stimuli of jazz and 20 stimuli of Scandinavian folk music were chosen by the first and second author, who are performers and teachers within each of these genres, and affiliated to the same higher music education institution as the participants¹. In addition, two practice stimuli were selected for each style. Each excerpt is about 42 seconds long. The measure onsets were manually annotated by the two first authors for later analysis. The tempi – based on these annotations – were between 109 to 155 beats per minute (BPM) (M=135, SD=12) for folk music and 55 and 294 BPM (M=183, SD=66) for jazz, respectively. Whereas the tempo means are similar, the Scandinavian folk examples, all related to dance, have a much smaller range of tempi. The Jazz stimuli in this study are mainly recorded between 1960-2000, originating in American post-bebop or a Nordic jazz tradition. The ensembles are mostly piano trios or quartets with one horn player.

3.3 Participants

The participants for the experiment were advanced students of either jazz or folk music programs at the Royal College of Music in Stockholm. In total 8 jazz musicians and 9 folk musicians participated in the study. Participants were between 20 and 29 years old (mean = 23.5 years), 6 male and 11 female.

Before participating in the experiment, each participant filled out a questionnaire with questions about their education, their experience with different musical genres, and their dancing experience. All participants lived in Stockholm and spoke Swedish. Many participants had a background in playing other genres beyond the focus of their study program. However, no jazz musicians stated that they play folk music, and only one folk musician stated that they have experience playing jazz music. The participants had had regular practice on a musical instrument or singing between 4 and 18 years (mean = 10.6 years). All folk musicians answered that they danced on occasion and had dancing as part of their study curriculum. On the other hand, only three jazz musicians danced occasionally whereas the other five jazz musicians responded that they never dance.

3.4 Procedure

The experiment started with the participant signing a consent form and then receiving verbal instructions on using the equipment. They were then instructed to tap the beat to the music excerpts they were going to listen to, and they

¹ Music stimuli, tapping data and complementary results histograms are provided here: <https://bit.ly/3uJ7EC7>.

were asked to emphasize the beat that they considered the first in each measure. Before presenting the musical stimuli, an isochronous clicking track with IOI of 0.5s was played, and the participant was asked to tap in synchrony to the sound. This served the purpose to adjust playback volume, and to check the amplitude of the sensor output. After that, music stimuli were presented in two blocks, one for each music genre. Each block started with the two practice stimuli, followed by the 20 stimuli in randomized order. Further, the order of the blocks was divided so that half of the participants from each group started with the jazz stimuli, and the other half started with the folk music stimuli. The whole experiment took about 50 minutes, including a concluding discussion in which participants had the opportunity to ask questions about the project.

3.5 Data analysis

The recorded responses were analyzed using a simple thresholding as proposed by [38], resulting in a list of time instances for tap locations for each response. We conducted three types of quantitative analyses using the responses and the obtained tap annotations:

1) Mutual agreement: Information Gain (IG) (see Section 2.5) was calculated between all tapping annotations of each group for a particular stimulus. The IG takes on values between 0 and 5.3, with low numbers indicating low agreement. Preliminary experiments indicated low IG values for human tapping responses as compared to values reported for accurate beat trackers, with a decrease of IG values for increasing tempi. To investigate this trend, the authors recorded their own tapping responses to isochronous clicks at four IOI rates between 60 and 180 bpm. From these sequences, a similar linear decrease was observed for IG with increasing tempo, as displayed in Figure 1. The slope (-0.0048) and intercept (3.232) of this trend were determined using linear regression. Using the information of the annotated tempi of the stimuli, the IG values for comparisons between the participants' responses were corrected using these regression values. It is worth pointing out that such a problem of a beat tracking metric has not been observed so far, and it is caused by the overall standard deviation of human tapping [1], a phenomenon absent from beat tracking algorithms.

2) Inter-tap intervals (ITI): Using the tap annotations, histograms of ITI were calculated for each group and stimulus. These were then analyzed in relation to the annotated tempo to investigate if certain metrical levels tend to be preferred depending on group and musical style.

3) Accent histograms: Whereas the previous two analyses use the tap annotations as input, the histogram analysis uses the recorded responses in which emphases of the first beat of a bar are reflected. Each response is differentiated over time and half-wave rectified to avoid eventual cancellation of positive and negative values. Each response is then normalized to have a maximum magnitude of one to compensate for varying tapping intensities between participants. All recordings have been annotated with the bar positions, and using this information the normalized re-

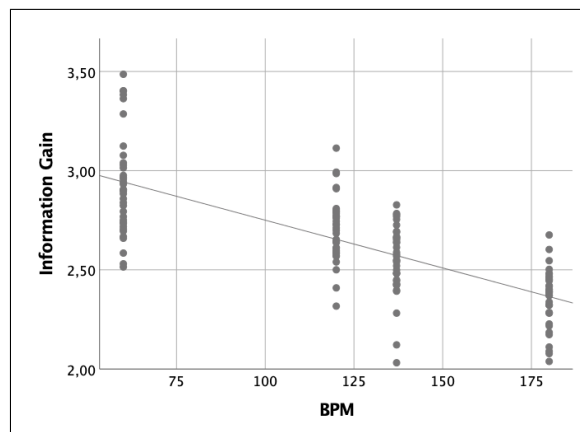


Figure 1. Linear decrease in Information Gain computed from the authors' tapping to isochronous clicks at four different IOI rates.

sponses are divided into bar length segments. To create histograms of equal length independent from tempo, each such bar length segment is divided into 60 equal-duration bins, and the values of the normalized response between the bin boundaries are added. For a whole song, these bar-length histograms are added for each individual participant, and the mean histogram across all participants of a group for each song is computed. These histograms will be analyzed regarding the relation between the strongest, and the second strongest peak, which will provide an estimate for the average emphasis of a downbeat position. Furthermore, the positions of the peaks will facilitate an analysis of the perceived position of the downbeat, and the degree of non-isochronous tapping (especially for Scandinavian folk music).

The two first authors individually annotated all files with aspects that make these files potentially difficult to tap to. These annotations will provide further background for the contextualization of the above-listed three quantitative analysis methods.

4. RESULTS

Our analysis resulted in tempo corrected information gain scores, ITI histograms and accent histograms for each group and music stimulus. The mean tempo corrected information gain scores for each stimulus within each group of musicians were analyzed using n-way ANOVA tests to determine if the averages in the dataset differed with respect to the type of musician and genre. The music genre of the stimuli had an impact on the agreement of the musicians as a whole ($F(\text{Genre}) = 34.62, p < 0.001$), with an interaction effect between genres and groups of musicians ($F(\text{Musician} * \text{Genre}) = 7.62, p = 0.007$). The average score for each combination of musician and music genre in Figure 2 reveals that jazz musicians agreed more on how to tap the beat to jazz music than how to tap the beat to folk music. At the same time, there was no significant difference between the genres for the folk musicians. The ANOVA indicated no difference between the overall performance

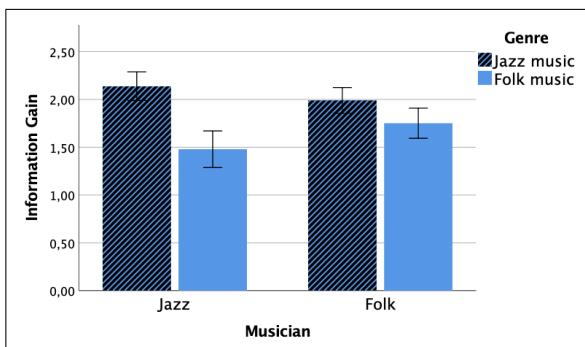


Figure 2. Mean tempo corrected information gain between groups of jazz (N=8) and folk (N=9) musicians tapping to jazz (N=20) and Scandinavian folk (N=20) music stimuli. 95% confidence intervals in brackets.

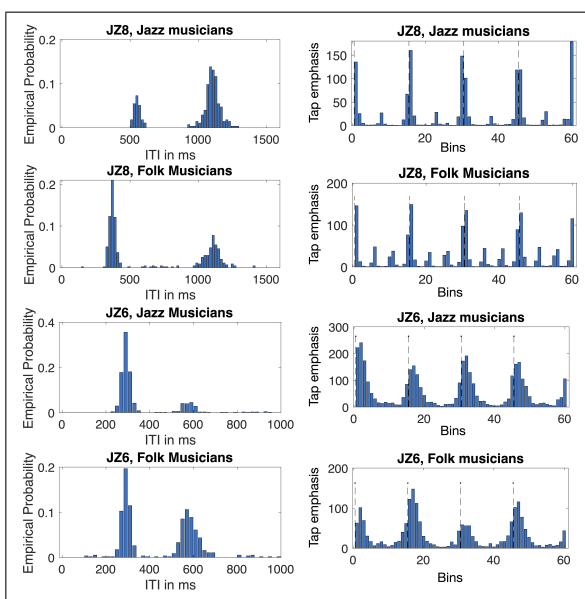


Figure 3. Tapping to jazz. ITI (left) and accent (right) histograms for the two groups of musicians with two examples of stimuli in slow (JZ8, 55 BPM) and fast (JZ6, 206 BPM) tempo. The dashed vertical lines in the accent histograms show isochronous beat positions.

of the two groups of musicians ($F(\text{Musician}) = 0.64, p = 0.426$).

The ITI and accent histograms facilitate a more detailed and complementary analysis of differences between the groups of musicians’ tapping behavior in the different examples. These differences relate to the preferred metrical level in jazz, to tapping with non-isochronous beat in folk music, and to the accentuation of metrical periodicity in folk music.

The ITI histograms displayed in Figure 3 exemplify jazz stimuli where the two groups tapped at different metrical levels. All jazz music stimuli were in duple meters, with four beats per measure, and with an annotated tempo between 55 and 294 BPM. In all jazz stimuli with tempi >200 BPM, a portion of the folk musicians’ tapping was at half the tempo compared to the majority of jazz musicians’. The accent histograms for some of these show

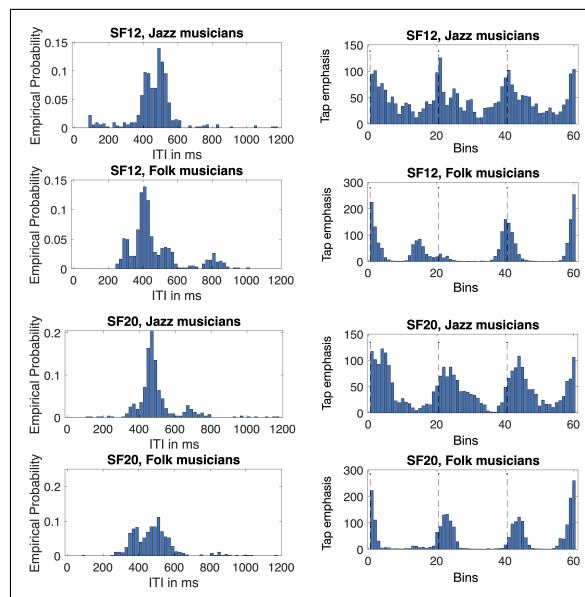


Figure 4. Tapping to Scandinavian folk music. ITI (left) and accent (right) histograms for the two groups of musicians with two examples of stimuli with asymmetric beat: *polska* with short-long-medium (SF12, 146 BPM), and long-medium-short (SF20, 130 BPM) beat patterns. The dashed vertical lines in the accent histograms show isochronous beat positions.

peaks on beat 2 and 4 for folk musicians, indicating that the half-tempo tapping was partly phase-shifted so that folk musicians tapped only on beat two and four of the four beats tapped by jazz musicians (JZ6 in Figure 3). In the two slowest jazz examples, with annotated tempi of 55 and 60 BPM, the group of folk musicians partly tapped at the triple tempo. In contrast, the group of jazz musicians stayed at the annotated tempo level or in one song partly tapped in double tempo, as exemplified by JZ8 in Figure 3.

Figure 4 illustrates two folk music stimuli where folk musicians tapped along with the asymmetric beat. The ITI histogram for folk musicians tapping to stimuli SF12 displays a peak around 300ms corresponding to a shorter beat, and the related accent histogram displays a short-long-medium pattern, including some variability in the tapping of the second beat. This variability could be explained by the *polska*-beat itself - varying between more or less acute asymmetry. However, listening to the recorded tapping responses for these tunes revealed that, among folk musicians, some were tapping an isochronous beat and others an asymmetric beat. For the second stimuli (SF20) depicted in Figure 4, the histograms illustrate that folk musicians tap consistent with a long-medium-short asymmetric beat pattern. For jazz musicians tapping to folk music stimuli with an asymmetric beat, no consistent non-isochronous tapping was found from these histograms. Instead, it appears that jazz musicians were tapping isochronously but with a low agreement between the musicians.

For most folk music stimuli, with folk musicians tapping, the triple meter was distinguishable in the accent his-

tograms, with a stronger signal on the first beat. The second beat then appeared weaker, and the third beat was a little bit stronger than the second, as exemplified with stimuli SF12 in Figure 4. No consistent metrical accentuation was found for jazz musicians tapping to folk, and for tapping to jazz, the first beat of four was accentuated only with a few of the stimuli.

Additionally, some observations were made during the experiments. All of the participants in the experiment marked metrical structures in addition to the tapping on the cushion. For instance, participants nodded their heads, tapped with their other hand on a surface next to them, or snapped their fingers. Most of the folk musicians were surprised when they were asked to tap the beat with their hands and to tap all the beats in the measure. One folk musician said, “It is more important to keep the beat with the feet, at least for us. We are used to stomping the beat on one and three”. Almost all of the folk musicians were tapping with their feet in addition to tapping fingers on the cushion. When hearing examples from the genre that the participants were not used to, all participants at first reacted to signal that they experienced discomfort. They either made a facial expression signaling surprise, or they laughed, signaling that they found the task difficult. Most of the participants stated afterward that they were not used to listening to the genre that they were less familiar with.

Two of the jazz musicians pointed out that it was harder to distinguish a clear beat pattern with folk music, since many folk music examples had only one instrument playing, while the jazz music examples had multiple instruments playing.

5. DISCUSSION

Cross-cultural studies shed light on universals and specifics in music cognition and perception [5]. This study adds to previous findings on the role of enculturation in the context of music meter perception by comparing advanced musicians trained in different genres but with otherwise similar cultural backgrounds. Our analysis of differences between groups combined computational measurements with analysis by genre-experts. A mutual agreement estimate (see Section 2.5) indicated significantly higher agreement among jazz musicians when tapping to jazz music than when tapping to Scandinavian folk music. Further analysis of tapping patterns and the dynamic emphasis of beats, using ITI and accent histograms, provided more insights into group-specific interpretations of meter and beat. For jazz musicians tapping to jazz, we found a more consistent alignment to the extreme tempi, compared to folk musicians who were more likely to tap at half-tempo (for faster tempi) or sub-divide (for slower tempi). For example, marking the triple-subdivision - the “swing” - at a slow tempo (see Figure 3) is mainly encountered among folk musicians. Furthermore, part of the non-jazz experts’ tapping was phase-shifted, marking only beat 2 and 4.

The folk music examples were all ternary meter, including non-isochronous, asymmetric styles of polska. In addition to this metric particularity, these tunes were performed

solo on bowed instruments with only occasional accompaniment of foot-tapping. These properties pose challenges for inexperienced listeners: the lack of clear transients at beat positions, more limited spectral spread of information, and, in general, that dance movements and foot-tapping are complementary to how rhythms relate to meter in these styles [10, 32] (audible foot-tapping on beat 1 and 3 was detected by the authors only in four of the 20 stimuli). Consequently, folk musicians tapped along with asymmetric beat patterns while jazz musicians, expecting beats to be isochronous, failed in finding consistent beat patterns, which resulted in a low agreement between the jazz musicians. Furthermore, the dynamic emphasis of beats in folk musicians’ tapping to folk music were consistent with descriptions of metric beat articulations [27].

We found a dependency between the mutual agreement metric and tempo in our material. We conducted an additional experiment, tapping to generated clicks at different IOIs, which confirmed this dependency and provided a correction factor for our results. Hence, the standard deviation of human tapping (SD_{asy}) introduces a bias in the computational metric, which so far had been employed for automatic beat tracking evaluation mainly. Further studies should investigate the robustness of beat tracking metrics in presence of motor noise typical for human production.

Our study used a selection of commercial and archive music recordings, which included genre-specific differences in conventions, settings and instruments. For instance, all jazz examples featured ensemble playing while folk music examples were all played solo on violin or hardingfela. Although studies could benefit from more neutrally designed stimuli, these differences reflect standard practice in these genres and thus reflect real-world situations that these musicians are likely to face.

6. CONCLUSION

This study employed mutual agreement metrics, ITI and accent histograms to analyse the sensorimotor synchronization of two groups of musicians when tapping to music from familiar and unfamiliar genres. We found group- and genre-specific behaviours for tapping with the main metrical level, tapping with asymmetric beat and the accentuation of beat cycles. The musicians shared the same cultural background but were specialist in either jazz or Scandinavian folk music, and our findings show a coherence with genre-specific meter conventions as a result of deep genre expertise. In addition, we identified a tempo bias in the mutual agreement metric caused by human motor noise, which motivates future studies of the validity of beat tracking metrics when applied to human tapping.

7. ACKNOWLEDGMENTS

Andre Holzapfel was supported by the Swedish Research Council (2019-03694) and the Marianne and Marcus Wallenberg Foundation (MMW 2020.0102).

8. REFERENCES

- [1] B. H. Repp and Y. H. Su, "Sensorimotor synchronization: A review of recent research (2006-2012)," *Psychonomic Bulletin and Review*, vol. 20, no. 3, pp. 403–452, 2013.
- [2] R. Polak, N. Jacoby, T. Fischinger, D. Goldberg, A. Holzapfel, and J. London, "Rhythmic prototypes across cultures: A comparative study of tapping synchronization," *Music Perception: An Interdisciplinary Journal*, vol. 36, no. 1, pp. 1–23, 2018.
- [3] C. Drake and J. B. El Heni, "Synchronizing with Music: Intercultural Differences," *Annals of the New York Academy of Sciences*, vol. 999, no. 1, pp. 429–437, nov 2003.
- [4] C. J. Stevens, "Music Perception and Cognition: A Review of Recent Cross-Cultural Research," *Topics in Cognitive Science*, vol. 4, no. 4, pp. 653–667, oct 2012.
- [5] S. J. Morrison and S. M. Demorest, "Cultural constraints on music perception and cognition," in *Progress in Brain Research*, J. Chiao, Ed. Elsevier, 2009, vol. 178, no. C, pp. 67–77.
- [6] C. M. Yates, T. Justus, N. B. Atalay, N. Mert, and S. E. Trehub, "Effects of musical training and culture on meter perception," *Psychology of Music*, vol. 45, no. 2, pp. 231–245, 2017.
- [7] B. Kalender, S. E. Trehub, and E. G. Schellenberg, "Cross-cultural differences in meter perception," *Psychological Research*, vol. 77, no. 2, pp. 196–203, 2013.
- [8] N. Jacoby and J. H. McDermott, "Integer Ratio Priors on Musical Rhythm Revealed Cross-culturally by Iterated Reproduction," *Current Biology*, vol. 27, no. 3, pp. 359–370, 2017.
- [9] E. E. Hannon, C. M. Vanden Bosch der Nederlanden, and P. Tichko, "Effects of perceptual experience on children's and adults' perception of unfamiliar rhythms," *Annals of the New York Academy of Sciences*, vol. 1252, no. 1, pp. 92–99, 2012.
- [10] J.-P. Blom, "The Dancing Fiddle, On the Expression of Rhythm in Hardingfele Slåtter," in *Norwegian folk music Series 1 Slåtter for the harding fiddle Vol. 7 Springar in 3/4 time*, ser. Norsk folkemusikksamling. Oslo: Universitetsforlaget, 1981, vol. Serie 1 B., p. 289.
- [11] S. Ahlbäck, "Metrisk analys - en metod att beskriva skillnaden mellan låttyper," *Norsk folkemusikklags skrifter*, no. 4, 1989.
- [12] J. Pressing, "Black Atlantic Rhythm: Its Computational and Transcultural Foundations," *Music Perception*, vol. 19, no. 3, pp. 285–310, 2002.
- [13] B. H. Repp, "Sensorimotor synchronization: A review of the tapping literature," *Psychonomic Bulletin & Review*, vol. 12, no. 6, pp. 969–992, 2005.
- [14] M. Hund-Georgiadis and D. Yves Von Cramon, "Motor-learning-related changes in piano players and non-musicians revealed by functional magnetic-resonance signals," *Experimental Brain Research*, vol. 125, no. 4, pp. 417–425, 1999.
- [15] S. Fujii, M. Hirashima, K. Kudo, T. Ohtsuki, Y. Nakamura, and S. Oda, "Synchronization error of drum kit playing with a metronome at different tempi by professional drummers," *Music Perception: An Interdisciplinary Journal*, vol. 28, no. 5, pp. 491–503, 2011.
- [16] C. Palmer and C. L. Krumhansl, "Mental Representations for Musical Meter," *Journal of experimental psychology. Human perception and performance*, vol. 16, no. 4, pp. 728–741, 1990.
- [17] J. London, R. Polak, and N. Jacoby, "Rhythm histograms and musical meter: A corpus study of Malian percussion music," *Psychonomic Bulletin and Review*, vol. 24, no. 2, pp. 474–480, 2017.
- [18] R. Polak, "Rhythmic Feel as Meter: Non-Isochronous Beat Subdivision in Jembe Music from Mali," *Music Theory Online*, vol. 16, no. 4, pp. 1–26, 2010.
- [19] A. Holzapfel, "Relation Between Surface Rhythm and Rhythmic Modes in Turkish Makam Music," *Journal of New Music Research*, vol. 44, no. 1, pp. 25–38, 2015.
- [20] E. E. Hannon, G. Soley, and S. Ullal, "Familiarity overrides complexity in rhythm perception: A cross-cultural comparison of American and Turkish listeners," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 38, no. 3, pp. 543–548, 2012.
- [21] P. Desain and H. Honing, "The formation of rhythmic categories and metric priming," *Perception*, vol. 32, no. 3, pp. 341–365, 2003.
- [22] E. E. Hannon, G. Soley, and R. S. Levine, "Constraints on infants' musical rhythm perception: effects of interval ratio complexity and enculturation," *Developmental Science*, vol. 14, no. 4, pp. 865–872, 2011.
- [23] B. van der Weij, M. T. Pearce, and H. Honing, "A probabilistic model of meter perception: Simulating enculturation," *Frontiers in Psychology*, vol. 8, no. MAY, pp. 1–18, 2017.
- [24] I. Bengtsson, "On notation of time, signature and rhythm in Swedish polskas," *Studia instrumentorum musicae popularis III.*, pp. 22–31, 1974.
- [25] T. Kivite, "Categories and timing: On the perception of meter," *Ethnomusicology*, vol. 51, no. 1, pp. 64–84, 2007.

- [26] M. Nilsson, “The swedish polska,” *Stockholm: Svenskt visarkiv/Musikverket*, 2017.
- [27] S. Ahlbäck, “About Asymmetrical Beat in the Polska,” in *The Polish Dance in Scandinavia and Poland*, M. Ramsten, Ed. Stockholm: Svenskt visarkiv, 2003, p. 165–80.
- [28] M. Johansson, “Rhythm into style: studying asymmetrical grooves in Norwegian folk music,” Ph.D. dissertation, University of Oslo, 2009.
- [29] E. Groven, “Musikkstudiar-ikkje utgjevne før. 1. Rytimestudiar,” *Heiderskrift til Eivind Groven.*, pp. 93–102, 1971.
- [30] T. Kvifte, “Fenomenet ’asymetrisk’ takt i norsk og svensk folkemusikk.” *Studia Musicologica Norvegica*, no. 25, pp. 387–430, 1999.
- [31] M. R. Haugen, “Investigating Periodic Body Motions as a Tacit Reference Structure in Norwegian Tele-springar Performance,” *Empirical Musicology Review*, vol. 11, no. 3-4, p. 295, 2017.
- [32] O. Misgeld and A. Holzapfel, “Towards the study of embodied meter in Swedish folk dance,” in *Folk Music Analysis workshop*, no. Mid, 2018, p. 6.
- [33] P. F. Berliner, *Thinking in jazz: The Infinite Art of Improvisation*. The University of Chicago Press, 1994.
- [34] P. Vuust, E. Brattico, M. Seppänen, R. Näätänen, and M. Tervaniemi, “Practiced musical style shapes auditory skills,” *Annals of the New York Academy of Sciences*, vol. 1252, no. 1, pp. 139–146, 2012.
- [35] A. Holzapfel, M. E. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, “Selective sampling for beat tracking evaluation,” *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.
- [36] O. Cornelis, J. Six, A. Holzapfel, and M. Leman, “Evaluation and recommendation of pulse and tempo annotation in ethnic music,” *Journal for New Music Research*, vol. 42, no. 2, pp. 131–149, 2013.
- [37] M. E. P. Davies, N. Degara, and M. D. Plumbley, “Evaluation methods for musical audio beat tracking algorithms,” Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06, 2009. [Online]. Available: <http://www.elec.qmul.ac.uk/people/markp/2009/DaviesDegaraPlumbley09-evaluation-tr.pdf>
- [38] N. Jacoby and J. H. McDermott, “Integer ratio priors on musical rhythm revealed cross-culturally by iterated reproduction,” *Current Biology*, vol. 27, no. 3, pp. 359–370, 2017.

SYMBOLIC MUSIC GENERATION WITH DIFFUSION MODELS

Gautam Mittal^{1*} Jesse Engel² Curtis Hawthorne² Ian Simon²

¹ University of California, Berkeley ² Google Brain

gbm@berkeley.edu, {jesseengel, fjord, iansimon}@google.com

ABSTRACT

Score-based generative models and diffusion probabilistic models have been successful at generating high-quality samples in a variety of continuous domains. However, due to their Langevin-inspired sampling mechanisms, their application to discrete symbolic music data has been limited. In this work, we present a technique for training diffusion models on symbolic music data by parameterizing the discrete domain in the continuous latent space of a pre-trained variational autoencoder. Our method is non-autoregressive and learns to generate sequences of latent embeddings through the reverse process and offers parallel generation with a constant number of iterative refinement steps. We show strong unconditional generation and post-hoc conditional infilling results compared to autoregressive language models operating over the same continuous embeddings.

1. INTRODUCTION

Denosing diffusion probabilistic models (DDPMs) [1, 2] are a promising new class of generative models that can synthesize comparably high-quality samples by learning to invert a diffusion process from data to Gaussian noise. Unlike many existing deep generative models, DDPMs sample through an iterative refinement process inspired by Langevin dynamics [3], which enables post-hoc conditioning of models trained unconditionally [4–7] for creative applications.

Despite these exciting advances, DDPMs have not yet been applied to symbolic music generation because their iterative refinement sampling process is confined to continuous domains such as images [2] and audio [8, 9]. Similarly, DDPMs cannot take advantage of the recent advances in modeling long-term structure [10–12] that use a two-stage process of modeling discrete tokens extracted by a separate low-level autoencoder.

In this paper, we demonstrate that it is possible to overcome these limitations by training DDPMs on the continuous latents of a low-level variational autoencoder (VAE)

* Work completed during an internship at Google Brain.

to generate long-form discrete symbolic music. Our key findings include:

- High-quality unconditional sampling of discrete melodic sequences (1024 tokens) with DDPMs through iterative refinement of lower-level VAE latents.
- DDPMs outperforming strong autoregressive baselines (TransformerMDN) in hierarchical modeling of continuous latents, partly due to a lack of teacher forcing and exposure bias during training.
- Post-hoc conditional infilling of melodic sequences for creative applications.

2. BACKGROUND

2.1 Denosing Diffusion Probabilistic Models

DDPMs [1, 2] are a class of generative models that define latents x_1, \dots, x_N of the same dimensionality as the data $x_0 \sim q(x_0)$. Diffusion models are comprised of a **forward process** and a **reverse process**. The **forward process** starts from the data x_0 and iteratively adds Gaussian noise according to a fixed noise schedule for N diffusion steps:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I) \quad (1)$$

$$q(x_{1:N}|x_0) = \prod_{t=1}^N q(x_t|x_{t-1}) \quad (2)$$

where $\beta_1, \beta_2, \dots, \beta_N$ is a noise schedule that converts the data distribution x_0 into latent x_N . The choice of noise schedule has been shown to have important effects on sampling efficiency and quality [2, 8].

The **reverse process** is defined by a Markov chain parameterized by θ that iteratively refines latent point $x_N \sim \mathcal{N}(0, I)$ into data point x_0 . The learned transition probabilities are defined as,

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)) \quad (3)$$

$$p_\theta(x_{0:N}) = p(x_N) \prod_{t=1}^N p_\theta(x_{t-1}|x_t) \quad (4)$$

where the objective is to gradually denoise samples at each reverse diffusion step t . In practice, σ_θ is set to an untrained time-dependent constant based on the noise schedule, and [2] found $\sigma_\theta(x_t, t) = \sigma_t = \frac{1 - \bar{\alpha}_t - 1}{1 - \bar{\alpha}_t} \beta_t$ to have reasonable practical results, where $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$.

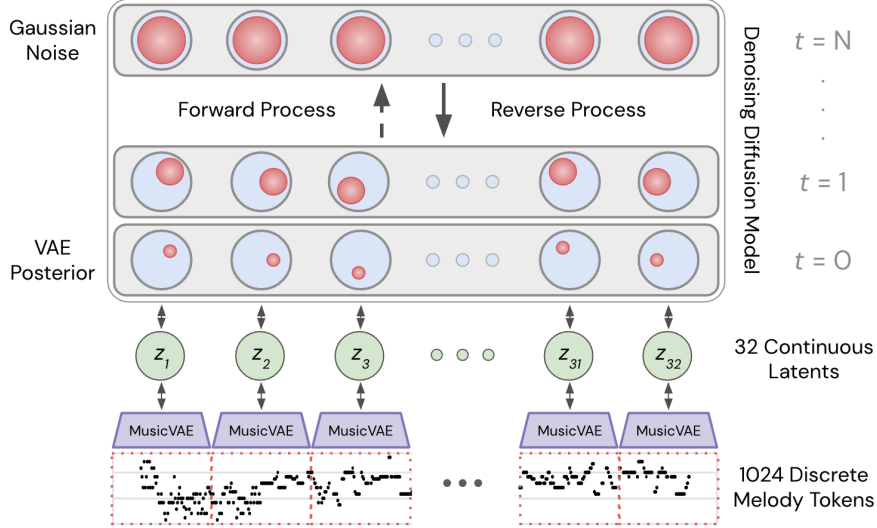


Figure 1. A diagram of our proposed framework. We use the pre-trained 2-bar melody MusicVAE [13] to embed discrete musical phrases (64 bars, 1024 tokens) into a sequence of continuous latent codes (32 latents, 512 dimensions each). These embeddings are used to train a diffusion model that iteratively adds noise such that after N diffusion steps the input embeddings are distributed $\mathcal{N}(0, I)$. To sample from this model, we initialize Gaussian noise and use the reverse process to iteratively refine the noise samples into a sequence of embeddings from the data distribution. These generated embeddings are fed through the MusicVAE decoder to produce the final MIDI sequence.

The training objective is to maximize the log likelihood of $p_\theta(x_0) = \int p_\theta(x_0, \dots, x_N) dx_{1:N}$, but the intractability of this marginalization leads to the following evidence lower bound (ELBO):

$$\begin{aligned} \mathbb{E} [\log p_\theta(x_0)] &\geq \mathbb{E}_q \left[\log \frac{p_\theta(x_{0:N})}{q(x_{1:N}|x_0)} \right] \\ &= \mathbb{E}_q \left[\log p(x_N) + \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] \end{aligned} \quad (5)$$

Additionally, [2] observed that the forward process can be computed for any step t such that $q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I)$, which can be viewed as a stochastic encoder. To simplify the above variational bound, [2] propose training on pairs of (x_t, x_0) to learn to parameterize this process with a simple squared L2 loss. The following objective is simpler to train, resembles denoising score matching [5, 14] and was found to yield higher-quality samples:

$$L(\theta) = \mathbb{E}_{x_0, \epsilon, t} \left[\left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t) \right\|^2 \right] \quad (6)$$

where t is sampled uniformly between 1 and N , $\epsilon \sim \mathcal{N}(0, I)$, and ϵ_θ is the learned diffusion model. [8] found that instead of conditioning on a discrete diffusion step t , it was beneficial to sample a continuous noise level $\sqrt{\bar{\alpha}} \sim \mathcal{U}(\sqrt{\bar{\alpha}_{t-1}}, \sqrt{\bar{\alpha}_t})$ where $t \sim \mathcal{U}(\{1, \dots, N\})$ and $\bar{\alpha}_0 = 1$.

We refer readers to Algorithms 2 and 3 based on [8] in our supplementary material¹ for the full training and sampling procedure.

¹Supplementary material including additional implementation details, audio, and tables is available at <https://goo.gl/magenta/symbolic-music-diffusion-examples>

2.2 Variational Autoencoders

Variational autoencoders [15] are generative models that define $p(y, z) = p(y|z)p(z)$ where z is a learned latent code for data point y . Additionally, the latent code is constrained such that z is distributed according to prior $p(z)$ where the prior is usually an isotropic Gaussian. The VAE is comprised of an encoder $q_\gamma(z|y)$ which models the approximate posterior $p(z|y)$ and a decoder $p_\theta(y|z)$ which models the conditional distribution of data y given latent code z .

The training objective is to maximize the log likelihood of $p_\theta(x) = \int p_\theta(y|z)p(z)dz$, but this marginalization is intractable, and we use the following variational bound maximized with $q_\gamma(z|y)$ as the approximate posterior:

$$\mathbb{E} [\log p_\theta(y|z)] - \text{KL}(q_\gamma(z|y)||p(z)) \leq \log p(y) \quad (7)$$

The flexible implementation of variational autoencoders allows them to learn representations over a wide variety of domains. Of particular interest to us are sequential autoencoders [13, 16] which use long short-term memory cells [17] to model temporal context in sequential data distributions.

In practice, there is a trade-off between the quality of reconstructions and the distance between the approximate posterior $q_\gamma(z|y)$ and the Gaussian prior $p(z)$. This makes sampling more difficult for VAEs with better reconstructions due to latent “holes” in the approximate posterior and is one of the primary shortcomings of these models.

3. MODEL

A diagram and description of our multi-stage diffusion model is shown in Figure 1. We refer readers to the sup-

plement for full implementation details.

3.1 Architecture

Our model learns to generate discrete sequences of notes (known as MIDI) by first training a VAE with parameters γ on the sequences and then training a diffusion model to capture the temporal relationships among the k VAE latents. Sequence VAEs such as MusicVAE are difficult to train on long sequences [13], which we overcome by pairing the short 2-bar MusicVAE model with a diffusion model capable of modeling dependencies between $k = 32$ latents, thus modeling 64 bars in total.

MusicVAE embeddings: Each musical phrase is a sequence of one-hot vectors with 16 quantized steps per measure and the vocabulary contains 90 possible tokens (1 note on + 1 note off + 88 pitches). We then parameterize each 2-bar phrase using the pre-trained 2-bar melody MusicVAE [13] and generate a sequence of continuous latent embeddings z_1, \dots, z_k to parameterize an entire sequence. The MusicVAE model employs bidirectional recurrent neural networks as an encoder and autoregressive decoding as shown in Figure 4 in the supplement. As we use the pretrained model from the original work, full model details can be found in [13]. After encoding each 2-measure phrase into a latent z embedding, we perform linear feature scaling such that the domain of each embedding is $[-1, 1]$. This ensures consistently scaled inputs starting from the isotropic Gaussian latent x_N for the diffusion model.

Transformer diffusion model: Our network $\epsilon_\theta(x_t, \sqrt{\alpha}) : \mathbb{R}^{k \times 42} \times \mathbb{R} \rightarrow \mathbb{R}^{k \times 42}$ is a transformer [18] where $k = 32$ is the length of each sequence of 42-dimensional preprocessed latent embeddings. The unperturbed data distribution used to train the diffusion model is $x_0 = [z_1, \dots, z_k]$. The network contains an initial fully-connected layer that projects the embeddings into a 128-dimensional space, followed by $L = 6$ encoder layers each with $H = 8$ self-attention heads and a residual fully-connected layer. All self-attention and fully-connected layers use layer normalization [19]. The output of the encoder is fed to $K = 2$ noise-conditioned residual fully-connected layers which generate the reverse process output. Each fully-connected layer contains 2048 neurons. We use a 128-dimensional sinusoidal positional encoding similar to [18] where j is the position index of a latent input embedding:

$$\omega = \left[10^{-\frac{4 \times 0}{63}} j, \dots, 10^{-\frac{4 \times 63}{63}} j \right] \quad e_j = [\sin(\omega), \cos(\omega)] \quad (8)$$

This positional encoding e_1, e_2, \dots, e_k is added to inputs x_t before being fed through the transformer encoder layers allowing the model to capture the temporal context of the continuous inputs.

Noise schedule and conditioning: As described in both the original diffusion model framework [2] and in [8], we use an additional sinusoidal encoding to condition the diffusion model on a continuous noise level during training and sampling. This noise encoding is identical to the positional encoding described above but with the frequency of

each sinusoid scaled by 5000 to account for the updated domain. We use feature-wise linear modulation [20] to generate γ (scale) and ξ (shift) parameters given a noise encoding and apply the transformation $\gamma\phi + \xi$ to the output ϕ of each layer normalization block in each residual layer, allowing for effective conditioning of the diffusion model. Our model uses a linear noise schedule with $N = 1000$ steps and $\beta_1 = 10^{-6}$ and $\beta_N = 0.01$.

3.2 Unconditional Generation

In the unconditional generation task, the goal is to produce samples that exhibit long-term structure. Because of our multi-stage approach, this works even in the scenario where the KL divergence between the marginal posterior $q_\gamma(z)$ and the Gaussian prior is quite large because the diffusion model accurately captures the structure of the latent space therefore improving the sample quality. Additionally, we extend the underlying VAE to samples longer than what it was trained to model by using the diffusion model to predict sequences of latent embeddings and attempt to generate unconditional samples with coherent patterns across a large number of measures.

3.3 Infilling

One of the benefits of using a sampling process that iteratively refines noise into data samples is that the trajectory of the reverse process can be steered and arbitrarily conditioned without the need for retraining the diffusion model. In creative domains, this post-hoc conditioning is especially useful for artists without the computational resources to modify or re-train deep models for new tasks. We demonstrate the power of diffusion modeling applied to music with conditional infilling of latent embeddings using an unconditionally trained diffusion model.

The infilling procedure extends the sampling procedure described in Algorithm 3 by incorporating information from a partially occluded sample s . At each step of sampling, we diffuse the fixed regions of s with the forward process $q(s_t|s) = \mathcal{N}(s_t; \sqrt{\alpha_t}s, (1 - \alpha_t)I)$ and use a mask m to add the diffused fixed regions to the updated sample x_{t-1} . The final output x_0 will be a version of s with the occluded regions inpainted by the reverse process.

We refer readers to Algorithm 1 for the modified sampling procedure that allows for post-hoc conditional infilling.

Algorithm 1 Infilling

Input: mask m , sample s , N steps, β_1, \dots, β_N

$x_N \sim \mathcal{N}(0, I)$

for $t = N, \dots, 1$ **do**

$\epsilon_1, \epsilon_2 \sim \mathcal{N}(0, I)$ if $t > 1$, else $\epsilon_1 = \epsilon_2 = 0$

$y = \sqrt{\alpha_t}s + \sqrt{1 - \alpha_t}\epsilon_1$ if $t > 1$, else s

$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \alpha_t}} \epsilon_\theta(x_t, \sqrt{\alpha_t}) \right) + \sigma_t \epsilon_2$

$x_{t-1} = x_{t-1} \odot (1 - m) + y \odot m$

end for

return x_0

4. METHODS

4.1 Data

We use the Lakh MIDI Dataset (LMD) [21] for all experiments. The dataset contains over 170,000 MIDI files with 99% of those files used for training and the remaining used for validation. We extracted 988,893 64-bar monophonic sequences for training and 11,295 for validation from the provided MIDI files. Each sequence was encoded into 32 continuous latent embeddings using MusicVAE. We set the softmax temperature for MusicVAE to 0.001 for decoding generated embeddings in all experiments. A diagram of the MusicVAE architecture used is shown in the supplementary material.

4.2 Autoregressive Baseline

We compare our model to an autoregressive transformer with a mixture density output layer [22] and train on the same dataset as the diffusion model. To ensure a fair comparison, we use the same architecture as our diffusion model with $L = 6$, $H = 8$, and $K = 2$ with 2048 neurons for each fully-connected layer. The mixture density layer outputs a mixture of 100 Gaussians to ensure sufficient mode coverage. In total, our baseline model has 38M trainable parameters. While the model is the same as the diffusion model (25.58M trainable parameters) up until the output layer, the autoregressive model has more parameters due to the much larger output layer. We refer to this model as TransformerMDN.

4.3 Training

All models were trained using Adam [23] with default parameters. We trained our diffusion model for 500K steps on a single NVIDIA Tesla V100 GPU for 6.5 hours using a learning rate of 10^{-3} annealed with a decay rate of 0.98 every 4000 steps and batch size 64. Unlike the diffusion model, which is non-autoregressive, we train TransformerMDN with teacher forcing. We use a batch size of 128, learning rate 3×10^{-4} , and train for 250K steps on a single NVIDIA Tesla V100 GPU for 6.5 hours.

We used the open-source implementation of MusicVAE written in TensorFlow [24] and the publicly available 2-bar melody checkpoints trained on LMD. We trained our diffusion and baseline models² with JAX [25] and Flax [26].

4.4 Framewise Self-similarity Metric

To evaluate the statistical similarity between our model’s qualitative output and the original training sequences, we present a metric that captures local self-similarity patterns across generated melodic sequences. Inspired by the statistical similarity evaluation described in [27], we evaluate our models with a modified framewise Overlapping Area (OA) metric.

We use a sliding 4-measure window with 2-measure hop size to capture local pitch and duration statistics across

the piece. Within each 4-measure frame, we compute the mean and variance of both pitch, which captures melodic similarity, and duration, which captures rhythmic similarity. These statistics specify a Gaussian PDF for pitch and duration for each frame ($p_P(k), p_D(k)$). We compute the Overlapping Area (OA) [27] of adjacent frames ($k, k + 1$) where each frame’s statistics are modeled as $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$, respectively:

$$\text{OA}(k, k + 1) = 1 - \text{erf}\left(\frac{c - \mu_1}{\sqrt{2}\sigma_1}\right) + \text{erf}\left(\frac{c - \mu_2}{\sqrt{2}\sigma_2}\right) \quad (9)$$

for both pitch (OA_P) and duration (OA_D), where erf is the Gauss error function and c is the point of intersection between Gaussian PDFs with $\mu_1 < \mu_2$. For a set of MIDI samples, we infer the *Consistency* and *Variance* from the mean (μ_{OA}) and variance (σ_{OA}^2) respectively of OA aggregated over all adjacent frames. We then use these aggregate values to compute the normalized relative similarity of pitch and duration consistency and variance to the training set (GT):

$$\begin{aligned} \text{Consistency} &= \max\left(0, 1 - \frac{|\mu_{\text{OA}} - \mu_{\text{GT}}|}{\mu_{\text{GT}}}\right) \\ \text{Variance} &= \max\left(0, 1 - \frac{|\sigma_{\text{OA}}^2 - \sigma_{\text{GT}}^2|}{\sigma_{\text{GT}}^2}\right) \end{aligned} \quad (10)$$

We clip *Consistency* and *Variance* such that samples with μ_{OA} or σ_{OA}^2 with greater than 100% percent error from the ground truth are considered to have zero relative similarity.

4.5 Latent Space Evaluation

We evaluate the similarity of latent embeddings generated by each of our models using the Fréchet distance (FD) [28] and Maximum Mean Discrepancy (MMD) [29] with a polynomial kernel, which are popular evaluation metrics in the generative modeling literature. These metrics measure the distance between the models’ continuous output distributions and the original data distribution in latent space. It is important to note that this metric does not measure long-term temporal consistency or quality of produced sequences and only measures the quality of the intermediate continuous representation before the final sequence is generated using the MusicVAE decoder.

5. RESULTS

5.1 Unconditional Generation

To evaluate unconditional sample quality, we compare batches of 1000 samples (32 latents each) generated by our proposed diffusion model with random draws from the training and test sets. We compare against a set of baseline generators including TransformerMDN, the independent $\mathcal{N}(0, I)$ MusicVAE prior, and spherical interpolation [30] between two MusicVAE embeddings at the start and end of an example from the test set.

As seen in Table 1, the diffusion model quantitatively produces samples most similar to the training data according to the relative framewise overlapping area metrics for

²Our implementation is available at <https://github.com/magenta/symbolic-music-diffusion>

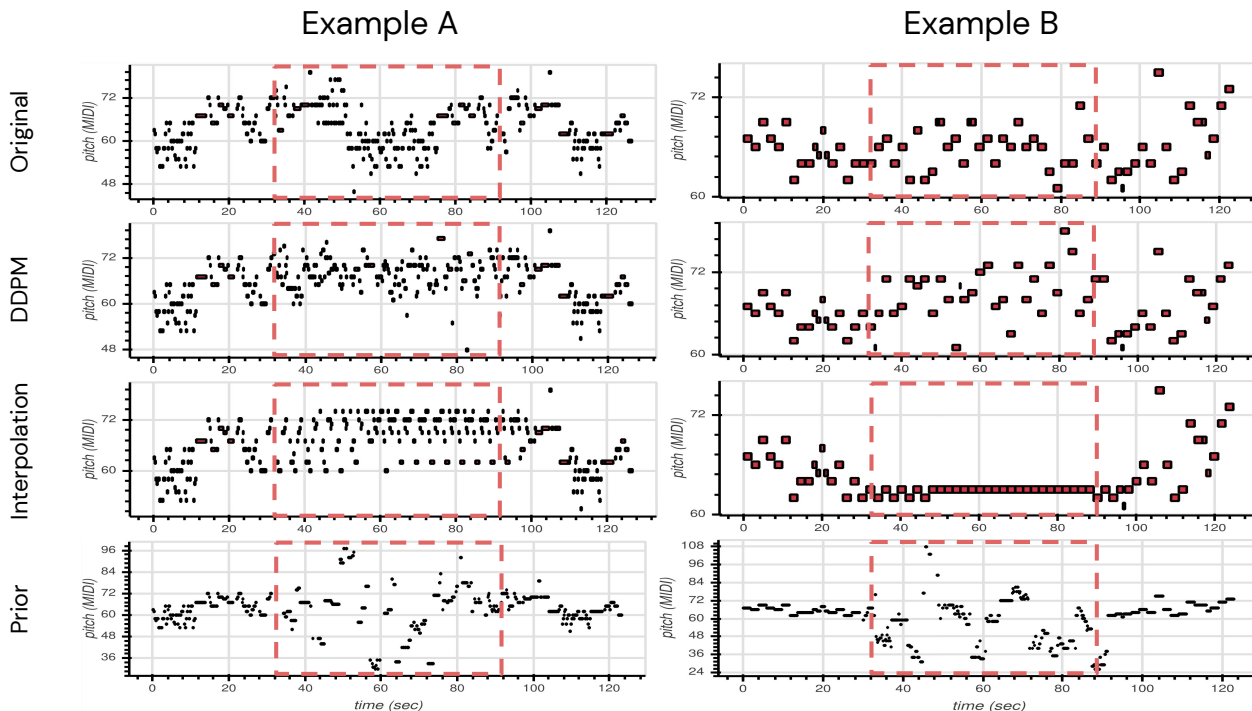


Figure 2. Example piano rolls of infilling experiments. For each example (A, B) the first and last 256 melody tokens are held constant, and the interior 512 tokens are filled in by the model (dashed red box). Even qualitatively, it is visually apparent that the diffusion model (second row) produces notes with a consistency and variance similar to the original data (first row), while the latent interpolation (third row) is too repetitive, and sampling independently from the prior (last row) produces outputs with too much variety and lack of local coherence.

Setting	Unconditional				Infilling			
	Pitch		Duration		Pitch		Duration	
Quantity	C	Var	C	Var	C	Var	C	Var
Metric								
Train Data	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Test Data	1.00	0.96	1.00	0.91	1.00	0.96	1.00	0.91
Diffusion	0.99	0.90	0.96	0.92	0.97	0.87	0.97	0.80
Autoregression	0.93	0.68	0.93	0.76	-	-	-	-
Interpolation	0.85	0.23	0.91	0.34	0.94	0.78	0.96	0.80
$\mathcal{N}(0, I)$ Prior	0.84	0.19	0.90	0.67	0.89	0.19	0.94	0.54

Table 1. Framewise self-similarity of consistency (C) and variance (Var), as defined in Equation 10, for note pitch and duration. For both unconditional sampling and infilling tasks, the diffusion model produces samples most similar to the real data. For diffusion samples, we use $N = 1000$ sampling steps with $\beta_1 = 10^{-6}$ and $\beta_N = 0.01$. For the TransformerMDN baseline we sample with a temperature of 1.0, meaning we sampled directly from the logits of mixture density layer. Absolute values of overlap area can be found in Table 2 in our supplementary material.

note pitch and duration. The diffusion model outperforms TransformerMDN, which is challenged by modelling the relatively high-dimensional continuous latents autoregressively, even with a mixture of Gaussians output. The autoregressive models are also trained with teacher forcing that results in exposure bias, leading to divergence from the data distribution during sampling. This is reflected in the lower pitch and duration consistency and higher variance

in the absolute overlapping area numbers seen in Supplemental Table 2. Additionally, the diffusion model is able to capture the joint dependencies of the sequences better because it learns to model all latents simultaneously as opposed to autoregressively. Note that the Gaussian prior also suffers from low consistency and high variance, due to lack of temporal dependencies, while the interpolated samples conversely suffer from low variance and too much consistency due to high repetition.

Table 3 in our supplement presents latent space evaluations of our generated samples. The TransformerMDN outperformed all other models, likely due to the Gaussian mixture prior on its output layer whereas the diffusion model must learn the output distribution from scratch. Furthermore, the latent space metrics are limited by assumptions about the latent manifold distribution and are unable to fully capture the detail of the entire space, further highlighting the necessity of our quantitative framewise self-similarity metrics and qualitative evaluations.

Figure 3 helps us to further understand the iterative refinement process by showing the improvement in sample quality as a function of iterative refinement time for both latent space and framewise self-similarity metrics. Interestingly, latent metrics improve steadily, while consistency similarity starts fairly high, and variance similarity only emerges at the end of refinement. We refer the reader to the supplementary material for extended visual and audio

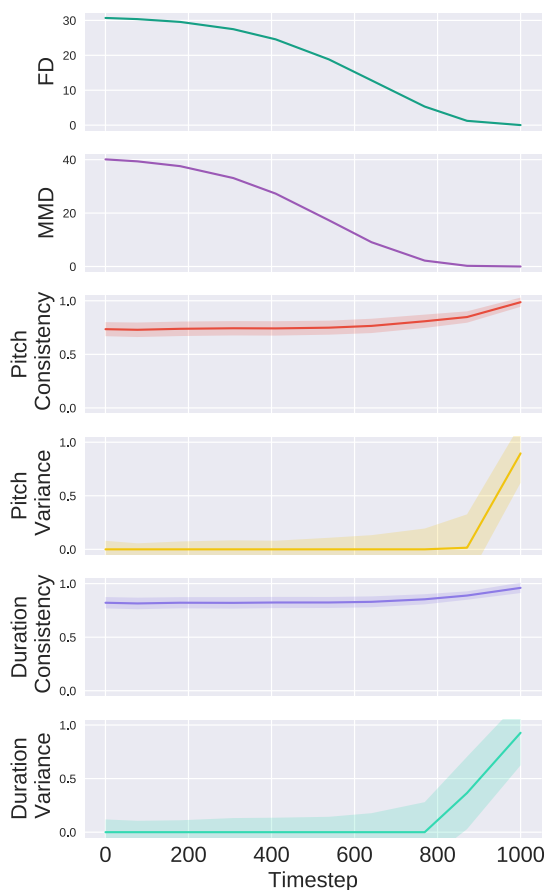


Figure 3. Sample quality improvement during iterative refinement. Latent space and framewise metrics evaluated at different stages of unconditional sampling. The metrics improve as the iterative refinement process progresses. We plot the means and standard deviations for 1000 samples.

samples of the generated sequences from each model³.

5.2 Infilling

To probe the diffusion model’s ability to perform post-hoc conditional generation, we remove the middle 32 measures (16 embeddings) and generate new embeddings following Algorithm 1 by conditioning on the first and last 8 embeddings. As in the unconditional setting, we compare to interpolation and independent samples from the prior.

Figure 2 visualizes the task by plotting the resulting note sequences for two different examples. Even qualitatively, it is visually apparent that the diffusion model produces notes with a consistency and variance similar to the original data. Latent interpolation is very consistent but unrealistically repetitive, and sampling independently from the prior produces a sequence with extremely large variance that is inconsistent in both pitch and duration.

The quantitative evaluations in Table 1 back up these observations. Similar to the unconditional generation task, the diffusion model outperforms the baselines in both consistency and variance similarity. We do not include the au-

to-regressive baseline because it is unable to condition on the final 8 embeddings.

6. RELATED WORK

Multi-stage learning: Several models have achieved long-term structure by first modeling some intermediate representation and then using that to guide the final generative process. Wave2Midi2Wave [31] uses a transformer to generate MIDI-like symbolic data and then a WaveNet [32] to synthesize that symbolic data into audio. Jukebox [11] and DALL-E [12] use similar approaches for text-conditioned generative music audio and image models. There has also been work investigating the extension of a single-measure VAE to multiple measures by using an autoregressive LSTM with a mixture density output layer [33], similar to TransformerMDN.

Iterative refinement: [34] use an orderless NADE with blocked Gibbs sampling to iteratively rewrite musical harmonies based on surrounding context. [35] use a gradient-based sampler combined with a restricted Boltzmann machine to generate polyphonic piano music. Similarly, [36] investigated the use of a score-based generative model [5] to generate Bach chorales with annealed Langevin dynamics. A key distinction between our method and prior work is the use of a VAE to parameterize the discrete space of musical notes for improved generation with a DDPM while previous methods have performed iterative refinement in the discrete space directly.

Conditional sampling from unconditional models: We build on top of the breadth of material that investigate steering generation in the latent space of an unconditionally trained generative model [37–40]. Most similar to our work is [4, 37, 41] which train an additional model on top of a pre-trained variational autoencoder to steer generation in the latent space of that autoencoder. Our approach builds on top of this by not only improving sampling and generation of the underlying autoencoder but also extending generation to sequences much longer than those used to train the VAE. We also use a diffusion model to refine generation and provide conditional infilling while the works mentioned primarily used conditional GANs and VAEs to extend the underlying autoencoder for a single latent embedding as opposed to a sequence of latent embeddings.

7. CONCLUSION

We have proposed and demonstrated a multi-stage generative model comprised of a low-level variational autoencoder with continuous latents modeled by a higher-level diffusion model. This approach enables using diffusion models on discrete data, and as priors for modeling long-term structure in multi-stage systems. We demonstrate that this model is useful for symbolic music generation, both in unconditional generation and conditional infilling. Future work will include extending this approach to other discrete data such as text, and exploring a greater array of approaches for post-hoc conditioning in creative applications.

³ <https://goo.gl/magenta/symbolic-music-diffusion-examples>

8. ACKNOWLEDGEMENTS

We thank Anna Huang, Carrie Cai, Sander Dieleman, Doug Eck and the rest of the Magenta team for helpful discussions, feedback, and encouragement.

9. REFERENCES

- [1] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2256–2265. [Online]. Available: <http://proceedings.mlr.press/v37/sohl-dickstein15.html>
- [2] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: <https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf>
- [3] M. Welling and Y. W. Teh, “Bayesian learning via stochastic gradient langevin dynamics,” in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML’11. Madison, WI, USA: Omnipress, 2011, p. 681–688.
- [4] J. Engel, M. Hoffman, and A. Roberts, “Latent constraints: Learning to generate conditionally from unconditional generative models,” in *International Conference on Learning Representations*, 2018. [Online]. Available: <https://openreview.net/forum?id=Sy8XvGb0->
- [5] Y. Song and S. Ermon, “Generative modeling by estimating gradients of the data distribution,” in *Advances in Neural Information Processing Systems*, 2019, pp. 11 895–11 907.
- [6] —, “Improved techniques for training score-based generative models,” in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020.
- [7] Y. Du and I. Mordatch, “Implicit generation and modeling with energy based models,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf>
- [8] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, “Wavegrad: Estimating gradients for waveform generation,” in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=NsMLjcFaO8O>
- [9] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, “Diffwave: A versatile diffusion model for audio synthesis,” in *International Conference on Learning Representations*, 2021. [Online]. Available: <https://openreview.net/forum?id=a-xFK8Ymz5J>
- [10] A. Razavi, A. van den Oord, and O. Vinyals, “Generating diverse high-fidelity images with vq-vae-2,” in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: <https://proceedings.neurips.cc/paper/2019/file/5f8e2fa1718d1bbcadf1cd9c7a54fb8c-Paper.pdf>
- [11] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *arXiv preprint arXiv:2005.00341*, 2020.
- [12] A. Ramesh, M. Pavlov, G. Goh, S. Gray, M. Chen, R. Child, V. Misra, P. Mishkin, G. Krueger, S. Agarwal, and I. Sutskever, “Dall-e: Creating images from text,” *OpenAI blog*, 2021. [Online]. Available: <https://openai.com/blog/dall-e>
- [13] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, “A hierarchical latent vector model for learning long-term structure in music,” in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4364–4373. [Online]. Available: <http://proceedings.mlr.press/v80/roberts18a.html>
- [14] P. Vincent, “A connection between score matching and denoising autoencoders,” *Neural computation*, vol. 23, no. 7, pp. 1661–1674, 2011.
- [15] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Second International Conference on Learning Representations*, 2014.
- [16] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” 2016.
- [17] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.
- [19] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

- [20] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [21] C. Raffel, “Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching,” Ph.D. dissertation, Columbia University, 2016.
- [22] C. M. Bishop, “Mixture density networks,” 1994.
- [23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.
- [25] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “JAX: composable transformations of Python+NumPy programs,” 2018. [Online]. Available: <http://github.com/google/jax>
- [26] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, “Flax: A neural network library and ecosystem for JAX,” 2020. [Online]. Available: <http://github.com/google/flax>
- [27] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, “Encoding musical style with transformer autoencoders,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 1899–1908.
- [28] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “Gans trained by a two time-scale update rule converge to a local nash equilibrium,” 2018.
- [29] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012. [Online]. Available: <http://jmlr.org/papers/v13/gretton12a.html>
- [30] T. White, “Sampling generative networks,” *arXiv preprint arXiv:1609.04468*, 2016.
- [31] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAE-STRO dataset,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=r11YRjC9F7>
- [32] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.
- [33] H. Jhamtani and T. Berg-Kirkpatrick, “Modeling self-repetition in music generation using generative adversarial networks,” 2019.
- [34] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by convolution,” in *Proceedings of ISMIR 2017*, 2017. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/187_Paper.pdf
- [35] S. Lattner, M. Grachten, and G. Widmer, “Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints,” *arXiv preprint arXiv:1612.04742*, 2016.
- [36] E. Zhang and R. Sirohi, “Generative modeling of bach chorales by gradient estimation,” 2020. [Online]. Available: https://www.ekzhang.com/assets/pdf/Generative_Music_Modeling.pdf
- [37] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, “Plug & play generative networks: Conditional iterative generation of images in latent space,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4467–4477.
- [38] N. Jaques, J. Engel, D. Ha, F. Bertsch, R. Picard, and D. Eck, “Learning via social awareness: improving sketch representations with facial feedback,” in *International Conference on Learning Representations*, 2018, workshop Track. [Online]. Available: <https://arxiv.org/abs/1802.04877>
- [39] A. Jahanian, L. Chai, and P. Isola, “On the “steerability” of generative adversarial networks,” *arXiv preprint arXiv:1907.07171*, 2019.
- [40] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, “Plug and play language models: A simple approach to controlled text generation,” *arXiv preprint arXiv:1912.02164*, 2019.
- [41] M. Dinculescu, J. Engel, and A. Roberts, Eds., *MidiMe: Personalizing a MusicVAE model with user data*, 2019.
- [42] M. D. Hoffman and M. J. Johnson, “Elbo surgery: yet another way to carve up the variational evidence lower bound.”
- [43] A. Alemi, B. Poole, I. Fischer, J. Dillon, R. A. Saurous, and K. Murphy, “Fixing a broken elbo,” in *International Conference on Machine Learning*. PMLR, 2018, pp. 159–168.

LEARNING FROM MUSICAL FEEDBACK WITH SONIC THE HEDGEHOG

Faraaz Nadeem

Massachusetts Institute of Technology

faraaz@mit.edu

ABSTRACT

Most videogame reinforcement learning (RL) research only deals with the video component of games, even though humans typically play while experiencing both audio and video. In this paper, we aim to bridge this gap in research, and present two main contributions. First, we provide methods for extracting, processing, visualizing, and hearing gameplay audio alongside video. Then, we show that in *Sonic The Hedgehog*, agents provided with both audio and video can outperform agents with access to only video by 6.6% on a joint training task, and 20.4% on a zero-shot transfer task. We conclude that game audio informs useful decision making, and that audio features are more easily transferable to unseen test levels than video features.

1. INTRODUCTION

Although classification and decision making are well studied areas of machine learning (ML), most research and practical application of these areas has involved the use of video or text based data, as opposed to audio. Of the audio research that exists, only an even smaller subset has analyzed environmental audio or music as a method of providing feedback.

This gap in research needs to be addressed since environmental and feedback-related audio is critical to achieving human performance on a number of tasks, especially when visual or textual clues are not sufficient. Self driving cars cannot achieve full automation without being able to react and respond to emergency vehicle sirens in the distance, or nearby car horns. Emergency responders listen for fire alarms and calls for help to locate the emergency when arriving at a scene. The clicking feedback produced by a mouse, keyboard, or button, confirms to the user that the hardware or software has received the input. [1].

In this work, we use *Sonic The Hedgehog* games for the SEGA Genesis as an environment to understand how RL agents can incorporate both audio and video observations, to make decisions with immediate consequence, feedback,

and long term goal.¹

2. BACKGROUND

2.1 Sonic The Hedgehog

Sonic The Hedgehog is a 2D side scrolling platforming game. The main idea of the game is to navigate Sonic through vertical loops, over bottomless pits, off of springs, while collecting rings. A central game mechanic is that Sonic can build great speed if his movement is not interrupted by stopping or bumping into obstacles, and this can then be used to launch him into the air off of ramps, or through an array of enemies. Each game consists of individually themed Zones, and each Zone has 1 to 3 Acts. Going forward, we will refer to each Act as a level.

Rings are placed in groups of 3 or more along the levels. They give points, and act like a shield. If Sonic is hit by an enemy or hazard without any rings, he loses a life. But if Sonic has at least 1 ring, then hitting an enemy or hazard knocks Sonic backwards and he forfeits up to 20 of his rings, without losing a life.

Many levels contain sections which are underwater. Sonic can survive approximately 30 seconds underwater until he needs to find an air bubble or jump out of the water to avoid drowning.

2.2 Related Work

Gotta Learn Fast [2] is a transfer learning benchmark with Gym Retro [3] on the Sonic games for the SEGA Genesis. It shows that pretraining on 47 train levels before fine tuning on 11 test levels achieves the best test result when compared to several other models. They do not perform any audio-related experiments.

Kim et al. [4] modify the Atari Learning Environment [5] to support audio queries, and demonstrate that latent audio/video features increase performance on H.E.R.O and Amidar on the Atari 2600, and transfer knowledge to accelerate learning in a door puzzle game.

Kaplan et al. [6] show that the additional modality of natural language suggestions can be used to improve performance on the Atari 2600 games. Ngiam et al. [7] present a series of tasks for multimodal learning and demonstrate cross modality feature learning, where better features for one modality (e.g., video) can be learned if mul-

¹ Link to Github repository with code and gameplay video examples provided here <https://github.com/faraazn/meng>



multiple modalities (e.g., audio and video) are present at feature learning time. Poria et al. [8] show that a multimodal system fusing audio, visual, and textual clues outperforms previous state of the art systems by 20% on a YouTube sentiment dataset.

Henkel et al. [9] show that RL can be useful in the music domain by applying it to the score following task. Various works have used neural network approaches to acoustic scene classification [10–12] and sound event detection [13–15], achieving state of the art results.

3. RELEVANT MUSIC THEORY

3.1 Western Associations in Music

In Western music theory and culture, there are clear associations of consonance, dissonance, and rhythmic patterns with certain emotions. The major mode is slightly more consonant, and therefore associated with "happy, merry, graceful, and playful", while the minor mode is slightly more dissonant, and associated with "sad, dreamy, and sentimental". Firm rhythms are perceived as "vigorous and dignified", while flowing rhythms are "happy, graceful, dreamy, and tender". Combining these ideas, complex dissonant harmonies are "exciting, agitating, vigorous, and inclined towards sadness", and simple consonant harmonies are "happy, graceful, serene, and lyrical" [16].

Studies have shown that these associations can largely be attributed to cultural conditioning, rather than universal human behavior. Residents of a village in the Amazon rainforest with no exposure to Western music showed no preference between consonant and dissonant sounds [17]. We can trace back Western preference for consonance at least to the early 18th century, when the name "diabolus in musica", or "the Devil in music" was attributed to the augmented fourth, the most dissonant interval [18].

3.2 Application to Videogame Sound Effects

Videogame music and sound effects are designed to leverage our implicit biases as a means of meeting expectation and lowering the barrier to entry for learning a new game. We can better understand this by applying it to the Sonic games.

It is good for Sonic to collect rings, since they give some protection before losing a life, award more points, and collecting enough of them result in an extra life. In line with our understanding of Western preference for simple consonance, the ring sound is a major triad, implying a sense of positive value for the act of acquiring a ring. It is comprised of the notes E5, G5, and C6, which form specifically a first inversion C major triad (Figure 1).

When Sonic loses rings after being hit by an enemy or hazard, this is bad because he is prone to losing a life with an additional hit. In line with our understanding of Western music theory, the corresponding sound is aggressively dissonant, implying a sense of negative value for the act of losing rings. This sound is created by rapidly alternating notes A6 and G6, which form a major second interval.

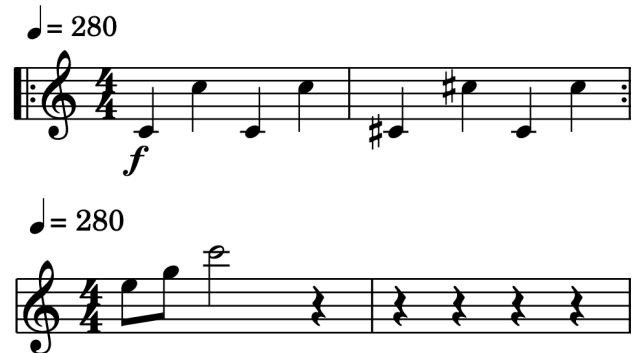


Figure 1. Drowning Theme and Ring Sound. Top: the drowning theme consists of dissonant jumps between octave intervals. Bottom: the three consonant notes of the ring acquiring sound.

When Sonic has spent too much time underwater without air, and is close to drowning, the drowning theme begins to play. It alternates between octave intervals on C, and a minor second interval jump to octave intervals on C# (Figure 1). The jumps lie on a dissonant interval, and the music gets increasingly loud and fast as Sonic gets closer to drowning, giving a strong sense of impending doom.

In general, consonance bias is an effective tool for conveying positive or negative value of player actions or setting emotional expectation with audio, without needing to spoon-feed explicit instructions or visual cues.

4. EXPERIMENTAL SETUP

4.1 Reinforcement Learning

Reinforcement learning [19, 20] is an increasingly popular field of machine learning research. It is a semi supervised approach to teaching an agent complex sequential decision making in an environment with potentially sparsely labeled data or delayed reward.

More concretely, we define an agent that operates under a policy π with parameters θ , and produces actions given an observation, $\pi_\theta : o \rightarrow A$. At time t , the agent takes an action $a \in A$ while in state $s \in S$, transitions to state $s' \sim \mathcal{T}(s, a, s')$, and receives observation $o' \sim \mathcal{O}(o', s')$ and reward $r = \mathcal{R}(s, a) \in R$.

In the Sonic games, we can imagine a scenario where Sonic is being approached by an enemy. The game state s tells us the magnitude and direction of the enemy's velocity. An observation o of this state might be a single frame of video which tells us the enemy is close to Sonic. When the player takes an action a to control Sonic, performing a spin attack may lead to a transition (s, a, s') to a new state s' where the enemy is defeated. This would result in a new visual observation o' of the defeated enemy, and a reward r of 100 points.

The goal of an RL agent is to maximize the total expected reward. To optimize our agent parameters θ and achieve the best performance, we use Proximal Policy Optimization (PPO) [21]. PPO is a policy gradient algorithm

that achieves state of the art results on a number of RL benchmarks. It builds on former advancements in policy gradient research [22–24], while also aiming to be fast and easy to implement. The details of its implementation are outside of the scope of this work; we used an open source implementation [25].

4.2 Task Outlines

To quantify the effect of audio on videogame RL, we set up two different tasks.

1. Joint PPO Training. We train a single agent on 47 of the 58 levels, which we call the training levels. The agent trains for 30 million time steps, and we self-evaluate the agent on the training levels every 5 million steps.
2. Zero-shot Transfer. We take the jointly trained agent from the previous experiment and evaluate it on the 11 test set levels, without any fine tuning. If the agent learned general techniques for level progression in task 1, it should be able to perform better than a random agent, or one following a simple policy. For each test level, there is at least one training level from the same Zone.

4.3 Open AI Gym Retro

We used the Open AI Gym Retro framework to set up the two experiments outlined above. Its API allows us to interface with games by inputting player actions, and receiving the updated game state, observations, and reward in return. There are thousands of games available to use with Gym Retro, including the original Sonic series for the SEGA Genesis.

However, the Gym Retro (version 0.8.0) package does not readily expose an API for interacting with the videogame emulator’s audio. To aid others facing this issue, we have open sourced our codebase, in which we were able to expose the audio features for training our RL models, and create various tools for visualization and playback. This includes tools for both online and offline playback, audio visualization, and neural network feature heat mapping for a given runthrough of a level.

4.4 Environment Setup

For all of our experiments, we use mostly the same base environment setup as described by Nichol et al. [2]. This includes usage of save states, episode boundaries, stochastic frame skips, action space reduction, and reward function. At a high level, this setup overrides the in-game point system to instead reward Sonic for making new progress to the right of the screen, and reduces the number of / speed with which buttons can be pressed, among other things. Due to computational constraints, we set the horizon of experience generation to 512 instead of 8192, number of workers per level to 3 instead of 4, and total train steps to 30 million instead of 400 million. The convolutional neural network (CNN) architecture we use to process audio and video observations is given by Figure 2.

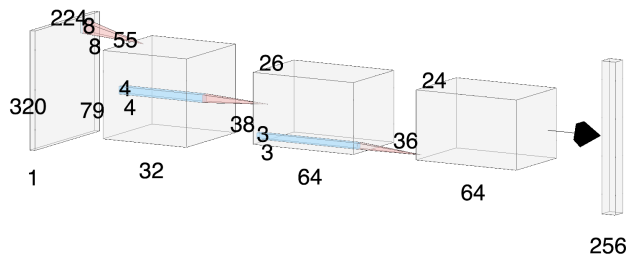


Figure 2. CNN architecture for the observation encoder. The input observation in this example is a grayscale frame of video, and the same structure is used to encode audio.

4.5 Agent Variants

We create 5 different agent variants to be able to differentiate the effects of incorporating audio.

Agent 1: conceptually the same as the one used in the Gotta Learn Fast [2] benchmark, albeit with fewer trainable parameters and shorter training time due to computational constraints.

Agent 2: we convert video inputs to grayscale and introduce a random start of 16–45 random actions (about 1-3 seconds) which we expect will result in improvements to generalization.

Agent 3: takes an additional video frame from the previous time step as input, and therefore also has double the model size as Agent 2. We hypothesize that Agent 3 may be able to learn temporal features, and therefore achieve better results than Agent 2.

Agent 4: only keeps the current grayscale video frame, but adds audio features. We construct these audio features by concatenating the past 16 frames, without frame skipping. Since the Gym Retro sample rate is 44100 Hz, this means we get 735 samples of audio per frame and 47040 samples per observation (about 1 second long). We process the 1D sequence of samples into a 2D mel spectrogram with 256 mels, window size 256, and hop length 128, for a resulting size of 367 by 256 pixels. We construct a new encoder for this audio with the same CNN architecture as video, so the overall model size is comparable to Agent 3.

Agent 5: the same as Agent 4 except we create a 2D mel spectrogram with 256 mels, window size 1024, and hop length 512, for a resulting size of 92 by 256 pixels. This spectrogram is 4 times smaller, so the overall model size is comparable to Agents 1 and 2.

To normalize the spectrogram values, we first convert from the power to decibel scale, setting the max decibel range to 80, and then dividing by 80 to put the range of values between 0 and 1. Separate CNNs are used to process the video and audio observations, each with hidden size 256, and concatenated together into a final hidden state size 512.

4.6 Baselines

4.6.1 "Hold Right" Baseline

By definition of the reward function defined for us [2], Sonic accumulates reward as he makes net progress to-

Agent	Video Frames	Rand. Start	RGB	Audio	Num Params
1	1	no	yes	no	14m
2	1	yes	no	no	14m
3	2	yes	no	no	28m
4	1	yes	no	yes	31m
5	1	yes	no	yes	18m

Table 1. Agent Variants.

wards the right. We can therefore define a simple yet effective policy that requires no machine learning: always make Sonic move right. We evaluate this policy and set it as a lower bound for the performance we expect our trained agent to achieve.

4.6.2 Human Baseline

There is an existing human baseline for the Sonic Genesis games. In this baseline, 4 test subjects practiced on the 47 Sonic training levels for 2 hours, before playing each of the 11 test levels over the course of an hour.

5. RESULTS

5.1 Overview

Table 2 shows us the most important result of this work, which is that audio+video agents outperform video-only agents on the joint training task, and achieve higher scores on the zero-shot transfer task. Agent 5, which is provided with the current frame of video and past 1 second of audio, outperforms Agent 3, which is provided with the current and previous frames of video, no audio, and 55% larger model size, by 6.6% on the joint task, and 20.4% on the zero-shot task. This result supports our hypothesis that Sonic game audio informs sequential decision making, and extracted audio features are more easily transferable to unseen test levels than video features.

The "Final Avg" section contains the primary scores used to evaluate overall agent performance. Final refers to the fact that these scores were computed by evaluating the final saved checkpoints of each agent, rather than averaging over the entire training run.

We add a "Best Ckpts" table, which aggregates the top per-level mean scores across all corresponding agent checkpoints. We saved checkpoints every 5 million train steps. The goal of adding this table is to demonstrate each agent’s best effort for each level.

Scores for each agent are presented as the mean \pm the first standard deviation taken along the *individual level averages*. The gameplay figures are overlaid with Score CAM heat maps [26, 27], and important parts of these figures are annotated with circles or arrows. Each agent was trained and evaluated with three different random seeds.

5.2 Video Agent Analysis

Agent 1 is only able to outperform the "hold right" policy by a small margin. We found that over the course of train-

Agent	Joint Final Avg	Zero-Shot Final Avg
1	1344.6 \pm 206.4	435.6 \pm 130.0
2	1479.5 \pm 911.3	578.5 \pm 409.7
3	1905.0 \pm 286.9	678.5 \pm 238.3
4	1893.7 \pm 225.3	817.3 \pm 320.2
5	2031.1 \pm 501.9	936.6 \pm 220.8

Agent	Joint Best Ckpts	Zero-Shot Best Ckpts
1	1780.2 \pm 1708.5	755.1 \pm 1028.1
2	2998.6 \pm 2010.63	1526.8 \pm 1257.4
3	3013.0 \pm 2186.2	1686.9 \pm 1155.1
4	3041.4 \pm 2227.9	1779.2 \pm 1403.1
5	2901.5 \pm 2011.1	1756.9 \pm 1298.3

Baseline	Joint Final Avg	Zero-Shot Final Avg
Hold Right	1099.1 \pm 1092.8	321.9 \pm 277.5
Human [2]	–	7438.2
N. et al. [2]	5083.6	\sim 1000

Table 2. Performance summary of all 5 agent variants and 3 baselines over both tasks. Note that N. et al. trained with significantly more computational power.

ing, it tended to overfit to specific levels every few million train steps.

Agent 2 uses grayscaleing instead of RGB, and adds a random start between 1 and 3 seconds to the Agent 1 setup. Adding the random start makes it much more difficult for Agent 2 to try to memorize a high reward path through each level. This substantially improves scores on the joint and zero-shot tasks, because increased starting state diversity forces Agent 2 to learn more general features for level progression. If the agent learns that jumping over spikes is good in one level (Figure 3), the the lower layers of the CNN that were used to recognize the spikes will help to transfer that knowledge to other levels.

Agent 3 builds upon Agent 2 by doubling the hidden layer size to 512 and increasing the number of video frame observations from 1 to 2. By including the video frame from the previous time step, Agent 3 now has the ability to learn temporal meta-variables, such as relative velocity of objects on screen. Agent 3 does actually learn to use this information to determine, for example, when Sonic does not have enough momentum to go up a steep ramp, and must backtrack in order to build the momentum needed to try again (Figure 4). It improves over the Agent 2 joint train and zero-shot scores by 29% and 17%, and also provides lower variance results.

5.3 Audio+Video Agent Analysis

Agents 4 and 5 replace the second video frame with a spectrogram representing the last 1 second of in-game audio.

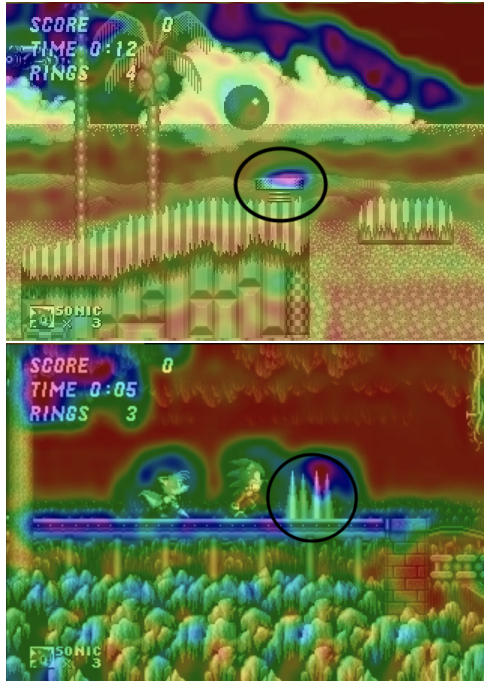


Figure 3. Agent 2 detects various objects during training. Top: the agent jumps on a spring to launch Sonic over the pit. Bottom: the agent recognizes that Sonic needs to jump over the spikes.

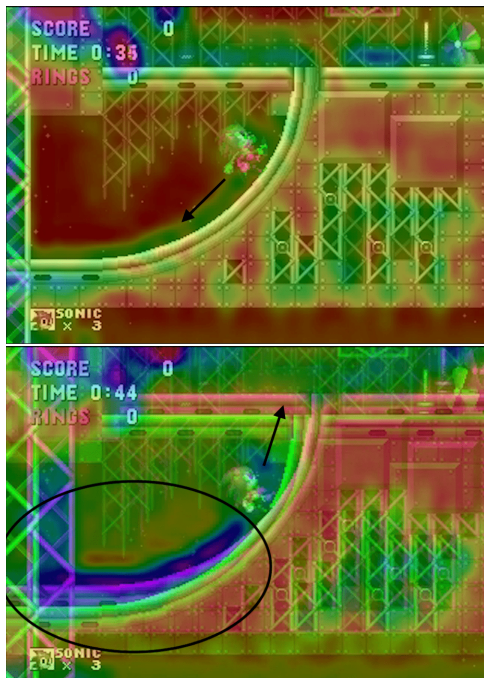


Figure 4. Agent 3 uses temporal information to judge its progression up the ramp. Top: the agent decides it does not have enough momentum, and begins to backtrack to the left. Bottom: the Score CAM weighting shows the agent understands that the screen shifting upwards means that Sonic has enough momentum to make it up the ramp.

5.3.1 Relative Receptive Field Comparison

Agent 4 constructs a spectrogram with window size 256 and hop length 128, while Agent 5 uses window size 1024 and hop length 512. This leads to spectrogram sizes 368 by 256 pixels and 92 by 256 pixels, respectively. Both convey the same amount of information, but result in audio CNNs with different numbers of trainable parameters (Agent 4 has about 14 million more than Agent 5) and different relative receptive field sizes (Agent 4 has a 4 times smaller relative receptive field). In other words, Agent 4 learns a larger number of smaller audio details, and Agent 5 learns a smaller number of larger ones.

In general, we found that Agent 5 was better able to learn high level audio features. Agent 4 was more prone to noise and overfitting because it could only recognize smaller parts of overall sound effects.

5.3.2 Important Audio Features

Rings are small, keep rotating, and turn into even smaller stars when grabbed by Sonic, so the act of acquiring them is not recognized by the agent’s visual CNN component. The corresponding sound is easy to see in a spectrogram, so it is readily learned by the audio CNN component. The same is true for when Sonic loses his rings.

An agent will respond to the ring acquiring sound by increasing its likelihood of taking the "move right" action, since rings are generally located along paths that lead to forward progress in the level. Figure 5 shows an example of Agent 5 doing this on a zero-shot run of Hill Top Zone Act 2. Ironically, losing rings also produces the effect of increased "move right" likelihood, despite having a dissonant sound and negative connotation for humans, because Sonic is granted temporary invincibility after losing his rings. The agent uses this as an opportunity to get past the danger that was originally in the way.

Visual indicators that Sonic is about to drown are small see-through bubbles with numbers that count down and do not appear every frame. These cues are easily missed by the visual CNN, and video-only agents do not act on them. However, our audio+video agents learn to pick up on the drowning theme and quickly locate an oxygen-restoring air bubble or exit the water entirely to avoid losing a life (Figure 6).

Before Sonic gets to the drowning stage, there are pings that play every 5 seconds Sonic is underwater. In some levels where prolonged underwater travel is not always required, such as Angel Island Zone Act 2, Agent 5 exits the water after hearing the first one of these pings.

Surprisingly, a significant number of sounds highlighted by the Score CAM algorithm come from the background musical score, which is not supposed to be feedback-related. We hypothesize that these learned features are mainly an artifact of the audio CNN incorrectly attributing accumulated reward to the part of the score that happened to be playing at the time. Typically these features do not seem to have a significant effect on agent actions, but they can be coincidentally reinforced when the same parts of the music repeat.

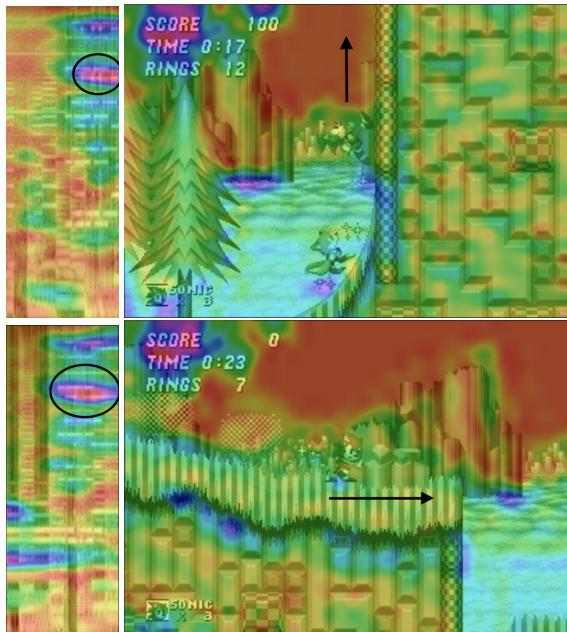


Figure 5. In Hill Top Zone Act 2, during a zero-shot run, Agent 5 looks at ring sound as motivation to move right. Note that the left column shows spectrogram heat maps from the last second of audio. Top: the agent acquires rings while holding right and continues to run up a steep cliff. Bottom: the agent continues to run into the rock blocking its path until the ring sound is freed from its memory.

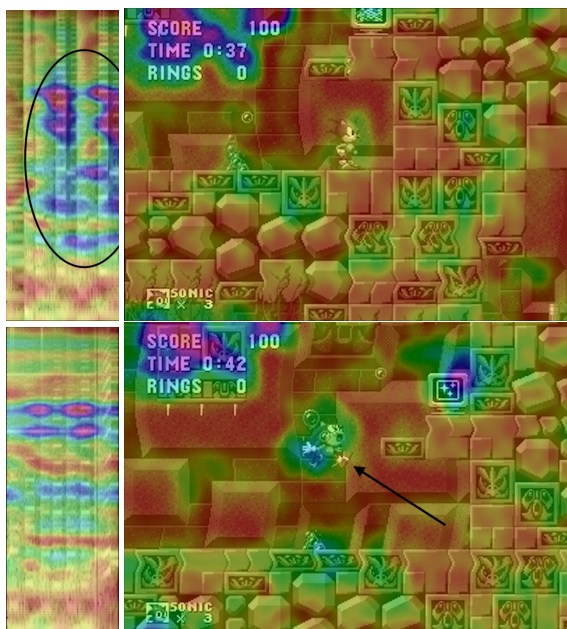


Figure 6. Agent 5 avoids drowning. Top: the agent hears the beginning of the drowning theme. Bottom: the agent navigates Sonic to the bubble stream on the left, and waits for a bubble to appear. The model highlights Sonic as he breathes in the air bubble, confirming the danger has passed.

5.3.3 Unimportant Audio Features

There are a few sound effects that we expected to be more important for our training tasks. We thought that the jump sound effect would be an important temporal indicator of Sonic’s jump trajectory, i.e. determining if he is rising or falling. We also expected the sound effect produced by defeating an enemy to be important for learning to defeat enemies before getting hit and losing a life.

We conclude that these features were not learned because in Sonic 2 and Sonic 3&K, a side character follows Sonic around and produces many of the same sound effects while autonomously interacting with his local environment. This means that the jump sound or defeating an enemy sound is often not attributed to Sonic, and therefore loses its predictive power.

Our agents also did not learn to differentiate between consonant and dissonant sounds, or along any other axis of Western classical music theory explained earlier. This was expected, since the sounds in the Sonic games are not plentiful or diverse enough to be able to learn beyond simple classification. These theories should play a greater role if/when a large unsupervised audio model is trained on massive amounts of Western audio data and fine tuned on videogame RL tasks, similar to how BERT [28] and GPT-3 [29] have transformed the field of NLP.

5.3.4 Learned Video Features

None of this detailed audio analysis is to say that our audio agents have weaker visual components. In fact, Agent 5 is capable of navigating multiple multi-step visual challenges with what appears to be limited overfitting. This suggests that the audio component can be purely supplemental to the video component, although as mentioned previously, the agent may confuse itself in compounding ways when it chooses to assign reward responsibility to the wrong component.

6. CONCLUSION

The diverse nature of our learned audio features supports some of our original hypotheses, namely that environmental audio can reinforce existing visual ideas, and call attention to cues that are missed by the visual component, or simply non visual. We can expect that the supplementary role of our learned audio features, combined with their natural transfer ability, would extend beyond our work on *Sonic The Hedgehog*.

As video games become increasingly realistic, we expect that agents with the ability to process both audio and video will be the ones to achieve state of the art results on RL benchmarks. For future work, it would be interesting to see how audio features transfer across different games (say, Sonic and Mario), and how state of the art Atari models such as Agent 57 [30] might perform when given access to audio and games with richer audio/video components.

Overall, we hope that this work provides an insightful application to multi-modal videogame reinforcement learning, and motivates further such research in this field.

7. ACKNOWLEDGEMENTS

I would like to express my gratitude for the contributions of Eran Egozy, Professor of the Practice in Music Technology at MIT. He was an invaluable advisor and mentor, and should be considered as a secondary author for this work.

8. REFERENCES

- [1] B. Langkjær, “Making fictions sound real-on film sound, perceptual realism and genre,” *MedieKultur: Journal of media and communication research*, vol. 26, no. 48, pp. 13–p, 2009.
- [2] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, “Gotta learn fast: A new benchmark for generalization in rl,” *arXiv preprint arXiv:1804.03720*, 2018.
- [3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [4] D.-K. Kim, S. Omidshafiei, J. Papis, and J. P. How, “Crossmodal attentive skill learner: learning in atari and beyond with audio–video inputs,” *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, p. 16, 2020.
- [5] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
- [6] R. Kaplan, C. Sauer, and A. Sosa, “Beating atari with natural language guided reinforcement learning,” *arXiv preprint arXiv:1704.05539*, 2017.
- [7] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, “Multimodal deep learning,” in *ICML*, 2011.
- [8] S. Poria, E. Cambria, N. Howard, G.-B. Huang, and A. Hussain, “Fusing audio, visual and textual clues for sentiment analysis from multimodal content,” *Neuro-computing*, vol. 174, pp. 50–59, 2016.
- [9] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, “Score following as a multi-modal reinforcement learning problem,” *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, 2019.
- [10] D. Battaglino, L. Lepauloux, N. Evans, F. Mougins, and F. Biot, “Acoustic scene classification using convolutional neural networks,” *IEEE AASP Challenge on Detec*, 2016.
- [11] S. H. Bae, I. Choi, and N. S. Kim, “Acoustic scene classification using parallel combination of lstm and cnn,” in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2016 Workshop (DCASE2016)*, 2016, pp. 11–15.
- [12] J. Salamon and J. P. Bello, “Deep convolutional neural networks and data augmentation for environmental sound classification,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.
- [13] E. Cakir, T. Heittola, H. Huttunen, and T. Virtanen, “Polyphonic sound event detection using multi label deep neural networks,” in *2015 international joint conference on neural networks (IJCNN)*. IEEE, 2015, pp. 1–7.
- [14] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 6440–6444.
- [15] E. Cakir, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional recurrent neural networks for polyphonic sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.
- [16] K. Hevner, “Experimental studies of the elements of expression in music,” *The American Journal of Psychology*, vol. 48, no. 2, pp. 246–268, 1936.
- [17] Sabbi Lall, “Universal musical harmony,” "<http://news.mit.edu/2020/universal-musical-harmony-0701>", 2020, [Online; accessed 14-August-2020].
- [18] W. Apel, *The Harvard dictionary of music*. Harvard University Press, 2003.
- [19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [20] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.
- [21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [22] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Advances in neural information processing systems*, 2000, pp. 1008–1014.
- [23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning*, 2016, pp. 1928–1937.
- [24] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” 2014.
- [25] I. Kostrikov, “Pytorch implementations of reinforcement learning algorithms,” <https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail>, 2018.

- [26] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, “Score-cam: Score-weighted visual explanations for convolutional neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 24–25.
- [27] F. Nadeem, “Multi-modal reinforcement learning with videogame audio to learn sonic features,” Master’s thesis, Massachusetts Institute of Technology, 2020.
- [28] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: <http://arxiv.org/abs/1810.04805>
- [29] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [30] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, “Agent57: Outperforming the atari human benchmark,” *arXiv preprint arXiv:2003.13350*, 2020.

DARKGAN: EXPLOITING KNOWLEDGE DISTILLATION FOR COMPREHENSIBLE AUDIO SYNTHESIS WITH GANS

Javier Nistal,^{1,2} Stefan Lattner,² Gaël Richard¹

¹LTCI, Telecom Paris, IP Paris, France

²Sony Computer Science Laboratories (CSL), Paris, France

javier.nistalhurle@sony.com

ABSTRACT

Generative Adversarial Networks (GANs) have achieved excellent audio synthesis quality in the last years. However, making them operable with semantically meaningful controls remains an open challenge. An obvious approach is to control the GAN by conditioning it on metadata contained in audio datasets. Unfortunately, audio datasets often lack the desired annotations, especially in the musical domain. A way to circumvent this lack of annotations is to generate them, for example, with an automatic audio-tagging system. The output probabilities of such systems (so-called "soft labels") carry rich information about the characteristics of the respective audios and can be used to distill the knowledge from a teacher model into a student model. In this work, we perform knowledge distillation from a large audio tagging system into an adversarial audio synthesizer that we call DarkGAN. Results show that DarkGAN can synthesize musical audio with acceptable quality and exhibits moderate attribute control even with out-of-distribution input conditioning. We release the code and provide audio examples on the accompanying website.

1. INTRODUCTION

Generative Adversarial Networks (GANs) [1] have achieved impressive results in image and audio synthesis [2–6]. However, it is still an open challenge to learn comprehensible features that capture semantically meaningful properties of the data. In the graphical domain, semantic control is achieved with GANs using semantic layouts [4] or high-level attributes learned through unsupervised methods [2]. Other works achieve disentanglement through regularization terms [7] or explore the latent space for human-interpretable factors of variation [8, 9]. The great success of these approaches is partly enabled by the availability of large-scale image datasets containing rich semantic annotations [10–12]. However, the context is different in the audio domain, where datasets are scarce and often limited in size and availability of annotations.

Therefore, in this work, we test if limited annotations in audio datasets can be circumvented by taking a Knowledge Distillation (KD) approach. To that end, we utilize the soft labels generated by a pre-trained audio-tagging system for conditioning a GAN in an audio generation task. More precisely, we train the GAN on a subset of the NSynth dataset [13], which contains a wide range of instruments from acoustic, electronic, and synthetic sources. For that dataset we generate soft labels with a publicly available audio-tagging model [14], pre-trained with attributes of the AudioSet ontology [15]. This ontology contains a structured collection of sound events from many different sources and descriptions of around 600 attributes obtained from YouTube videos (e.g., "singing bowl", "sonar", "car", "siren", or "bird").

The soft labels indicate how much of the different characteristics are contained in a specific sound (e.g., a synthesizer sound may have some similarity with a singing bowl or a sonar pulse). We hope that the generative model can distill such characteristics (e.g., the "essence" of a singing bowl sound) and is then able to emphasize them in the generation. The slight similarities to specific categories in data that can be distilled using soft labels were coined "Dark Knowledge" in [16]. Therefore, we call the proposed model DarkGAN.

This paper introduces a generic audio cross-task KD framework for transferring semantically meaningful features into a neural audio synthesizer. We implement this framework in DarkGAN, an adversarial audio synthesizer for comprehensible and controllable audio synthesis. We perform an experimental evaluation on the quality of the generated material and the semantic consistency of the learned attribute controls. Numerous audio examples are provided in the accompanying web page,¹ and the code is released for reproducibility.²

In what follows, we first mention relevant state-of-the-art works in neural audio synthesis and KD (see Sec. 2). In Sec. 3, some background on dark knowledge and KD is given, and its application to controllable neural audio synthesis is motivated. Next, we describe the experimental framework of DarkGAN (see Sec. 4). In Sec. 5 we provide a discussion of the results, and conclude in Sec. 6.



© J. Nistal, S. Lattner, and G. Richard. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: J. Nistal, S. Lattner, and G. Richard, "DarkGAN: Exploiting Knowledge Distillation for Comprehensible Audio Synthesis with GANs", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ <https://an-1673.github.io/DarkGAN.io/>

² <https://github.com/SonyCSLParis/DarkGAN>

2. PREVIOUS WORK

In this section we review some of the most important works on neural audio synthesis and knowledge distillation, paying particular attention to those works tackling tasks similar to ours.

2.1 Neural Audio Synthesis

Many works have applied deep generative methods to address general audio synthesis. These can be categorised into *exact*, *approximate*, and *implicit* density estimation methods. In the first category, autoregressive models of raw audio are state-of-the-art in different audio synthesis tasks [13, 17, 18]. Popular *approximate* density estimation methods are based on Variational Auto-Encoders (VAE) [19]. One of the main advantages of VAEs compared to other approaches is the control they offer over the generative process by manipulating a latent space learned directly from the audio data [20]. Even though latent spaces tend to self-organize according to high-level dependencies in the data, these are still difficult to interpret. Therefore, some works try to impose musically meaningful priors over the structure of these spaces [21–23], or enforce an information bottle-neck by restricting such latent codes to discrete representations to capture fundamental and meaningful features [24, 25].

Generative Adversarial Networks (GANs) [1] belong to the *implicit* density estimation methods. Applications of GANs to audio synthesis have mainly focused on speech tasks [26–32]. The first application to synthesis of musical audio was WaveGAN [33]. Although it did not match autoregressive baselines such as WaveNet [17] in terms of audio quality, it could generate piano and drum sounds quickly and in an entirely unconditional way. Recent improvements in the stabilization and training of GANs [34–36] enabled GANSynth [5] to outperform WaveNet baselines on the task of audio synthesis of musical notes using sparse, pitch conditioning labels. Follow-up works building on GANSynth applied similar architectures to conditional drum sound synthesis using different metadata [6, 37]. DrumGAN [6] synthesizes a variety of drum sounds based on high-level input features describing timbre (e.g., boominess, roughness, sharpness). A few other works have used GANs in a variety of audio tasks like Mel-spectrogram inversion [31], audio domain adaptation [38, 39] or audio enhancement [40].

2.2 Knowledge Distillation

High-performing models are often built upon classifier ensembles that aggregate their predictions to improve the overall accuracy. Despite having excellent performance, these models tend to be large and slow, impeding their use in memory-limited and real-time environments. Different methods exist for optimizing memory consumption and reducing the size of large models or ensembles, e.g., pruning, transfer learning, or quantization. Model compression allows to transfer the function learned by a teacher ensemble or a single large discriminative model into a compact, faster student model exhibiting comparable performance [41]. Instead of training the student model directly

on a hand-labeled categorical dataset, this method employs a pre-trained teacher model to re-label the dataset and then train the compact neural network on this teacher-labeled dataset, using the raw predictions as the target. This training framework was shown to yield efficient models which perform better than if they had been trained on the hand-labeled dataset in a variety of discriminative tasks [41–43]. Model compression was further extended and formalized into the general Knowledge Distillation (KD) framework [16].

KD has been extensively applied in various fields and with other ends than model compression [44–46]. An interesting line of research that is closely related to ours proposes cross-task KD from image captioning and classification systems into an image synthesis generative neural-network [46, 47]. In audio, KD was extensively used on Automatic Speech Recognition (ASR) tasks in order to exploit large unlabelled datasets [43], distill the knowledge from deep Recurrent Neural Networks (RNN) [48] or, inversely, to improve the performance of deep RNN models by distilling knowledge from simple models as a regularization technique [49]. Works related to ours use KD as a means to adapt a model to a different audio domain task [50] or even data modality (by distilling knowledge from a video classifier) [51], where labeled datasets are scarce, and large models would easily overfit. Some works employ KD to fuse knowledge from different audio representations into a single compact model [52].

3. BACKGROUND

This section provides a brief introduction to dark knowledge and explains the general knowledge distillation framework.

3.1 Dark Knowledge

In the seminal work on Knowledge Distillation (KD) [16], the authors demonstrate that the improved performance of smaller models is due to the implicit information existent in the teacher’s output probabilities (i.e., soft labels). As opposed to hard labels, soft labels contain probability values for all of the output classes. The relative probability values that a specific data instance takes for each class contain information about how the teacher generalized the discriminative task. This hidden information existent in the relative probability values was termed *dark knowledge* [53]. An interesting observation on dark knowledge is that in KD, the student model can learn to correctly classify categories even if the training set does not contain examples thereof [16]. In DarkGAN, we test if this principle can be transferred to audio generation. Many of the AudioSet attributes are not directly linked with the actual training data (e.g., the attributes "reverberation", "meow", or "drum" have little or no relationship to the tonal instrument sounds of the NSynth dataset). However, we hope that the implicit dark knowledge existent in the teacher-labeled data can help DarkGAN learn a coherent feature control over such attributes.

3.2 Knowledge Distillation

Multi-label classifiers typically produce a probability distribution over a set of classes by using a *sigmoid* output layer that converts the so-called logit (the NN output before the activation function), z_i , computed for the i th class into a probability q_i as

$$q_i = \frac{1}{1 + e^{-\frac{z_i}{T}}}, \quad (1)$$

where T is a temperature that is typically set to 1. In KD, knowledge is transferred to the distilled model by training it on the teacher-labeled data, using a higher temperature. By that, the distribution gets "compressed," emphasizing lower probability values. The same (higher) temperature is used while training the distilled model, but the temperature is set back to 1 after training. As for cost function, the binary cross-entropy is used as

$$H_s(q) = -\frac{1}{N} \sum_{i=1}^N p_i \log(q_i) + (1 - p_i) \log(1 - q_i), \quad (2)$$

where $N = 128$ is the number of attributes, p_i are the soft-labels predicted by the teacher, and q_i is the probability predicted by the student model for the i th class.

4. EXPERIMENTS

In this section, details are given about the conducted experiments. We describe the AudioSet ontology, the teacher and student architectures, the metrics employed for evaluation, and the baselines used for comparison.

4.1 Dataset

We employ a subset from NSynth [13] for our experiments. NSynth contains approximately 300k single-note audios played by more than 1k different instruments from 10 different families. Each sample's onset occurs at time 0. The dataset contains various labels (e.g., pitch, velocity, instrument type), but we only use (i.e., condition the model on) pitch information in this work. Each sample is four seconds long, with a 16kHz sample rate. For computational simplicity, we use only the first second of each sample. Also, we only consider samples with a MIDI pitch range from 44 to 70 (103.83 - 466.16 Hz), resulting in a subset of approximately 90k sounds equally distributed across the pitch classes. For the evaluation, we perform a 90/10% split of the data.

Previous works on adversarial audio synthesis [5, 54] demonstrated that the Magnitude and Instantaneous Frequency of the STFT works well as a representation for harmonic sounds. We use an FFT size of 2048 bins and an overlap of 75%.

4.2 The AudioSet Ontology

AudioSet [15] is a large-scale dataset containing audio data and an ontology of sound events that seek to describe real-world sounds. It was created to set a benchmark in

the development of automatic audio event recognition systems, similar to those in computer-vision, such as ImageNet [10]. The dataset consists of a structured vocabulary of 632 audio event classes and a collection of approximately 2M human-labeled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchy of categories with a maximum depth of 6 levels, covering a wide range of human and animal sounds, musical genres and instruments, and environmental sounds. We encourage the reader to visit the corresponding website for a complete description of the ontology.³

In this work, we do not employ all of the AudioSet attributes, as many of them refer to properties that are too vague for musical sounds or describe broader time-scale aspects of the sound (e.g., music, chatter, sound effect). Instead, we rank the attributes based on the geometric mean of their 90th percentile (calculated on the predicted class probabilities for each attribute across the dataset), and the teacher's reported accuracy as $\sqrt{p_{90th}^i \times acc^i}$. Then, we take the first 128 attributes according to this ranking.

4.3 Models

In the following, we introduce the teacher model and DarkGAN's architecture.

4.3.1 Pre-trained AudioSet Classifier

In this work, we distill the knowledge from a pre-trained audio-tagging neural network (PANN) trained on raw audio recordings from the AudioSet collection [14]. PANNs were originally proposed for transferring knowledge to other audio pattern recognition tasks. However, we use them to transfer the knowledge to a generative model and steer the generation process through a comprehensible vocabulary of attributes.

We employ the *CNN-14* model from the PANNs [14]. *CNN-14* is built upon a stack of 6 convolution-based blocks containing 2 CNN layers with a kernel size of 3x3. Batch Normalization is applied after every convolutional layer, and a ReLU non-linearity is used as the activation function. After each convolutional block, they apply an average-pooling layer of size 2x2 for down-sampling. Global pooling is applied after the last convolutional layer to summarize the feature maps into a fixed-length vector. An extra fully-connected layer is added to extract embedding features before the output Sigmoid activation function. For more details on the architecture, please refer to [14].

4.3.2 DarkGAN

The proposed GAN architecture, illustrated in Fig. 1, follows the architecture of DrumGAN [6]. The input to G is a concatenation of 128 teacher-labeled AudioSet attributes $\alpha \in [0, 1]^{128}$ (see Sec. 4.2), a one-hot vector $p \in \{0, 1\}^{26}$ containing the pitch class, and a random vector $z \sim \mathcal{N}_{32}(0, 1)$. The resulting vector is placed as a column in the middle of a 4D tensor with $128 + 32 + 26$ convolutional maps. Then, it is fed through a stack of convolutional and box up-sampling blocks to generate the out-

³ research.google.com/audioset/ontology/

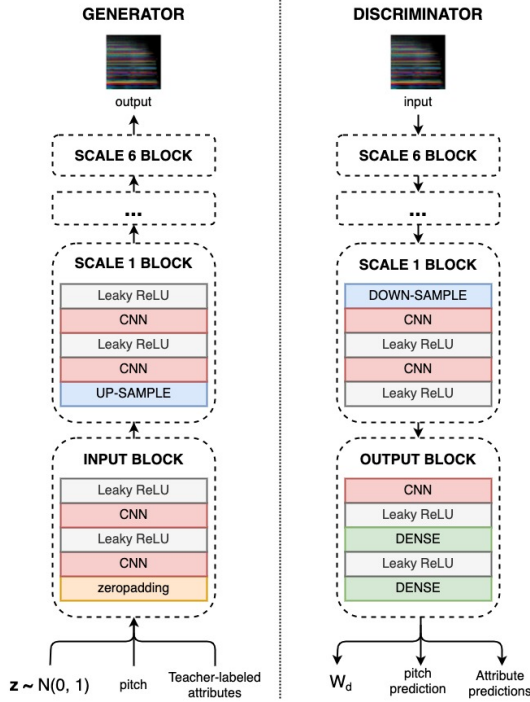


Figure 1. Proposed architecture for DarkGAN [6]

put signal $x = G_\theta(z, p, \alpha)$. The number of feature maps decreases from low to high resolution as $\{256, 128, 128, 128, 128, 64\}$. The discriminator D mirrors G 's configuration and estimates the Wasserstein distance W_d between the real and generated distributions [35], and predicts the AudioSet features accompanying the input audio in the case of a real batch, or those used for conditioning in the case of generated audio. In order to promote the usage of the conditioning information by G , we add to the objective function an auxiliary binary cross-entropy loss term for the distillation task and a categorical cross-entropy for the pitch classification task [55].

4.4 Evaluation

The task of synthesizing perceptually realistic audio is hard to formalize. In conditional models, as is the case in this work, an additional challenge is to assess whether the model is soundly responsive to the conditional input. In order to evaluate these properties of our model, a diverse set of objective metrics are computed. We compute these metrics for DarkGAN when trained under different temperature values in the distillation process (see Sec. 2.2), as well as for various baselines. In this section, we describe these metrics as well as the baselines used for comparison.

4.4.1 Scores and distances

Following previous methodology [6, 54, 56], we compare real and generated distributions employing these metrics:

- **Inception Score (IS)** [36] penalizes models whose samples cannot be reliably classified into a single class or that only belong to a few from all possible classes. We report on the Pitch Inception Score (PIS) and the Instrument Inception Score (IIS) [54].
- **Kernel Inception Distance (KID)** [57] measures the dissimilarity between embeddings of real and

generated samples. A low KID means that the generated and real distributions are close to each other.

- **Fréchet Audio Distance (FAD)** [58] measures the distance between continuous multivariate Gaussians fitted to embeddings of real and generated data. The lower the FAD, the smaller the distance between distributions of real and generated data.

4.4.2 Consistency of Attribute Controls

This work aims to learn semantically meaningful controls with DarkGAN by distilling knowledge from an audio-tagging system trained on attributes from the AudioSet ontology. Therefore, we evaluate if changing an input attribute is reflected in the corresponding output of DarkGAN. To that end, we examine the change in the prediction of the teacher model (w.r.t. the output of DarkGAN) when changing a particular DarkGAN input attribute. A second property to assess is whether the *dark knowledge* helps DarkGAN learn well-formed representations of specific attributes and generalize to out-of-distribution input combinations. To assess these two aspects, we perform the following tests:

1. *Attribute correlation*: we generate 10k samples using attribute vectors from the validation set as input to DarkGAN. The generated samples are fed to the teacher model to predict the attributes again. Then, for each attribute i , we compute the correlation between the input vector α and the predictions $\hat{\alpha}$ as

$$\rho^i(\hat{\alpha}, \alpha) = \rho(F^i(G(z, p, \alpha)), \alpha),$$

where F^i is the classifier's prediction for the i th attribute, p is the pitch, and z is the random noise.

2. *Out-of-distribution Attribute Correlation*: for each attribute i exhibiting a positive correlation, i.e., $\mathcal{S} = \{\rho^i : \rho^i > 0\}$, test (1) is repeated 50 times, but using 1k samples instead of 10k. In each repetition, a specific attribute is progressively incremented by an amount $\delta_l := 10^{-3+l\frac{3.6}{50}}$, $l = 0, 1, \dots, 50^*$ and we calculate

$$\bar{\rho}_{\delta_l} = \frac{1}{|\mathcal{S}|} \sum_{\mathcal{S}} \rho^i(\hat{\alpha}, \alpha + \delta_l).$$

3. *Increment consistency*: for the 50 attributes with the highest correlation, we compute

$$\Delta F_{\delta_k} = \sum_{i=1}^{50} \sum_{j=1}^{100} \frac{F^i(G(z_j, p_j, \alpha_j + \delta_k)) - F^i(G(z_j, p_j, \alpha_j))}{50 \times 100 \times \text{std}(F^i(G(\mathbf{z}, \mathbf{p}, \alpha)))},$$

where α_j is the j th original feature vector from a set of 100 samples randomly picked from the validation set, and $\delta_k := \frac{k}{5}$, $k = 0, 1, \dots, 25$. Intuitively, it is defined as the average difference of the predicted attributes of the generated audios (i.e., the difference before and after the attribute increment) as a function of the increment δ_k . We express the result in terms of standard deviations of the non-incremented generated examples as $\text{std}(G(z, p, \alpha))$.

*The step of δ_l is defined to obtain more density of points in the range of variation of the attributes (i.e., $[0, 1]$) as well as $\delta_l > 1$.

4.4.3 Baselines

We compare the metrics described above with *real data* to obtain a baseline for each metric. Also, GANSynth [5], the state-of-the-art on audio synthesis with GANs, is used for comparison.⁵ As GANSynth generates 4-second long sounds, the waveform is trimmed down to 1 second for comparison with our models. Additionally, we examine the effect that KD has on these metrics by comparing against a model analogous to DarkGAN, but without using the AudioSet feature conditioning (*baseline*). Experiment results for DarkGAN are shown for different temperature values $T \in \{1, 1.5, 2, 3, 5\}$ (1) as part of the KD process (see Sec. 3.2), and we report separate results for conditional attributes obtained from the training (tr) and validation (val) set.

5. RESULTS

In this section, we present the results from the evaluation procedure described in Sec. 4.4. Furthermore, we validate the quantitative results based on an informal assessment of the generated content.

5.1 Quantitative Metrics

Table 1 presents the metrics scored by DarkGAN_T and the baseline models, as described in Sec. 4.4.3. Note that we condition DarkGAN on attribute vectors randomly sampled from the validation set. Overall, DarkGAN_{T∈{1.5,2}} obtains better results than the baselines and is close to *real data* in most metrics. All models score higher PIS than *real data*, with GANSynth in the first place, suggesting that the generated examples have a clear pitch and that the distribution of pitch classes follows that of the training data. This is not surprising, as all the models have explicit pitch conditioning. In contrast, we do not provide conditioning attributes for the instrument class. Therefore, we observe a slight drop in IIS for all models compared to *real data*. DarkGAN_{T∈{1.5,2}} achieves the highest IIS, suggesting that the model captured the timbre diversity existent in the dataset and, also, that the generated sounds can be reliably classified into one of all possible instruments. In terms of KID, DarkGAN_{T∈{1.5,2}} and *baseline* are on a par with *real data*. A KID equal to *real data* indicates that the Inception embeddings are similarly distributed for real and generated data. As our Inception classifier is trained on pitch and instrument classification and predicting AudioSet features, similarities in such an embedding space indicate common timbral and tonal characteristics between the generated and the real audio data distribution. This trend is maintained in the case of the FAD, where DarkGAN_{T=2} obtains the best scores followed closely by DarkGAN_{T∈{1,1.5}}.

From the results discussed above, we can conclude that distilling knowledge from the AudioSet classifier helps DarkGAN learning the real data distribution. Furthermore, using slightly higher temperatures in the distillation process yields an improvement over the *baseline* without feature conditioning. We speculate that the additional super-

Model	PIS↑		IIS↑		KID↓ ^a		FAD↓	
<i>real data</i>	17.7		5.7		6.7		0.1	
GANSynth [5]	19.6		4.0		7.1		4.5	
<i>baseline</i>	18.5		4.3		6.7		0.8	

DarkGAN _T	tr		val		tr		val	
$T = 1$	18.4	18.3	4.0	4.0	6.8	6.8	0.7	0.7
$T = 1.5$	19.0	19.0	4.5	4.5	6.7	6.7	0.7	0.7
$T = 2$	19.1	19.0	4.2	4.1	6.7	6.8	0.6	0.6
$T = 3$	19.1	19.1	4.2	4.1	6.8	6.8	0.8	0.8
$T = 5$	19.2	19.1	4.0	4.0	6.8	6.8	0.8	0.8

^a × 10⁻⁴

Table 1. PIS, IIS, KID and FAD (see Sec. 4.4)

Attribute	T=1	T=1.5	T=2	T=3	T=5
Acoustic guitar	0.20	0.36	0.39	0.23	0.10
Bass guitar	0.30	0.38	0.46	0.38	0.19
Brass Instrument	0.28	0.49	0.38	0.26	0.00
Cello	0.24	0.29	0.26	0.17	0.00
Chime	0.15	0.33	0.39	0.31	0.03
Guitar	0.28	0.37	0.42	0.34	0.13
Plucked string	0.27	0.37	0.42	0.32	0.11
Saxophone	0.25	0.41	0.41	0.41	0.03
Trombone	0.18	0.41	0.29	0.16	0.00
Trumpet	0.16	0.46	0.36	0.25	0.00
...					
Didgeridoo	0.06	0.16	0.21	0.20	0.08
Drum	0.05	0.21	0.24	0.12	0.01
Electronic tuner	0.35	0.44	0.50	0.29	0.13
Percussion	0.04	0.19	0.30	0.14	0.08
Sine wave	0.28	0.32	0.27	0.17	0.10
Singing bowl	0.08	0.20	0.24	0.21	0.03
Siren	0.13	0.19	0.24	0.10	0.08
Tuning fork	0.22	0.29	0.35	0.29	0.10
Zither	0.03	0.18	0.19	0.07	-0.01
...					
Cat	-0.01	-0.01	-0.01	-0.01	0.00
Chicken, rooster	0.00	-0.06	-0.02	-0.01	-0.01
Domestic animals, pets	-0.01	-0.02	-0.02	0.00	0.00
Frog	0.00	0.03	0.07	0.06	-0.03
Insect	0.00	-0.02	-0.02	-0.02	-0.01
Speech	-0.04	-0.10	-0.07	-0.05	0.01

Table 2. A few examples of attribute correlation coefficients $\rho^i(\hat{\alpha}, \alpha)$ (see Sec. 4.4.2).

vised information that the teacher model provides to DarkGAN’s discriminator results in a more meaningful gradient for the generator. Also, attribute conditioning (i.e., attribute vectors sampled from the validation set) may help the generator synthesize diverse samples closer to the training data distribution.

5.2 Attribute Consistency and Generalisation

Note that the metrics discussed in this section are not guaranteed to relate directly to human perception, but we consider them suitable indicators of whether the model responds coherently to the input conditioning. There exists the threat of the generator producing adversarial examples, but we argue that this is prevented by the discriminator having to satisfy the Wasserstein criterion (as adversarial examples would exhibit out-of-distribution artifacts). This assumption is also supported by informal listening tests where we find that the metrics correlate with our perception (see Sec. 5.3).

Table 2 shows the results for the *attribute correlation* $\rho^i(\hat{\alpha}, \alpha)$ (see Sec. 4.4.2). At the top of the table, we show a few attributes corresponding to classes represented in the NSynth dataset (e.g., "guitar", "trumpet"). In the middle, we show attributes that, while not being present in the dataset (e.g., "siren", "tuning fork"), still exhibit (relatively) high correlation. At the bottom, attributes that ob-

⁵ <https://github.com/magenta/magenta/tree/master/magenta/models/gansynth>

tain low correlations are presented (e.g., "cat", "insect"). We can observe that models trained with $T \in \{1.5, 2, 3\}$ generally obtain better results than $T \in \{1, 5\}$ in most attributes. Specifically, $\text{DarkGAN}_{T=2}$ yields the highest correlations, followed by $\text{DarkGAN}_{T=1.5}$. Note that temperatures higher than 1 also improve the correlation for attributes that do not have corresponding classes in the dataset (e.g., "didgeridoo", "percussion", "singing bowl"). This suggests that DarkGAN can extract dark knowledge (which is emphasized by increasing T) from the soft labels. The soft labels indicating the presence of (potentially just slight) timbral characteristics in various sounds are helping the model to learn linearly dependent feature controls for those attributes.

A more in-depth analysis of feature errors and the distribution of features in the dataset would be required to further characterize the results for each attribute. However, it is reasonable that those classes obtaining higher correlations share some timbral features with the training data (e.g., clearly, "violins" are contained in the data set, and a "tuning fork" is similar to a "mallet"). In contrast, those attributes obtaining low correlations may be related to underrepresented features in the training set or features that the model failed to capture.

Fig. 2 shows the correlation coefficient when increasing each attribute by a value δ_l in the input conditioning. The plot reveals that the trend of Table 2 is maintained throughout an ample range of variation of the attributes. Interestingly, while the correlation of $\text{DarkGAN}_{T=1}$ considerably declines after an increase $\delta_l > 10^{-0.8}$, using a temperature $T \in \{1.5, 2, 3\}$ the decline is more moderate, and we observe some correlation even for a $\delta_l > 1$, which is outside the range of the attributes.

As the correlation coefficient provides normalized results (regarding scale and offsets), we evaluate the attribute control using the *increment consistency* metric $\overline{\Delta F}_{\delta_k}$ (see Fig. 3). We observe that for low increments of the features ($\delta_k < 1$) temperatures $T \in \{1, 1.5, 2\}$ yield comparable input-output relationships of the features. A temperature $T = 1.5$, however, yields more consistent feature differences for increments $\delta_k > 1$ of the conditional input features. In conclusion, while $\text{DarkGAN}_{T=2}$ yields better correlation over all the data (i.e., conditional and predicted attributes are more strongly dependent), for attributes with particularly high correlation, $\text{DarkGAN}_{T=1.5}$ performs best in over-emphasizing dark knowledge contained in the data (i.e., the degree of change is higher, especially for $\delta_k > 1$).

5.3 Informal Listening

In the accompanying website,⁶ we show sounds generated under various conditioning settings, including generations with feature combinations randomly sampled from the validation set, generations where we fix α and p while changing z , timbre transfer, scales, and more. Overall, we find the results of PIS, IIS, KID, and FAD, discussed in Sec. 5.1, to align well with our perception. The quality of the generated audio is acceptable for all models. Also,

⁶ <https://an-1673.github.io/DarkGAN.io/>

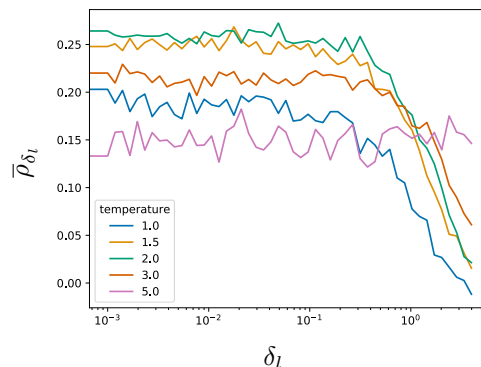


Figure 2. Out-of-distribution average attribute correlation $\bar{\rho}_{\delta_l}$ (see Sec. 4.4.2)

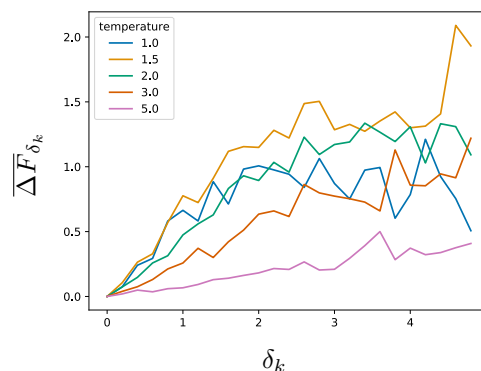


Figure 3. Increment consistency $\overline{\Delta F}_{\delta_k}$ (see Sec.4.4.2)

we find the generated examples to be diverse in terms of timbre, and the tonal content is coherent with the pitch conditioning. Moreover, we perceive that most of the attributes exhibiting high correlations (see Table 2) are audible in the generated output, particularly in the case of $\text{DarkGAN}_{T \in \{1, 1.5, 2\}}$. For higher temperatures $T \in \{3, 5\}$, the model’s responsiveness to the attribute conditioning drops substantially. We find the model to be particularly responsive to attributes such as "drum", "tuning fork", "theremin", "choir", or "cowbell". To other attributes (e.g., "accordion", "piano", or "organ"), even though the analysis yields moderate correlations, the model does not seem to produce perceptually satisfactory outputs.

6. CONCLUSION

In this work, we distilled knowledge from a large-scale audio tagging system into DarkGAN, an adversarial synthesizer of tonal sounds. The goal was to enable steering the synthesis process using attributes from the AudioSet ontology. A subset of the NSynth dataset was fed to a pre-trained audio tagging system to obtain AudioSet predictions. These predictions were then used to condition DarkGAN. The proposed Knowledge Distillation (KD) framework was evaluated by comparing different temperature settings and employing a diverse set of metrics. Results showed that DarkGAN can generate audio resembling the true dataset and enables moderate control over a comprehensible vocabulary of attributes. By slightly increasing the temperature during the distillation process, we can further improve the responsiveness of the attribute controls. It is also notable that KD can be performed even when the original dataset (i.e., the AudioSet collection) is not involved.

7. ACKNOWLEDGEMENTS

This research is supported by the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 765068 (MIP-Frontiers).

8. REFERENCES

- [1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, “Generative adversarial nets,” in *Proc. of the 28th International Conference on Neural Information Processing Systems NIPS*, Montreal, Quebec, Canada, Dec. 2014.
- [2] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*. Seattle, WA, USA: IEEE, June 2020, pp. 8107–8116.
- [3] A. Brock, J. Donahue, and K. Simonyan, “Large scale GAN training for high fidelity natural image synthesis,” in *7th International Conference on Learning Representations, ICLR*. New Orleans, LA, USA: OpenReview.net, May 2019.
- [4] T. Park, M. Liu, T. Wang, and J. Zhu, “Semantic image synthesis with spatially-adaptive normalization,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Long Beach, CA, USA: Computer Vision Foundation / IEEE, June 2019, pp. 2337–2346.
- [5] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, “Gansynth: Adversarial neural audio synthesis,” in *Proc. of the 7th International Conference on Learning Representations, ICLR*, May 2019.
- [6] J. Nistal, S. Lattner, and G. Richard, “Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks,” in *Proc. of the 21st Int. Society for Music Information Retrieval Conf.*, Montréal, Canada, 2020.
- [7] W. S. Peebles, J. Peebles, J. Zhu, A. A. Efros, and A. Torralba, “The hessian penalty: A weak prior for unsupervised disentanglement,” in *Computer Vision - ECCV 2020 - 16th European Conference*, ser. Lecture Notes in Computer Science, vol. 12351. Glasgow, UK: Springer, August 2020, pp. 581–597.
- [8] A. Voynov and A. Babenko, “Unsupervised discovery of interpretable directions in the GAN latent space,” in *Proceedings of the 37th International Conference on Machine Learning, ICML*, ser. Proceedings of Machine Learning Research, vol. 119. Virtual Event: PMLR, July 2020, pp. 9786–9796.
- [9] Y. Shen, J. Gu, X. Tang, and B. Zhou, “Interpreting the latent space of gans for semantic face editing,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*. Seattle, WA, USA: IEEE, June 2020, pp. 9240–9249.
- [10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, June 2009.
- [11] H. Caesar, J. R. R. Uijlings, and V. Ferrari, “Coco-stuff: Thing and stuff classes in context,” in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. Salt Lake City, UT, USA: IEEE Computer Society, June 2018, pp. 1209–1218.
- [12] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” *CoRR*, vol. abs/1708.07747, 2017.
- [13] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with wavenet autoencoders,” in *Proc. of the 34th International Conference on Machine Learning, ICML*, Sydney, NSW, Australia, Aug. 2017.
- [14] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, “Panns: Large-scale pretrained audio neural networks for audio pattern recognition,” *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 2880–2894, 2020.
- [15] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. New Orleans, LA, USA: IEEE, March 2017, pp. 776–780.
- [16] G. E. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *CoRR*, vol. abs/1503.02531, 2015.
- [17] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” in *Proc. of the 9th ISCA Speech Synthesis Workshop*, Sunnyvale, CA, USA, Sept. 2016.
- [18] S. Vasquez and M. Lewis, “Melnet: A generative model for audio in the frequency domain,” *CoRR*, vol. abs/1906.01083, 2019.
- [19] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Proc. of the 2nd International Conference on Learning Representations, ICLR*, Banff, AB, Canada, Apr. 2014.
- [20] C. Aouameur, P. Esling, and G. Hadjeres, “Neural drum machine: An interactive system for real-time synthesis of drum sounds,” in *Proc. of the 10th International Conference on Computational Creativity, ICC*, June 2019.

- [21] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, “Universal audio synthesizer control with normalizing flows,” *Journal of Applied Sciences*, 2019.
- [22] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, “Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces,” in *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR*, Paris, France, Sept. 2018.
- [23] —, “Generative timbre spaces with variational audio synthesis,” in *Proc. of the 21st International Conference on Digital Audio Effects DAFX-18*, Aveiro, Portugal, Sept. 2018.
- [24] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A generative model for music,” *CoRR*, vol. abs/2005.00341, 2020.
- [25] O. Cífka, A. Ozerov, U. Simsekli, and G. Richard, “Self-supervised VQ-VAE for one-shot music style transfer,” *CoRR*, vol. abs/2102.05749, 2021.
- [26] Y. Saito, S. Takamichi, and H. Saruwatari, “Statistical parametric speech synthesis incorporating generative adversarial networks,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 26, pp. 84–96, 2018.
- [27] T. Kaneko and H. Kameoka, “Parallel-data-free voice conversion using cycle-consistent adversarial networks,” *CoRR*, vol. abs/1711.11293, 2017.
- [28] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, “Timbretron: A wavenet(cyclegan(cqt(audio))) pipeline for musical timbre transfer,” in *Proc. of the 7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019.
- [29] M. Binkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, “High fidelity speech synthesis with adversarial networks,” in *8th International Conference on Learning Representations, ICLR*, Addis Ababa, Ethiopia, April 2020.
- [30] J. Kong, J. Kim, and J. Bae, “Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis,” in *Annual Conference on Neural Information Processing Systems, NeurIPS*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., Virtual conference, December 2020.
- [31] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, “Melgan: Generative adversarial networks for conditional waveform synthesis,” in *Proc. of the Annual Conference on Neural Information Processing Systems, NIPS*, Vancouver, BC, Canada, Dec. 2019.
- [32] R. Yamamoto, E. Song, and J. Kim, “Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. Barcelona, Spain: IEEE, May 2020, pp. 6199–6203.
- [33] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” in *Proc. of the 7th International Conference on Learning Representations, ICLR*, May 2019.
- [34] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” in *International Conference on Learning Representations, ICLR*, May 2018.
- [35] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein gans,” in *Proc. of the International Conference on Neural Information Processing Systems, NIPS*, Long Beach, CA, USA, Dec. 2017.
- [36] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” in *Proc. of the International Conference on Neural Information Processing Systems, NIPS*, Barcelona, Spain, Dec. 2016.
- [37] J. Drysdale, M. Tomczak, and J. Hockman, “Adversarial synthesis of drum sounds,” in *DAFX*, 2020.
- [38] E. Hosseini-Asl, Y. Zhou, C. Xiong, and R. Socher, “A multi-discriminator cyclegan for unsupervised non-parallel speech domain adaptation,” in *Proc. of the 19th Annual Conference of the International Speech Communication Association*, Hyderabad, India, Sept. 2018.
- [39] D. Michelsanti and Z. Tan, “Conditional generative adversarial networks for speech enhancement and noise-robust speaker verification,” in *Proc. of the 18th Annual Conference of the International Speech Communication Association, Interspeech*, Stockholm, Sweden, Aug. 2017.
- [40] A. Biswas and D. Jia, “Audio codec enhancement with generative adversarial networks,” in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. Barcelona, Spain: IEEE, May 2020, pp. 356–360.
- [41] C. Bucila, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, Eds. Philadelphia, PA, USA: ACM, August 2006, pp. 535–541.
- [42] J. Ba and R. Caruana, “Do deep nets really need to be deep?” in *Annual Conference on Neural Information Processing Systems, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds.*, Montreal, Quebec, Canada, December 2014, pp. 2654–2662.

- [43] J. Li, R. Zhao, J. Huang, and Y. Gong, “Learning small-size DNN with output-distribution-based criteria,” in *15th Annual Conference of the International Speech Communication Association, INTERSPEECH*, H. Li, H. M. Meng, B. Ma, E. Chng, and L. Xie, Eds. Singapore: ISCA, September 2014, pp. 1910–1914.
- [44] N. Papernot, M. Abadi, Ú. Erlingsson, I. J. Goodfellow, and K. Talwar, “Semi-supervised knowledge transfer for deep learning from private training data,” in *5th International Conference on Learning Representations, ICLR*, Toulon, France, April 2017.
- [45] R. Anil, G. Pereyra, A. Passos, R. Ormándi, G. E. Dahl, and G. E. Hinton, “Large scale distributed neural network training through online distillation,” in *6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, May 2018.
- [46] M. Yuan and Y. Peng, “CKD: cross-task knowledge distillation for text-to-image synthesis,” *IEEE Trans. Multim.*, vol. 22, no. 8, pp. 1955–1968, 2020.
- [47] ———, “Text-to-image synthesis via symmetrical distillation networks,” in *2018 ACM Multimedia Conference on Multimedia Conference, MM*, S. Boll, K. M. Lee, J. Luo, W. Zhu, H. Byun, C. W. Chen, R. Lienhart, and T. Mei, Eds. Seoul, Republic of Korea: ACM, October 2018, pp. 1407–1415.
- [48] W. Chan, N. R. Ke, and I. Lane, “Transferring knowledge from a RNN to a DNN,” in *16th Annual Conference of the International Speech Communication Association, INTERSPEECH*. Dresden, Germany: ISCA, September 2015, pp. 3264–3268.
- [49] Z. Tang, D. Wang, and Z. Zhang, “Recurrent neural network training with dark knowledge transfer,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. Shanghai, China: IEEE, March 2016, pp. 5900–5904.
- [50] T. Asami, R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono, “Domain adaptation of DNN acoustic models using knowledge distillation,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. New Orleans, LA, USA: IEEE, March 2017, pp. 5185–5189.
- [51] Y. Aytar, C. Vondrick, and A. Torralba, “Soundnet: Learning sound representations from unlabeled video,” in *Annual Conference on Neural Information Processing Systems, NeurIPS*, Barcelona, Spain, December 2016, pp. 892–900.
- [52] L. Gao, K. Xu, H. Wang, and Y. Peng, “Multi-representation knowledge distillation for audio classification,” *CoRR*, vol. abs/2002.09607, 2020.
- [53] G. Hinton, O. Vinyals, and J. Dean, “Dark knowledge,” in *Toyota Technological Institute at Chicago, TTIC*, 2014.
- [54] J. Nistal, S. Lattner, and G. Richard, “Comparing representations for audio synthesis using generative adversarial networks,” in *Proc. of the 28th European Signal Processing Conference, EUSIPCO2020*, Amsterdam, NL, Jan. 2021.
- [55] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” in *Proc. of the 34th International Conference on Machine Learning, ICML*, Sydney, NSW, Australia, Aug. 2017.
- [56] C. Gupta, P. Kamath, and L. Wyse, “Signal representations for synthesizing audio textures with generative adversarial networks,” *CoRR*, vol. abs/2103.07390, 2021.
- [57] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton, “Demystifying MMD gans,” in *Proc. of the 6th International Conference on Learning Representations, ICLR*, Vancouver, BC, Canada, Apr. 2018.
- [58] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet audio distance: A metric for evaluating music enhancement algorithms,” *CoRR*, vol. abs/1812.08466, 2018.

PHASE-AWARE JOINT BEAT AND DOWNBEAT ESTIMATION BASED ON PERIODICITY OF METRICAL STRUCTURE

Takehisa Oyama¹ Ryoto Ishizuka¹ Kazuyoshi Yoshii^{1,2}

¹Graduate School of Informatics, Kyoto University, Japan

²PRESTO, Japan Science and Technology Agency (JST), Japan

{ooyama, ishizuka, yoshii}@sap.ist.i.kyoto-u.ac.jp

ABSTRACT

This paper describes a phase-aware joint beat and downbeat estimation method mainly intended for popular music with a periodic metrical structure and steady tempo. The conventional approach to beat estimation is to train a deep neural network (DNN) that estimates the *beat presence probability* at each frame. This approach, however, relies heavily on a periodicity-aware post-processing step that detects beat times from the noisy probability sequence. To mitigate this problem, we have designed a DNN that estimates the *beat phase* at each frame whose period is equal to the beat interval. The estimation losses computed at all frames not limited to a fewer number of beat frames can thus be effectively used for backpropagation-based supervised training, whereas a DNN has conventionally been trained such that it constantly outputs zero at all non-beat frames. The same applies to downbeat estimation. We also modify the post-processing method for the estimated phase sequence. For joint beat and downbeat detection, we investigate multi-task learning architectures that output beat and downbeat phases in this order, in reverse order, and in parallel. The experimental results demonstrate the importance of phase modeling for stable beat and downbeat estimation.

1. INTRODUCTION

Rhythm analysis of music signals such as beat, downbeat, and tempo estimation often constitutes the crucial front end of automatic music transcription [1, 2] and music structure analysis [3]. The typical approach to beat estimation consists of (1) computing an onset strength signal (OSS) from a music signal and (2) detecting regularly-spaced beat times from the OSS with autocorrelation analysis or comb filtering [4–6]. The step (1) has recently been implemented with a deep neural network (DNN) that outputs the probability of the presence of a beat at each frame [7–11]. In particular, convolutional neural networks (CNNs) attained the noticeable improvement of beat estimation [7–9]. Since the same applies to downbeat estimation, we often focus on only beat estimation in the remainder of the paper.

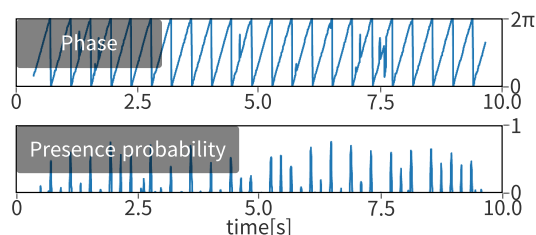


Figure 1. For beat estimation, the proposed DNN aims to estimate a sawtooth-shaped beat phase sequence, whereas a conventional DNN aims to estimate an impulsive beat presence probability sequence.

The major problem of the typical approach is that the periodic nature of beat times is not considered explicitly in the step (1). The performance of DNN-based beat estimation thus heavily depends on the periodicity-aware post-processing step (2) that detects beat times from the noisy probability sequence. This calls for the improved accuracy of the raw output of a DNN used in the step (1).

To solve this problem, in this paper we propose a new approach to beat estimation that aims to estimate not the *beat presence probability* but the *beat phase* at each frame (Fig. 1). Note that a sequence of beat phases is represented as a semi-continuous sawtooth wave whose period corresponds to beat intervals, whereas a sequence of beat presence probabilities as an impulse train that takes one at only beat frames and zero at the other frames. The key advantage of the phase-based representation is that all frames not limited to beat frames give meaningful information about the periodic beat structure. Since a DNN is usually trained with backpropagation such that the sum of frame-level estimation losses is minimized, the phase-based representation would be a more suitable target for periodicity-aware supervised training.

We also propose a post-processing method that detects beat times from the noisy phase sequence (Fig. 2). In recent beat estimation [7–10], a dynamic Bayesian network (DBN) based on the bar-pointer model [12], which is approximately implemented as a hidden Markov model (HMM) [13], is used for picking beat times from a number of peaks included in a beat probability sequence. We modify the observation model of the DBN to deal with a beat phase sequence. The global tempo can be estimated by identifying the most dominant frequency component from the Fourier transform of the noisy sinusoidal wave converted from the estimated phase sequence.

Since beat and downbeat times form the hierarchical metrical structure of music in a mutually dependent manner, we investigate three multi-task learning architectures for joint beat and downbeat estimation. More specifically, one can (1) predict beat phases from an audio spectrogram and then predict downbeat phases from the spectrogram and the estimated beat phases, (2) predict the downbeat and beat phases in this order (the reverse order of (1)), and (3) predict both the beat and downbeat phases in parallel. Several examples estimated by the proposed method are available at <https://phase2bdbt.github.io/>.

2. RELATED WORK

Classical beat estimation methods focus on the periodicity of a music signal with signal processing techniques [5, 6]. In recent years, DNNs have actively been used for directly estimating the probability of the presence of a beat at each frame, given the spectrogram or acoustic features of a music signal. The first attempt for DNN-based beat estimation used a long short-term memory (LSTM) network that estimates a sequence of beat presence probabilities from mel spectrograms with different window lengths [14]. More recently, the temporal convolutional network (TCN) [15] was shown to improve the performance with shorter training time [9]. It consists of dilated convolution layers like WaveNet [16] originally proposed for audio synthesis such that the receptive field of a deeper layer becomes wider exponentially to capture the long-term dependency of time-series data. For better estimation, the TCN used in [9] has a non-causal architecture, *i.e.*, both the past and future input data are used for making a prediction at the current frame, whereas the original TCN has a causal architecture.

For tempo estimation, one can estimate the tempo from a kind of onset strength signal (OSS) extracted from a music signal and detect the most dominant period corresponding to the tempo from the OSS with autocorrelation analysis, comb filtering, or the discrete Fourier transform (DFT) [17–19]. In the same way as beat estimation, the tempo has recently been estimated directly from a music signal with a DNN [20, 21], where the tempo estimation is interpreted as a classification problem. Schreiber *et al.* [20] attempted to estimate local tempos as well as the global tempo. Foroughmand *et al.* [21] proposed a new representation of the DNN input called harmonic constant-Q modulation (HCQM) that represents the harmonic series considering tempo frequencies.

Several multi-task methods have been proposed for joint estimation of mutually-dependent multiple kinds of metrical elements. In the earliest years, Goto [22] proposed a method based on signal processing techniques and expert knowledge of metrical structure for real-time joint beat and downbeat estimation. In recent years, the LSTM was used for joint beat and downbeat estimation [10] and the TCN was used for joint beat and tempo estimation [8], which was extended for joint beat, downbeat, and tempo estimation [7], resulting in the state-of-the-art performances. In [7], a single TCN was shared over three tasks and was trained with a data augmentation technique.

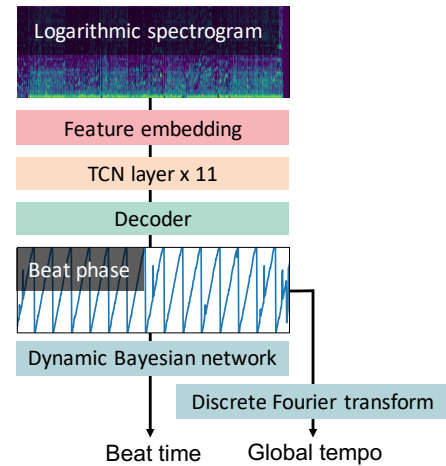


Figure 2. The proposed phase-aware beat estimation followed by tempo estimation.

3. PROPOSED METHOD

This section describes the proposed phase-aware method for rhythm analysis. Our goal is to estimate beat and downbeat times and the global tempo from the log-magnitude spectrogram $\mathbf{X} \in \mathbb{R}^{F \times T}$ of a music signal, where F is the number of frequency bins and T is the number of frames. For beat estimation, we perform DNN-based phase classification (Section 3.1) followed by DBN-based peak picking (Section 3.2) and tempo estimation (Section 3.3). We then describe three possible architectures of multi-task learning for joint beat and downbeat estimation (Section 3.4).

3.1 DNN-Based Beat/Downbeat Phase Classification

We tackle beat phase estimation in terms of a DNN-based classification problem. In our preliminary investigation on the Beatles dataset [23], we found that when a DNN is used for phase regression, it often fails to decrease the estimation loss without careful pretraining. The beat phase is reset to zero at a beat frame and linearly increases to 2π until the next beat frame, *i.e.*, the phase sequence forms a sawtooth wave (Fig. 1). The phase resolution is set to $2\pi/K$, *i.e.*, the phase is quantized into K classes.

Let $\mathbf{Z}^b \triangleq \{\mathbf{z}_t^b\}_{t=1}^T$ be a sequence of beat phases, where $\mathbf{z}_t^b \in \{0, 1\}^K$ is a K -dimensional one-hot vector at frame t whose k -th element z_{tk}^b takes one when the beat phase z_t^b satisfies $\frac{2\pi(k-1)}{K} \leq z_t^b < \frac{2\pi k}{K}$. Let $\mathbf{Z}^d \triangleq \{\mathbf{z}_t^d\}_{t=1}^T$ be a sequence of downbeat phases defined in the same way. Hereafter, $*$ is denoted as b or d . In practice, we use a *blurry* version of \mathbf{Z}^* as target data for training a DNN-based classifier. More specifically, when $z_{tk}^* = 1$, we assume that $z_{t,k\pm 1}^* = 0.75$, $z_{t,k\pm 2}^* = 0.50$, and $z_{t,k\pm 3}^* = 0.25$.

Let $\boldsymbol{\psi}^* \triangleq \{\boldsymbol{\psi}_t^*\}_{t=1}^T$ be a sequence of class probability vectors estimated by the DNN, where $\boldsymbol{\psi}_t^* \in [0, 1]^K$ is a K -dimensional normalized vector of frame t . The DNN is trained in a supervised manner such that it maximizes the posterior probability of \mathbf{Z}^* given by

$$\mathcal{J}_{\text{phase}}^* = \frac{1}{T} \sum_{t=1}^T \sum_{k=1}^K z_{tk}^* \log \psi_{tk}^*. \quad (1)$$

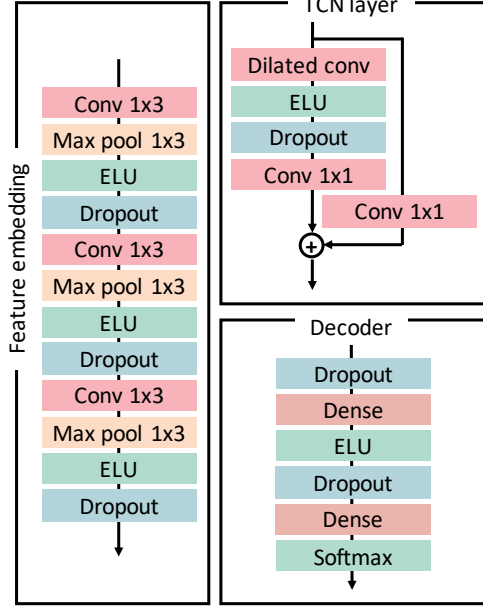


Figure 3. Network architectures.

The overall structure of the proposed method is shown in Fig. 2 and the detailed architecture of the DNN is shown in Fig. 3. The DNN used for phase classification in this study is basically the same as one used in the latest study [7] except that skip connections in the TCN used for tempo estimation are removed. It takes as input the log-magnitude spectrogram of a music signal on a logarithmic frequency axis, which is fed to the feature extraction layer referred to as *Feature embedding* in Fig. 2. The extracted feature vectors with 20 channels at each frame are fed to a stack of eleven TCN layers, which is referred to as *TCN layer* $\times 11$, followed by *Decoder* that outputs a sequence of beat or downbeat phases. For the K-class outputs, we slightly modify the components of *Decoder* as in Fig. 3. One can refer in particular to [9] for detailed descriptions of the other DNN components.

3.2 DBN-Based Beat/Downbeat Detection

We modify the existing dynamic Bayesian network (DBN) used in [13] for detecting beat and downbeat times from a noisy sequence of the estimated beat and downbeat phases. The main modification lies in the change of the observed variable of the DBN from the presence probabilities to the phases.

3.2.1 State Space

For each frame t , we represent the tempo S_t^v as the number of frames per beat, $S_t^v \in \{s_{\min}^v, s_{\min}^v + 1, s_{\min}^v + 2, \dots, s_{\max}^v\}$ ($S_t^v \in \mathbb{Z}$) where s_{\min}^v and s_{\max}^v are calculated as follows:

$$s_{\min}^v = \left\lceil \frac{60 \times \text{fps}}{\text{BPM}_{\max}} \right\rceil, \quad s_{\max}^v = \left\lfloor \frac{60 \times \text{fps}}{\text{BPM}_{\min}} \right\rfloor, \quad (2)$$

where BPM_{\min} (BPM_{\max}) indicates the minimum (maximum) BPM, fps indicates the number of frames per second, and $\lceil x \rceil$ denotes the closest integer to x . Let BPB be the number of beats per measure, and $N_t = \text{BPB} \times S_t^v$ is

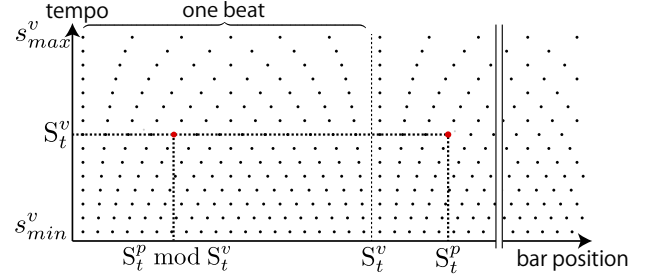


Figure 4. Two-dimensional representation of hidden state space at frame t used in DBN.

the number of frames per measure. Let $S_t^p \in \{1, 2, \dots, N_t\}$ be the position in a measure at frame t , and $\mathbf{S} \triangleq \mathbf{S}_{1:T} = (S_{1:T}^p, S_{1:T}^v)$ be a sequence of hidden states. The hidden states at frame t is shown in Fig 4

3.2.2 State Transition Model

We use the same transition model as [13], where the transition probabilities are computed as follows:

$$p(\mathbf{S}_t | \mathbf{S}_{t-1}) = p(S_t^p, S_t^v | S_{t-1}^p, S_{t-1}^v) \\ = p(S_t^p | S_{t-1}^p, S_{t-1}^v) p(S_t^v | S_t^p, S_{t-1}^v). \quad (3)$$

The first term of (3) represents a bar transition model, which is defined as

$$p(S_t^p | S_{t-1}^p, S_{t-1}^v) \\ = \begin{cases} 1 & (S_t^p - 1 \equiv S_{t-1}^p \pmod{N_{t-1}}); \\ 0 & (\text{otherwise}). \end{cases} \quad (4)$$

The second term of (3) represents a tempo transition model and the tempo is only allowed to change at beat times. $p(S_t^v | S_t^p, S_{t-1}^v)$ is defined as follows: if $S_t^p \in \mathcal{B}$,

$$p(S_t^v | S_t^p, S_{t-1}^v) = \exp\left(-\lambda \times \left| \frac{S_t^v}{S_{t-1}^v} - 1 \right|\right), \quad (5)$$

otherwise

$$p(S_t^v | S_t^p, S_{t-1}^v) = \begin{cases} 1 & (S_t^v = S_{t-1}^v); \\ 0 & (\text{otherwise}), \end{cases} \quad (6)$$

where \mathcal{B} is the set of positions that corresponds to beats and $\lambda \in \mathbb{Z}_{\geq 0}$ is the parameter to determine the steepness of the above distribution.

3.2.3 Observation Model

We formulate an observation model that stochastically generates an acoustic feature sequence \mathbf{X} from a latent state sequence \mathbf{S} . We use the output of the DNN as the probability distribution of a certain phase sequence \mathbf{Z}^* given the acoustic features \mathbf{X} as follows:

$$p(\mathbf{Z}^* | \mathbf{X}) = \prod_{t=1}^T \prod_{k=1}^K (\psi_{tk}^*)^{z_{tk}^*}. \quad (7)$$

We represent \mathbf{Z}^b and \mathbf{Z}^d together as $\mathbf{Z} = (\mathbf{Z}^b, \mathbf{Z}^d)$. To compute $p(\mathbf{X} | \mathbf{S})$ using $p(\mathbf{Z} | \mathbf{X})$, $p(\mathbf{X} | \mathbf{S})$ is transformed as

$$p(\mathbf{X} | \mathbf{S}) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} | \mathbf{S}) = \sum_{\mathbf{Z}} p(\mathbf{X} | \mathbf{Z}, \mathbf{S}) p(\mathbf{Z} | \mathbf{S}). \quad (8)$$

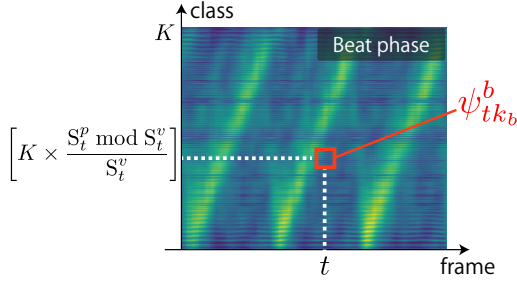


Figure 5. Observation probabilities $p(\mathbf{X}_t|\mathbf{S}_t)$ are represented as product of $\psi_{tk_b}^b$ and $\psi_{tk_d}^d$.

Let \mathbf{Z}_S be the beat and downbeat series \mathbf{Z} corresponding to the given latent state \mathbf{S} . Because \mathbf{Z}_S is uniquely determined by \mathbf{S} , $p(\mathbf{X}|\mathbf{S})$ can be calculated as follows:

$$p(\mathbf{Z}|\mathbf{S}) = \delta_{\mathbf{Z}, \mathbf{Z}_S}, \quad (9)$$

$$p(\mathbf{X}|\mathbf{S}) = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z}, \mathbf{S}) \delta_{\mathbf{Z}, \mathbf{Z}_S} = p(\mathbf{X}|\mathbf{Z}_S). \quad (10)$$

Because $p(\mathbf{Z}_S)$ is a uniform distribution and the term $p(\mathbf{X})$ is negligible, we get

$$p(\mathbf{X}|\mathbf{Z}_S) = \frac{p(\mathbf{Z}_S|\mathbf{X})p(\mathbf{X})}{p(\mathbf{Z}_S)} \propto p(\mathbf{Z}_S|\mathbf{X}). \quad (11)$$

Finally, $p(\mathbf{X}|\mathbf{S})$ can be written using the estimated probabilities as follows:

$$p(\mathbf{X}|\mathbf{S}) \propto p(\mathbf{Z}_S|\mathbf{X}) = \prod_{t=1}^T \psi_{tk_b}^b \psi_{tk_d}^d, \quad (12)$$

$$k_b = \left\lceil K \times \frac{S_t^p \bmod S_t^v}{S_t^v} \right\rceil, \quad (13)$$

$$k_d = \left\lceil K \times \frac{S_t^p}{\text{BPB} \cdot S_t^v} \right\rceil = \left\lceil K \times \frac{S_t^p}{N_t} \right\rceil. \quad (14)$$

3.3 Tempo Estimation

Global tempo is computed from the beat phases estimated by the DNN using the DFT. The beat phase series is firstly converted into a sinusoidal curve $\mathbf{Y} \triangleq \{y_t\}_{t=1}^T$ as follows:

$$y_t = \sin\left(\frac{2\pi}{K} \times \arg \max_{1 \leq k \leq K} \psi_{tk}^b\right). \quad (15)$$

The DFT is applied to the series \mathbf{Y} and we can calculate the global tempo V as follows:

$$V = \frac{60 \omega_{\max} \times \text{fps}}{2\pi} [\text{beats}/60[\text{s}], \quad (16)$$

where $\omega_{\max} [\text{rad}/\text{frame}]$ denotes the angular velocity with the largest absolute value of the Fourier coefficient in the DFT result. We can thus compute the most plausible tempo from a noisy beat phase sequence estimated by the DNN.

3.4 Joint Beat and Downbeat Estimation

We compare three architectures for joint estimation of beat and downbeat phases (Fig. 6). In the first architecture, we add another *Decoder* for downbeat estimation to the DNN described in Section 3.1. The components of the added decoder are the same as those used for beat estimation. The

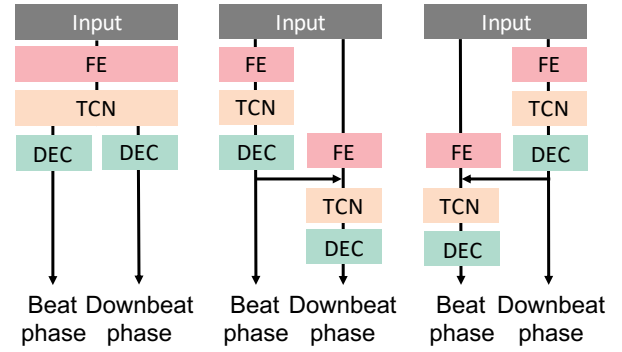


Figure 6. Three network architectures for joint beat and downbeat estimation. FE, TCN, and DEC correspond to “Feature embedding”, “TCN layer $\times 11$ ”, and “Decoder” in Fig. 2, respectively.

beat and downbeat phases are estimated in parallel. In the second architecture, the DNN described in Section 3.1 estimates beat or downbeat phases and then another DNN with the same architecture estimates the other, where the output of the former DNN in addition to the acoustic features are fed to the latter DNN. The third architecture estimates the beat and downbeat phases in the reverse order of the second architecture.

In the second architecture, when the output of the former DNN is fed to the latter DNN, the output probability series ψ^* is converted into the phase series $\hat{\mathbf{Z}}^* \triangleq \{\hat{z}_t^*\}_{t=1}^T$ as follows:

$$\hat{z}_t^* = \frac{2\pi}{K} \times \mathbf{a}^T \text{Gumbel-softmax}(\psi_t^*), \quad (17)$$

where $\mathbf{a} = [1, \dots, K]^T$ is a K -dimensional index vector and $\text{Gumbel-softmax}(\psi_t^*)$ is a differentiable function that samples a random variable \hat{z}_t^* that follows a discrete distribution ψ_t^* . We concatenate the phase sequence $\hat{\mathbf{Z}}^*$ with the 20-dimensional vector obtained from the *Feature embedding* of the second DNN and input the 21-dimensional vector into the main *TCN layer* of the second DNN. Calculating the phase series in a differentiable state enables us to back-propagate the two DNNs at the same time.

4. EVALUATION

This section reports experiments conducted for validating the effectiveness of the proposed phase-aware DNN training and comparing the performances of the three multi-task learning architectures.

4.1 Experimental Conditions

To increase the amount of training data with various tempos, each song was pitch-shifted by -12 , -6 , $+6$, and $+12$ semitones and time-stretched by min_rate , $(\text{min_rate} + 1) / 2$, $(\text{max_rate} + 1) / 2$, and max_rate times, where min_rate and max_rate are given by

$$\text{min_rate} = \frac{\text{BPM}_{\min}}{\text{bpm}}, \quad \text{max_rate} = \frac{\text{BPM}_{\max}}{\text{bpm}}, \quad (18)$$

Method	F-measure	CMLt	AMLt
<i>RWC</i>			
Baseline	0.835	0.717	0.85
Proposed	0.846	0.765	0.861
<i>Beatles</i>			
Baseline	0.798	0.69	0.777
Proposed	0.806	0.729	0.798
<i>SMC</i>			
Baseline	0.502	0.214	0.424
Proposed	0.428	0.312	0.39
<i>RWC</i>			
Baseline	0.87	0.745	0.909
Proposed	0.883	0.787	0.901
<i>Beatles</i>			
Baseline	0.847	0.763	0.866
Proposed	0.834	0.748	0.819
<i>SMC</i>			
Baseline	0.569	0.466	0.621
Proposed	0.538	0.422	0.565

Table 1. The performances of beat estimation. The upper half of the table is the result using peak-picking and the lower half using the DBN as a post-processing step.

where bpm is the original tempo and $BPM_{\max} = 215$ and $BPM_{\min} = 55$ were used. The log-magnitude spectrogram was obtained by short-time Fourier transform (STFT) with a window size of 2048 and a hop length of 441 (100 frames per second) and then transformed into the log-frequency scale consisting of 81 bins from 30 to 17,000 Hz.

The DNN was trained with Adam optimizer [24]. The batch size was 1. The learning rate was initialized to 1×10^{-3} and then halved gradually if the validation loss did not improve for 15 epochs. The training was terminated when the validation loss did not improve for 50 epochs or when 200 epochs were finished. To prevent gradient explosion, gradients greater than 0.5 were clipped. The kernel size of the dilated convolutions was set to 5 and the dropout rate was set to 0.1. The phase value was quantized into $K = 360$ classes. Other parameters are shown in Fig. 3. In the DBN-based post-processing, the maximum and minimum BPMs were the same as those in the data augmentation, and the λ was set to 100.

We separately conducted eight-fold cross validations on the RWC Popular Music dataset [25] and the Beatles dataset [23]. Although our main target is popular music, we also used the SMC dataset [26], which contains various genres such as classical, blues, and film music. As in [7], we used the F-measure, CMLt, and AMLt as evaluation metrics for beat and downbeat estimation, and Accuracy 1 and Accuracy 2 for tempo estimation. F-measure has a tolerance window of ± 70 ms around the ground-truth beats. CMLt considers a beat to be correct if its tempo and phase are within a 17.5% tolerance of those of the ground-truth beat. In addition to CMLt, AMLt allows a series of double/half and triple/third tempo of the annotated beats. Accuracy 1 considers an estimated tempo correct with a tolerance of 4% of the correct tempo, and Accuracy 2 considers correct

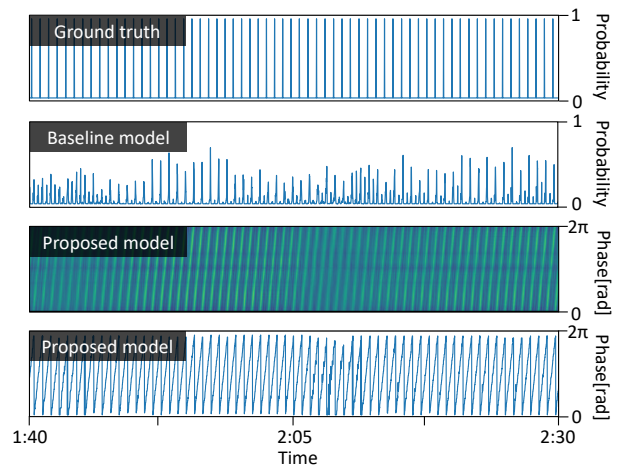


Figure 7. Estimation results for “Dear Prudence” by The Beatles. From top to bottom, the ground truth, the beat presence probability series estimated by the baseline method, the probability of each phase at each time estimated by the proposed method, and the phase series with the highest probability at each time in the third panel.

if the estimated tempo satisfies the Accuracy 1 also with tempo in double/half and triple/third target tempo.

First, we compared methods that estimate only beats. The proposed method estimates beat phases, whereas the baseline method estimates beat presence probabilities. The decoder of the baseline method was modified so that it consists of dropout, dense, and sigmoid layers as in [7–9]. We tested both a naive peak-picking method and the periodicity-aware DBN in a post-processing step that detects beat times. The peak-picking algorithm first identifies all the maxima and then selects the peaks with intervals greater than a specified horizontal distance in the order of increasing magnitude². We used 40 frames (400ms) as the distance value.

Next, we compared the three multitask learning architectures that jointly estimate beat and downbeat times followed by tempo estimation. For comparison, we implemented the existing method [7] by adding another decoder for downbeat estimation that is equivalent to the decoder for beat estimation in the baseline method, and adding the decoder for tempo estimation described in [7].

4.2 Experimental Results

Table 1 shows the performances of the proposed and baseline methods that estimate only beat times on the RWC, Beatles, and SMC datasets. When the basic peak-picking method was used for post-processing, the proposed method outperformed the baseline method, especially in terms of the CMLt by a large margin. This indicates that the proposed method better captures the periodic nature of metrical structure, resulting in the better regularity of the estimated beat times. We found that the proposed method achieved a better CMLt for not only popular music (RWC

² We use the peak-picking function specified here: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

	<i>Beat</i>			<i>Downbeat</i>			<i>Tempo</i>	
	F-measure	CMLt	AMLt	F-measure	CMLt	AMLt	Accuracy 1	Accuracy 2
<i>RWC</i>								
beat-and-downbeat	0.907	0.817	0.907	0.878	0.822	0.887	0.861	1.00
beat-to-downbeat	0.884	0.783	0.902	0.854	0.802	0.880	0.850	0.990
downbeat-to-beat	0.920	0.845	0.914	0.884	0.843	0.890	0.900	0.990
Böck <i>et al.</i> [7]	0.914	0.83	0.952	0.902	0.850	0.941	0.853	0.980
<i>Beatles</i>								
beat-and-downbeat	0.823	0.722	0.786	0.753	0.683	0.748	0.872	0.955
beat-to-downbeat	0.832	0.738	0.819	0.774	0.703	0.778	0.861	0.961
downbeat-to-beat	0.825	0.740	0.800	0.767	0.708	0.767	0.883	0.966
Böck <i>et al.</i> [7]	0.862	0.779	0.895	0.825	0.767	0.871	0.860	0.967

Table 2. The performances of joint beat and downbeat estimation. “beat-and-downbeat” denotes the architecture that simultaneously estimates beat and downbeat, “beat-to-downbeat” denotes the architecture that first estimates beats and subsequently estimates downbeats, and “downbeat-to-beat” denotes the reverse version of the “beat-to-downbeat”.

and Beatles dataset) but also various music genres (SMC dataset). When the DBN was used for post-processing, in contrast, the baseline method worked best in almost all cases except for the F-measure and CMLt on the RWC dataset. Further refinement of the DBN suitable for a sequence of phases should be investigated in the future.

Fig. 7 shows an example of the estimation results obtained by the baseline and proposed methods. While the baseline method showed high beat probabilities even at non-beat times, the proposed method estimated high probabilities at the target phases. Thus, our method has the potential to continuously estimate the phase with a constant angular velocity. This is in line with the high accuracy of CMLt in peak-picking results. However, as can be seen from the third panel in Fig. 7, the phase probabilities in the proposed method tend to be blurred in difficult segments to detect the periodicity. The baseline method, in such segments, showed high probabilities in aperiodic locations instead of blurring the beat probabilities. This difference in the output behavior in the difficult section is considered to have an effect on the DBN.

Table 2 shows the comparison of the methods used for simultaneously estimating beat, downbeat, and tempo. In the three architectures of the proposed method, “downbeat-to-beat” worked best for the RWC dataset, whereas “beat-to-downbeat” worked well for the Beatles dataset. Because downbeat estimation can be performed accurately for the RWC dataset compared with the Beatles dataset, the beat detection of “downbeat-to-beat” is considered to have the best accuracy by leveraging the downbeat estimation. By contrast, in the Beatles dataset, the estimated beat phases are considered to help improve the downbeat estimation because “beat-to-downbeat” had higher accuracy. In both datasets, “beat-and-downbeat” did not show the highest accuracy in the beat and downbeat evaluation. We consider that this is because “beat-to-downbeat” or “downbeat-to-beat” can use additional information for training and estimation. For example, in the case of “beat-to-downbeat”, the latter DNN which estimates downbeats can utilize the result of beat estimation, which is expected to improve the downbeat estimation, and the parameters of the for-

mer DNN are optimized to output the better downbeat estimation, which is expected to improve the beat estimation as well. In the results of tempo estimation, “downbeat-to-beat” showed the best accuracy in Accuracy 1. This is considered to have a relationship with the better accuracy of CMLt in our method when applying the peak-picking method since our tempo estimation method relies on the estimated phase series.

5. CONCLUSION

We proposed a phase-aware beat, downbeat, and tempo estimation method. More specifically, we trained a DNN to estimate a phase at each frame instead of a beat presence probability and calculate beat and downbeat times and tempo on the basis of the estimated phase. The experimental results showed that the proposed method could estimate more periodic beats than the conventional method that depends on a post-processing step.

For future work, we plan to utilize tempo to estimate more periodic beat times. Considering that beat and downbeat times are closely related to the other components used in MIR (*e.g.* drum scores), it would be beneficial to train an end-to-end model that directly estimates beat times and other components from music signals in the framework of multi-task learning.

6. ACKNOWLEDGEMENT

This work is funded by JST PRESTO No. JPMJPR20CB and JSPS KAKENHI Nos. 19H04137 and 20K21813.

7. REFERENCES

[1] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, “Bayesian singing transcription based on a hierarchical generative model of keys, musical notes, and F0 trajectories,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 2020, pp. 1678–1691.

- [2] R. Ishizuka, R. Nishikimi, and K. Yoshii, “Global structure-aware drum transcription based on self-attention mechanisms,” in *arXiv*, 2021.
- [3] G. Shibata, R. Nishikimi, and K. Yoshii, “Music structure analysis based on an LSTM-HSMM hybrid model,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 15–22.
- [4] G. Peeters, “Beat-marker location using a probabilistic framework and linear discriminant analysis,” in *International Conference on Digital Audio Effects (DAFx)*, 2009.
- [5] D. P. Ellis, “Beat tracking by dynamic programming,” in *Journal of New Music Research*, 2007, pp. 51–60.
- [6] M. E. Davies and M. D. Plumbley, “Context-dependent beat tracking of musical audio,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 2007, pp. 1009–1020.
- [7] S. Böck and M. E. Davies, “Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 574–582.
- [8] S. Böck, M. E. Davies, and P. Knees, “Multi-task learning of tempo and beat: Learning one to improve the other,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 486–493.
- [9] E. Matthew Davies and S. Böck, “Temporal convolutional networks for musical audio beat tracking,” in *European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.
- [10] S. Böck, F. Krebs, and G. Widmer, “Joint beat and downbeat tracking with recurrent neural networks,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 255–261.
- [11] —, “A multi-model approach to beat tracking considering heterogeneous music styles,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 603–608.
- [12] W. Nick, T. C. Ali, and J. G. Simon, “Bayesian modelling of temporal structure in musical audio,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2006, pp. 29–34.
- [13] F. Krebs, S. Böck, and G. Widmer, “An efficient state-space model for joint tempo and meter tracking,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 72–78.
- [14] S. Böck and M. Schedl, “Enhanced beat tracking with context-aware neural network,” in *International Conference on Digital Audio Effects (DAFx)*, 2011, pp. 135–139.
- [15] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” in *arXiv preprint arXiv:1803.01271*, 2018.
- [16] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” in *arXiv preprint arXiv:1609.03499*, 2016.
- [17] G. Percival and G. Tzanetakis, “Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 2014, pp. 1765–1776.
- [18] S. Böck, F. Krebs, and G. Widmer, “Accurate tempo estimation based on recurrent neural networks and resonating comb filters,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 625–631.
- [19] F.-H. F. Wu, T.-C. Lee, J.-S. R. Jang, K. K. Chang, C.-H. Lu, and W.-N. Wang, “A two-fold dynamic programming approach to beat tracking for audio music with time-varying tempo,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 191–196.
- [20] H. Schreiber and M. Müller, “A single-step approach to musical tempo estimation using a convolutional neural network,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 98–105.
- [21] H. Foroughmand and G. Peeters, “Deep-rhythm for tempo estimation and rhythm pattern recognition,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [22] M. Goto, “An audio-based real-time beat tracking system for music with or without drum-sounds,” in *Journal of New Music Research*, 2001, pp. 159–171.
- [23] M. E. Davies, N. Degara, and M. D. Plumbley, “Evaluation methods for musical audio beat tracking algorithms,” in *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.
- [24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, 2015.
- [25] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Popular, classical and jazz music databases,” in *International Society for Music Information Retrieval Conference (ISMIR)*, 2002, pp. 287–288.
- [26] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, “Selective sampling for beat tracking evaluation,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, 2012, pp. 2539–2548.

AGREEMENT AMONG HUMAN AND AUTOMATED TRANSCRIPTIONS OF GLOBAL SONGS

Yuto Ozaki¹, John McBride², Emmanouil Benetos³, Peter Q. Pfordresher⁴, Joren Six⁵, Adam T. Tierney⁶, Polina Proutskova³, Emi Sakai⁷, Haruka Kondo¹, Haruno Fukatsu¹, Shinya Fujii¹, Patrick E. Savage^{1*}

¹Keio University, Fujisawa, Japan

²Center for Soft and Living Matter, Institute for Basic Science, South Korea

³Queen Mary University of London, UK

⁴University at Buffalo, NY, USA

⁵Ghent University, Belgium

⁶Birkbeck, University of London, UK

⁷No affiliation

*psavage@sfc.keio.ac.jp

ABSTRACT

Cross-cultural musical analysis requires standardized symbolic representation of sounds such as score notation. However, transcription into notation is usually conducted manually by ear, which is time-consuming and subjective. Our aim is to evaluate the reliability of existing methods for transcribing songs from diverse societies. We had 3 experts independently transcribe a sample of 32 excerpts of traditional monophonic songs from around the world (half a cappella, half with instrumental accompaniment). 16 songs also had pre-existing transcriptions created by 3 different experts. We compared these human transcriptions against one another and against 10 automatic music transcription algorithms. We found that human transcriptions can be sufficiently reliable (~90% agreement, $\kappa \sim .7$), but current automated methods are not (<60% agreement, $\kappa < .4$). No automated method clearly outperformed others, in contrast to our predictions. These results suggest that improving automated methods for cross-cultural music transcription is critical for diversifying MIR.

1. INTRODUCTION

Cross-cultural analysis is essential to explore diversity and universality of music [1-2]. Such analyses require symbolic representations of sounds such as score notation. However, transcription into notation is usually conducted by ear, which is time-consuming and subjective [3-4].

Automated methods of music transcription and melody extraction might potentially solve these problems [5-7]. However, automated extraction of fundamental frequency (F0) alone is not sufficient. Instead, a continuous fundamental frequency must be segmented into discrete notes

with the categorical pitches and rhythms that are distinctive features of almost all the world's music [8]. This challenge is particularly important for variable pitch instruments such as the voice (the most universal instrument [8-9]). However, to our knowledge, agreement among human and automated transcription has not been objectively quantified using cross-cultural samples or multiple human transcribers.

The main objective of this paper is to evaluate the degree of agreement among human and automated transcriptions for a global song sample. We demonstrate that the degree of agreement between human transcriptions is substantially higher than the agreement between humans and machines. Our evaluation also reveals that no single algorithm outperforms the others, and there are no clear differences between signal-processing-based methods and data-driven methods.

2. RELATED WORK

2.1 Subjectivity of manual transcription

Manual transcription is central to musicological research, but to our knowledge, agreement among different human transcriptions of the same songs has never been objectively measured. Even qualitative evaluation is rare. A notable exception was a 1963 symposium on transcription where four leading ethnomusicologists independently transcribed a single recording (“A Hukwe* song with musical bow”), resulting in “four rather different transcriptions” [1, 4]. In contrast, List compared transcriptions of three songs (“Rumanian carol”, “Yiddish lullaby”, “Thai lullaby”) by between 2-9 transcribers and concluded that “transcriptions made by ear in notated form are sufficiently accurate, sufficiently reliable to provide a valid basis for analysis” [3]. More recently, Mehr et al. [9] combined transcriptions by 3 experts of 118 diverse traditional songs into a single set of “consensus” transcriptions, and had 10 experts rate their accuracy on a subjective scale from 1 (“Terrible”) to 8 (“Perfect”), finding a median rating of 6 (“Very accurate”). Yet none of these studies provided an objective measurement of the degree of agreement between individual transcribers.



© Yuto Ozaki, John McBride, Emmanouil Benetos, Peter Q. Pfordresher, Joren Six, Adam T. Tierney, Polina Proutskova, Emi Sakai, Haruka Kondo, Haruno Fukatsu, Shinya Fujii, Patrick E. Savage. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yuto Ozaki, John McBride, Emmanouil Benetos, Peter Q. Pfordresher, Joren Six, Adam T. Tierney, Polina Proutskova, Emi Sakai, Haruka Kondo, Haruno Fukatsu, Shinya Fujii, Patrick E. Savage. “Agreement among human and automated transcriptions of global songs”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

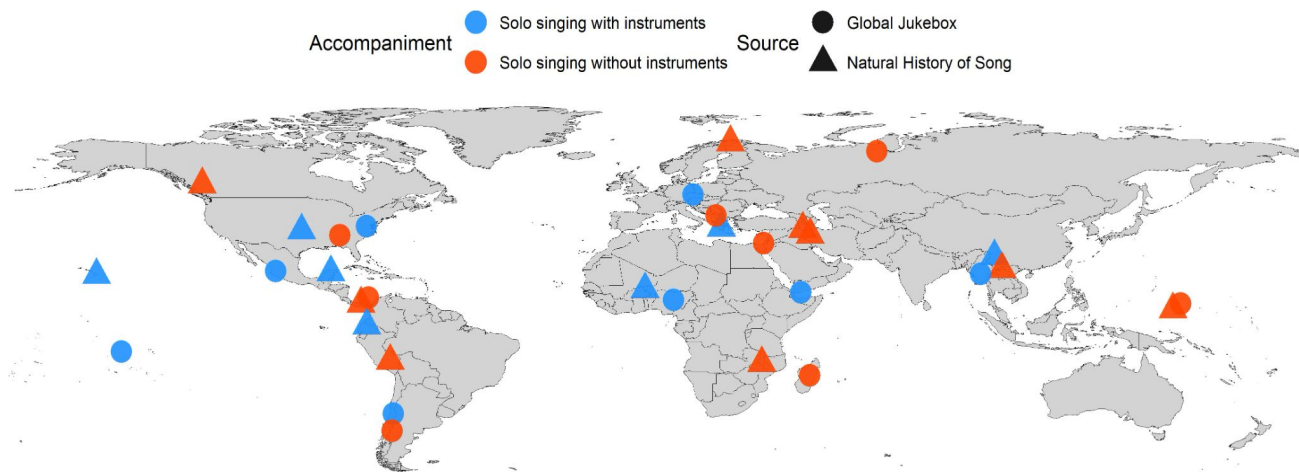


Figure 1. Map of the 32 songs transcribed and analyzed.

2.2 Reliability of automated transcription

Automatic transcription reliability has been evaluated extensively for piano music and some other genres of Western music, but rarely for non-Western music. Ycart et al. [10] evaluated the performance of four automated transcription systems against perceptual ratings from 186 participants over 153 examples of piano music taken from the MAPS dataset of MIDI-aligned piano recordings [11]. They found an average Fleiss’ Kappa coefficient of 0.59, or “borderline between moderate and substantial agreement” on participant ratings. Holzapfel and Benetos asked 16 musicologists from 3 European universities to transcribe 8 excerpts of sousta, a traditional Greek instrumental dance genre, either from scratch or starting from an automatic transcription, finding “no quantitative advantage of using [automatic music transcription]” [12]. Although computer-assisted transcription studies exist [13], recent reviews by musicologists argued that computational tools for musical analysis are either useful for only low-level analysis or not widely adopted within mainstream musicology [14-15]. Overall, there is a clear need for objective measurements of agreement among automated and human transcriptions for a cross-cultural sample of songs.

3. METHODS AND DATASET

3.1 Audio data

To examine the degree of agreement among human and automated transcriptions of diverse songs, we collected a sample of 14-second excerpts from 32 traditional songs evenly distributed across 8 geographic regions (Fig. 1). 16 songs were sampled from the publicly available 14-second excerpts of the Natural History of Song (NHS) Discography dataset [9] and manually extracted 14-second excerpts of the Global Jukebox audio files [16], respectively. We choose these datasets since they cover traditional songs from a global sample of societies. Sampling is randomly conducted using the following criteria:

- Songs are sampled equally from each of the eight regions previously used by NHS for their sampling

(i.e. 4 songs per region from North America, Oceania, etc.).

- To assess capabilities of extracting vocal melodies from instrumental accompaniment, songs are sampled to consist of half solo singing without instruments and half solo singing with instruments.

One exception is that the NHS dataset contains no audio recordings of solo singing with instruments in the Middle East region, so two solo singing excerpts without instrument examples were chosen from this region instead. We deliberately let sampled audio recordings contain various degrees of noise, reflecting the real-world challenges of analyzing traditional recordings. We did not include songs with polyphonic singing since polyphonic transcription is substantially more challenging for both humans and automated methods [5], and is beyond the scope of this study.

3.2 Automated methods

We selected 10 automatic music transcription/vocal melody extraction/pitch detection methods. We first choose methods listed in [10] as a baseline. However, that study focused on the systems designed for piano music, so we add methods designed for extracting pitch from human vocals. Considering the difference in the approach of the pitch estimation, our selection consists of automated methods from non-data-driven models and data-driven models. If the model employs a machine learning method (such as artificial neural networks) to learn model parameters from data in a training step, we call it data-driven, otherwise non-data-driven. Table 1 summarizes the selected automated methods. Regarding pYIN [17], we used the TONY [18] software to obtain its F0 estimation. Recently, several symbolic-level automatic transcription methods have been developed [19-21]. However, some models were evaluated with only MIDI synthesized sounds and were not specifically designed for singing voice, so we did not select those methods.

3.3 Transcription process

Twelve-tone equal temperament (12-TET) with A4 = 440 Hz is used to transcribe audio into staff notation by humans. Equal temperament is also applied to automated methods to standardize their outputs. As explained in the introduction, it is essential to obtain symbolic representations of pitch contours to analyze acoustic stimuli as melody. However, 12-TET is not completely appropriate since the pitch quantized into 12-TET does not always correspond to the actual scales/modes and perceptual tonal models even for Western singing, let alone non-Western [30-31].

Method	Target sound	Unit	Category
pYIN [17]	Monophonic vocal	Frame	Non data-driven: parameters specified manually
TONY [18]	Monophonic vocal	Note	
Melodia [22]	Vocal melody	Frame	
STF [23]	Multiple 12-tone ET	Frame	
CREPE [24]	Monophonic vocal	Frame	Data-driven: parameters optimized by training with datasets.
SPICE [25]	Monophonic vocal	Frame	
SS-nPNN [26]	Vocal melody	Frame	
AD-NNMF [27]	Multiple piano sound	Note	
OAF [28]	Multiple piano sound	Note	
madmom [29]	Multiple piano sound	Note	

Table 1. Summary of the selected automated methods. Unit indicates if the F0 estimation is frame-level or note-level [5] that the latter predicts onset and offset timing.

While binning continuous F0 into a simplified discrete set of 12 100-cent intervals loses information about micro-tonal nuance, 100 cents (1 semitone) is both the most commonly used system and roughly corresponds to general levels of variability in singing intonation (imprecision and inaccuracy) [31-32], making it a reasonable choice to use to evaluate accuracy. It's also what was used by Mehr et al. [9] when creating the dataset we use, enabling us to compare our results with theirs. In summary, we decide to take advantage of the convenience and comparability of 12-TET, while acknowledging that it does not capture all musical nuances.

This study focuses on the evaluation of agreement among melodies, and we discard temporal/rhythmic information so we only extract pitch from transcriptions to create a sequence of notes. However, regarding the notes representing unison melodic intervals (i.e. repeated notes), we create two transcription patterns. This is because not all selected automated methods can perform note segmentation. The change in pitch class can be used to segment two notes in the case of the other intervals, but the determining

boundary between the notes of the same pitch class would require a note segmentation algorithm.

Firstly, the raw note sequences are created as a note sequence which includes the unison interval. Based on this version, we also create a note sequence which discards repeated notes and treats the notes of the unison interval as a tied single note (i.e. “CCFGGC” becomes “CFGC”). We call this version “non-unison”. This treatment enables us to evaluate how much the pitch estimation itself, which is a baseline function of automatic transcription, determines performance. In addition, 12-TET has enharmonic equivalent pitch classes, so we only use flat notes for the same sounding sharp and flat notes.

3.3.1 Transcription by humans

We asked three Japanese experts with professional training in Western classical music to independently transcribe the 32 recordings. One of them has professional experience of transcribing non-Western music using Western staff notation. None of them had seen the transcriptions contained in the NHS dataset. They were instructed to use MuseScore3 [33] as a tool to create transcriptions. Following Mehr et al. [9]¹, we also created a consensus version of our 3 new human transcriptions. Importantly, however, while Mehr et al. only analyzed and published their consensus transcription, we include the three independent transcriptions as well as their combined consensus version to allow us to measure agreement between individual human transcribers. Our three coauthors who undertook transcription were blinded from our hypothesis testing and were asked to create transcriptions prior to discussions about coauthorship.

3.3.2 Transcription by automated methods

In order to standardize the output of each method, we apply post-processing steps including manual work, such as the quantization of frequency, smoothing of pitch contour, or the selection of melody contour by the Viterbi algorithm with manually specifying frequency range of melody for the case of multi-pitch estimation methods (cf. supplementary materials for details). Note that songs used in the evaluation contain solo singing with instrumental accompaniment but chosen methods are not designed to estimate the F0 of those styles of singing except for Melodia and SS-nPNN. Therefore the automated methods other than Melodia and SS-nPNN may include the pitch estimation of instrumental sounds, which is excluded from human transcriptions.

3.4 Sequence alignment and evaluation metrics

We use the Needleman-Wunsch algorithm [34] to align note sequences (cf. supplementary materials for further details). Agreement between two string sequences can be quantified in various ways. We mainly use Fleiss' Kappa inter-rater reliability coefficient (κ), which measures how much the observed agreement exceeds chance [35]. However, κ does not provide other relevant information such as how many notes actually differ among note sequences or whether differences are due to disagreement about note

¹ Mehr et al.'s full consensus transcriptions are published at <https://osf.io/jh7t5/>

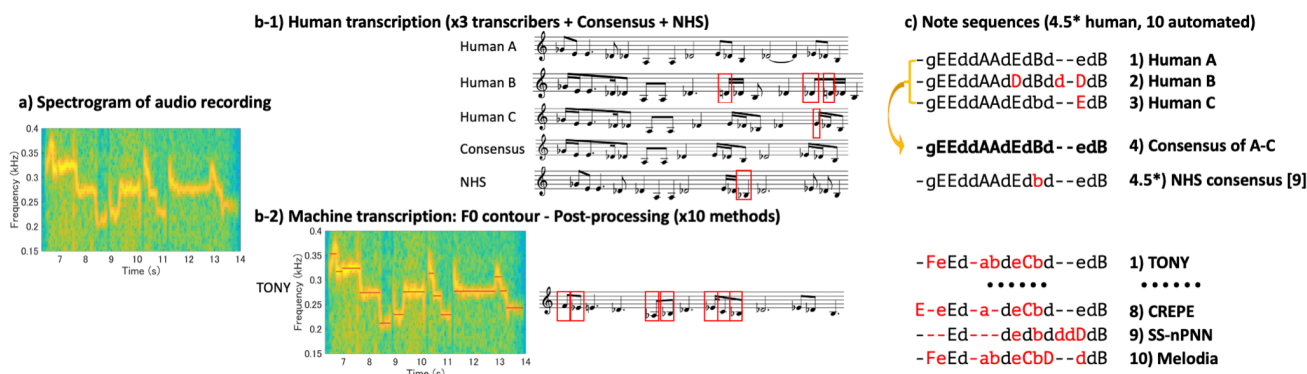


Figure 2. Overview of the agreement evaluation using an example 8-second excerpt from NAIV-075 (Healing song, Kwakwaka'wakw people, 00:06-00:14 from <https://osf.io/y29wp>). Red indicates disagreement with our new consensus transcription (#4, made by combining the three individual transcriptions #1-3). For visibility, only the automated transcription produced by TONY is shown, and octave information is omitted from the note sequences. The degree of human-human and human-machine agreement is calculated based on the note sequences (c). For example, #4.5 (NHS consensus) is 95% identical to our consensus #4 (14 out of 15 notes each), while TONY is only 48% identical (7 identical notes out of average note length of 14.5), corresponding to Fleiss' Kappa values of .94 and .34, respectively. *NB: NHS consensus transcriptions were not available for the 16 songs from the Global Jukebox sample.

pitch (i.e., substitution) or note segmentation (i.e., insertion/deletion). We also report such quantities by using percent identity (PID) [36-37] (cf. Fig. 2 for an example and for Supplementary Material for detailed explanation and additional analyses using Levenshtein distances). Although our approach did not utilize the real time information of note events, we confirmed it can still make meaningful alignment of notes as in the previous study [37] from the pilot experiment. Meanwhile, we also admit there is the technical difficulty of the extraction of note event timing (i.e. segmentation of sounds). Importantly, our work differs from the related studies evaluating the agreement between human and automated methods' melody annotation [38-39] in aiming symbolic-level note comparison rather than frame-level F0 comparison. In addition, previous studies only reported individual metrics (e.g. either distance-like metrics or inter-rater reliability, but not both), while our study explored agreement of symbolic-level melody using multiple metrics.

3.5 Transposition

We applied transposition in the note sequence alignment process to exclude the effect of disagreement by the discrepancy of the key when calculating κ , PID and Levenshtein distance. The transposition interval was searched from -2 semitones to +2 semitones. For human-human transcription comparison, the transposition was performed to maximize PID. Regarding the human-machine transcription comparison, the transposition interval was searched to maximize the average PID of all 10 human-machine pairs for each song and each human transcriber.

4. HYPOTHESES

We pre-registered² the following two primary hypotheses and 10 corresponding predictions based on pilot analysis of 4 songs not included in our main analyses:

H1: Human transcriptions are sufficiently reliable. This predicts a Fleiss' Kappa coefficient significantly greater than 0 when comparing our consensus transcription against the consensus transcriptions of Mehr et al. [9]. Note sequences including unison intervals are used.

H2: TONY is the most reliable method of automated singing transcription. We predicted this because unlike other methods TONY was designed to perform note segmentation for human vocal melody, better matching human standards for transcription. This predicts that Fleiss' Kappa comparing TONY with our consensus note sequences will be significantly greater than for the other 9 algorithms when evaluated against the note sequences including unison intervals.

5. RESULTS

5.1 Q1. To what degree do humans' transcriptions agree?

The left-hand side of Figure 3 shows inter-rater reliability and percent identity results comparing human transcribers. As predicted, there was significant agreement between our new consensus transcriptions and the pre-existing NHS consensus transcriptions (median $\kappa = .74$, $p < .001$; median PID = 88%). When we compare the results using individual transcriptions rather than consensus transcriptions, we see that agreement is slightly lower but still relatively high (lowest median κ of .64 and PID 83% for Transcribers A & B). The left-most two boxes show that individual vs. consensus yields higher agreement than individual vs. individual combinations (e.g. A-Cons, A-B, A-C) for all

² <https://osf.io/bjemd>

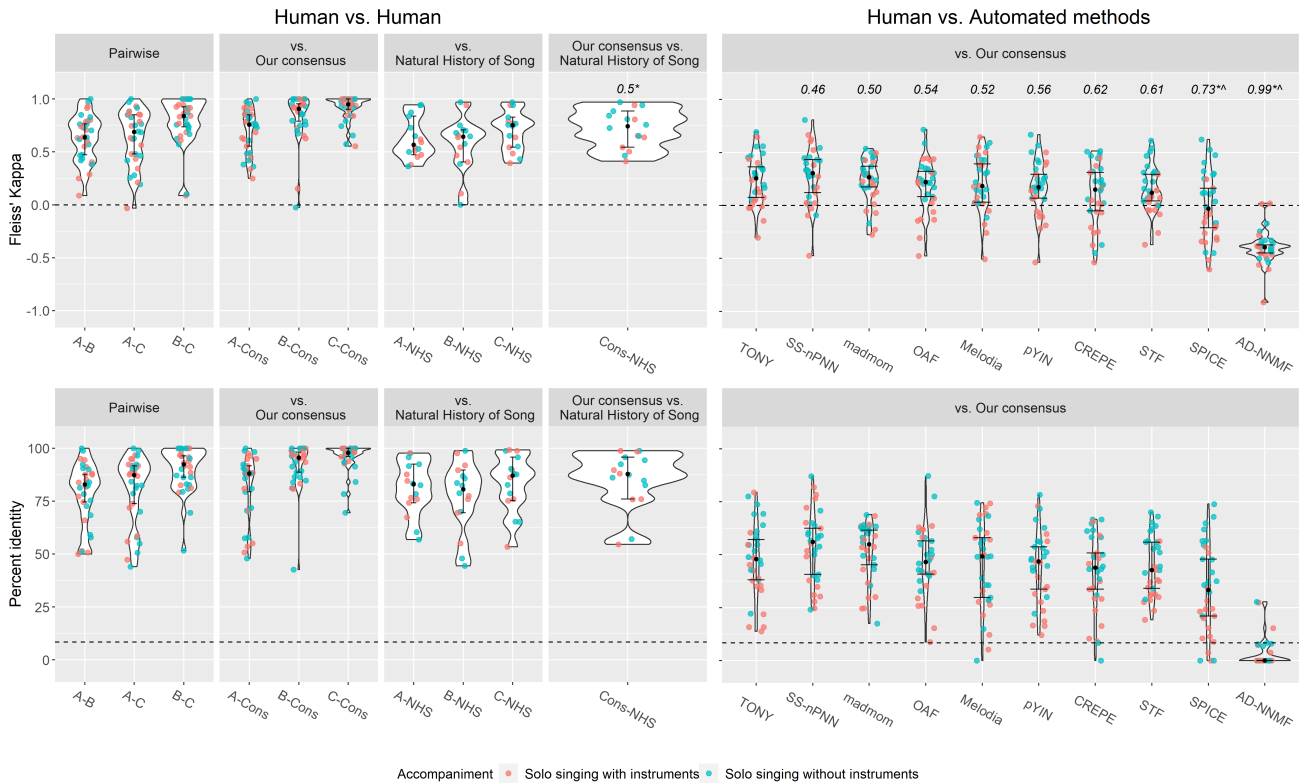


Figure 3. Agreement among human-human and human-machine transcribed note sequences. “A”, “B”, and “C” represent the three individual transcribers. The dashed line at $\kappa = 0$ and 8.3% identity indicates chance levels of agreement. The numbers appearing above the violin plots indicate effect sizes for our 10 pre-registered predictions, and * and ^ indicate significant p-value and posterior probability, respectively (cf. text for details). Black circles indicate medians, and bars represent 95% confidence intervals of the median [40]. Results using alternative transcription methods, and full p-value and posterior probabilities are available in the supplementary material (figure S3-S7 and table S2-S3).

transcribers. This means that consensus is indeed reflecting the elements of those three transcribers' transcriptions rather than the particular pairs. These results suggest that transcription of pitch contour could be reliable even for non-Western music.

We also analyze some low agreement results. There are 25 pairwise κ values lower than 0.4, all of which involved 7 songs (NAIV-033, NAIV-100, NAIV-117, T5431R27, T5482R03, NAIV-015 and NAIV-048). In particular, NAIV-033 (Maya healing song) is a near-monotone chant, and so the degree of agreement by chance is so large that it negates the proportion of agreement. As the unison interval dominates, the note sequences of this song are highly homogeneous ($PID > 0.9$ for all 6 pairs). Other than this song, the remaining disagreement is mainly caused by disagreement of the pitch rather than segmentation. (cf. supplementary material table S1). In other words, transcribers generally captured the same note events, but the assigned pitch sometimes differs by 1-2 semitones.

Incidentally, we observed that using raw note sequences yielded the median of κ very close to zero ($\kappa = -0.019$) due to cases where the tonal center differed by a semitone (and sometimes a whole tone, cf. supplementary material figure S1-S2 for a sample figure and the results). Therefore, our evaluation actually focused on whether relative pitch, or the shape of the pitch contour, matches between transcribers.

5.2 Q2. Which automated method agrees best with transcription of non-Western music by humans?

The right-hand side of Figure 3 shows κ and PID obtained by comparing the machine note sequences and our consensus version's note sequences. Contrary to our prediction, there is no evidence for the superiority of TONY except when compared with AD-NNMF and SPICE. The figure also indicates generally low reliability of automated transcription methods (median κ values are all below 0.4). In particular, SPICE and AD-NNMF both had median κ below 0, suggesting they performed worse than chance. Especially, AD-NNMF failed to pick up notes correctly in many cases and indeed, sometimes the length of note sequence of AD-NNMF is zero (cf. supplementary material figure S8-S9). In such cases, the proportion of agreement between human note sequences also becomes around zero, but chance agreement probability is still positive by its definition, resulting in many negative Kappa values.

In addition, SPICE and CREPE had difficulty estimating F0 of the particular tracks of monophonic singing, which is apparent from the drop in the plot of note sequence length (cf. supplementary material figure S8-S9). As predicted, κ of automated methods designed for monophonic vocal melody (i.e. TONY, pYIN, CREPE and SPICE) show a relatively large difference dependent on instrumental accompaniment compared to the other methods,

but STF also suffered from instrumental sounds. (cf. Figure 3 and supplementary material figure S10).

See supplementary material for additional analysis details including results of measuring agreement with Levenshtein distances (which were generally similar to results found using PID).

6. DISCUSSION AND FUTURE WORK

Overall, we observed that the degree of agreement of transcriptions of diverse traditional songs among human transcribers was relatively high ($\sim 90\%$ agreement, $\kappa \sim 0.7$; Fig. 3), while the degree of agreement between human and automated methods was relatively low ($< 60\%$ agreement, $\kappa < .4$; Fig. 3). Automated methods where less than 60% of estimated notes agree with human judgments are unlikely to produce satisfactory results for the kinds of tasks we hope to use them for, such as cross-cultural comparison of scale and interval systems [41-42]. Landis and Koch [43] suggested that κ of .61-.8 be considered "substantial" agreement .21-0.4 as "fair" agreement, but some have suggested that less than .4 is unacceptably low [44]. Our qualitative examination of the transcriptions (e.g., Fig. 2) supports the interpretation that human transcriptions of diverse traditional songs can be sufficiently reliable, but current state-of-the-art automated methods are not. However, high agreement does not necessarily equate to high quality. The quality of transcription may depend on its goal [45-46], so future research should expand on our results to evaluate transcriptions for specific applications (e.g., tonal analysis [41-42]).

Different combinations of human transcribers and songs had varying levels of agreement, but overall the agreement among three female Japanese experts and the consensus transcription by three white American male experts was surprisingly high, with more differences appearing between individuals than between the two groups. Of course, by definition the experts had been trained in Western music and transcription methods - future studies should explore perceptual variability among listeners with varying degrees of training in different musical systems [47].

Disagreement among humans appeared to primarily involve assignment of pitch to different pitch classes. In contrast, disagreement in automated methods appeared to primarily reflect segmentation, rather than F0 estimation. Future studies might be able to clarify this point by collecting both F0 annotations and score transcriptions by humans. This might also allow us to compare our results with more conventional metrics used in research on pitch estimation algorithms such as frame-wise and note-wise F0 agreement and the use of true positive and false positive scores [10] (though we emphasize that our work brings into question the idea that a single 'ground-truth' annotation might even exist that all can agree on for diverse songs). [48] developed a method for evaluating the degree of agreement of F0-estimates among multiple automated methods, and such a method would be especially advantageous to assess the overall reliability of automated methods against global songs once F0 annotation is collected.

We were surprised that all automated methods performed so poorly even for the relatively simple task of transcribing only pitch sequences for monophonic songs.

Some might argue it is unfair to evaluate MIR methods designed primarily for F0 transcription of Western instrumental music using symbolic notation transcriptions of (mostly) non-Western songs. Our feelings are somewhat opposite - it is unfair and unethical to limit MIR to a narrow slice of the world's music [49]. Since our goal was to evaluate the ability of existing MIR algorithms to transcribe global songs using symbolic notation, we believe it is fair and necessary to evaluate state of the art algorithms even though - in fact especially because - they were not designed for this application. Our results thus confirm the strong need for automatic music transcription and other MIR tasks to expand algorithms and datasets beyond the traditional focus on Western classical and popular music to be suitable for more diverse musical styles [49]. Moving from a reliance on convenient but restricted datasets (e.g., the MAPS dataset of MIDI-aligned piano recordings commonly used to evaluate automatic transcription [11]) to cross-cultural datasets like the one presented here and elsewhere [9, 16, 50] will be essential for diversifying MIR.

The formalization of a general algorithm that agrees with human pitch recognition and note segmentation is an ongoing challenge related to a central issue in MIR: the "correctness" of the algorithm depends on the degree of perceptual variability in the human ground-truth data [51]. Thus, accounting for diversity and subjectivity in human transcriptions is equally critical to advance research on the automatic analysis of music. For example, while we found relatively high agreement among expert transcribers using Western 12-TET notation, we do not know whether the singers whose songs we transcribed would agree with our transcriptions, or whether transcription using a different notation system (e.g., Middle Eastern 24-note microtonal notation, 'Are'Are 7-note equiheptatonic notation [52], Killick's "global notation" [53], etc.) would give better or worse results. We see our current results using 12-TET - with all its known problems and cultural baggage [1-5, 45-46] - as a baseline against which future studies can test whether other methods of cross-cultural transcription may be able to improve.

Furthermore, here we solely focused on pitch, but a more comprehensive description of music necessitates other dimensions such as rhythm, timbre, and social context [54]. Other cross-cultural systems of music analysis such as Cantometrics [54-55] and CantoCore [56] have been designed to capture such features. Somewhat counterintuitively, our current results show substantially higher agreement using Western staff notation to analyze a global song sample ($\kappa \sim 0.7$) than was found using these cross-cultural song classification systems ($\kappa \sim 0.3-0.5$ [8, 16, 56]). This suggests a need for MIR to better account for diversity in human ground-truth representations of all dimensions of music, not only pitch [57].

Musical diversity is a crucial challenge and opportunity for MIR. Quantifying diversity in human "ground-truth" cross-cultural data is an important first step for diversifying MIR. Our study demonstrates that there is still substantial room for improvement for automated methods of music transcription, and provides quantitative estimates of diversity among human transcriptions to help guide development of future MIR methods.

7. AUTHOR CONTRIBUTIONS

Conceptualization: P.E.S, J.M, P.Q.P., J.S., A.T.T., S.F., E.B., Y.O., P.P.; Transcription: E.S., H.K., H.F.; Methodology / Analysis / Investigation / Visualization: Y.O., P.E.S., E.B., J.M.; Project administration / Supervision / Funding acquisition: P.E.S.; Writing – original draft: Y.O., P.E.S.; Writing – review & editing: J.M., E.B., P.Q.P., J.S., A.T.T., P.P., S.F.

8. ACKNOWLEDGEMENTS

We thank Olga Velichkina and members of the Keio SFC CompMusic and NeuroMusic labs for feedback on earlier versions of this manuscript. This work was supported by Grant-in-Aid no. 19KK0064 from the Japan Society for the Promotion of Science and by startup grants from Keio University (Keio Global Research Institute, Keio Research Institute at SFC, and Keio Gijuku Academic Development Fund) to P.E.S.

9. DATA/CODE AVAILABILITY

Audio files, transcriptions (individual and consensus), aligned note sequences, and analysis scripts are available at <https://github.com/comp-music-lab/agreement-human-automated> (codes, transcriptions, sequence data), <https://doi.org/10.5281/zenodo.4941863> (audio). The PDF of the supplementary material is also available at the above GitHub link.

10. REFERENCES

- [1] B. Nettl, *The Study of Ethnomusicology: Thirty-Three Discussions, 3rd ed.*, Champaign, IL, USA: University of Illinois Press, 2015.
- [2] P. E. Savage, and S. Brown, "Toward a new comparative musicology," *Analytical Approaches to World Music*, vol. 2, no. 2, pp. 148–197, 2013.
- [3] G. List, "The reliability of transcription," *Ethnomusicology*, vol. 18, no. 3, pp. 353–377, 1974.
- [4] N. M. England, R. Garfias, M. Kolinski, G. List, W. Rhodes, and C. Seeger, "Symposium on transcription and analysis: A Hukwe* song with musical bow," *Ethnomusicology*, vol. 8, no. 3, pp. 223–233, 1964.
- [5] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [6] M. Müller, E. Gómez, and Y. Yang, Eds., "Computational methods for melody and voice processing in music recordings (Dagstuhl seminar 19052)," *Dagstuhl Reports*, vol. 9, no. 1, pp. 125–177, 2019.
- [7] Z. Rafii, A. Liutkus, F. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307-1335, 2018, doi: 10.1109/TASLP.2018.2825440.
- [8] P. E. Savage, S. Brown, E. Sakai, and T. E. Currie, "Statistical universals reveal the structures and functions of human music," *Proc. National Academy of Sciences USA*, vol. 112, no. 29, pp. 8987–8992, 2015.
- [9] S. A. Mehr et al., "Universality and diversity in human song," *Science*, vol. 366, eaax0868, 2019, doi: 10.1126/science.aax0868.
- [10] A. Ycart, L. Liu, E. Benetos, and M. T. Pearce, "Investigating the perceptual validity of evaluation metrics for automatic piano music transcription," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 68–81, 2020, doi: 10.5334/tismir.57.
- [11] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.
- [12] A. Holzapfel, and E. Benetos, "Automatic music transcription and ethnomusicology: A user study," in *Proc. 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 678-684.
- [13] E. Gómez, and J. Bonada, "Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms as applied to a cappella singing," *Computer Music Journal*, vol. 37, no. 2, pp. 73–90, 2013.
- [14] S. Cottrell, "Big music data, musicology, and the study of recorded music: Three case studies," *The Musical Quarterly*, vol. 101, no. 2-3, pp. 216-243, doi: 10.1093/musqtl/gdy013.
- [15] L. Tilley, "Analytical ethnomusicology: How we got out of analysis and how to get back in," in *Springer Handbook of Systematic Musicology*, R. Bader, Eds. Berlin, Germany: Springer, 2018, pp. 953–977.
- [16] A. L. C. Wood et al., "The Global Jukebox: A public database of performing arts and culture," *PsyArXiv Preprint*. doi: org/10.31234/osf.io/4z97j.
- [17] M. Mauch, and S. Dixon, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Florence, Italy, 2014, pp. 659–663.
- [18] M. Mauch et al., "Computer-aided melody note transcription using the tony software: Accuracy and efficiency," in *Proc. 1st International Conference on Technologies for Music Notation and Representation*, Paris, France, 2015.
- [19] R. Nishikimi, E. Nakamura, M. Goto, and K. Yoshii, "Audio-to-score singing transcription based on a CRNN-HSMM hybrid model," *APSIPA Transactions on Signal and Information Processing*, vol. 10, no. e7, pp. 1-13, 2021, doi: 10.1017/ATSIP.2021.4.

- [20] R. Gunter, C. Carvalho, and P. Smaragdis, "Towards end-to-end polyphonic music transcription: Transforming music audio directly to a score," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New York, USA, 2017, pp. 151-155, doi: 10.1109/WASPAA.2017.8170013.
- [21] M. A. Román, A. Pertusa, and J. Calvo-Zaragoza, "A holistic approach to polyphonic music transcription with neural networks," in *Proc. 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 731-737.
- [22] J. Salamon, and E. Gomez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759-1770, 2012, doi: 10.1109/TASL.2012.2188515.
- [23] L. Su, and Y. Yang, "Combining spectral and temporal representations for multipitch estimation of polyphonic music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 10, pp. 1600-1612, 2015, doi: 10.1109/TASLP.2015.2442411.
- [24] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proc. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, Calgary, AB, Canada, 2018, pp. 161-165, doi: 10.1109/ICASSP.2018.8461329.
- [25] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, "SPICE: Self-supervised pitch estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1118-1128, 2020, doi: 10.1109/TASLP.2020.2982285.
- [26] W.-T. Lu, and L. Su, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," in *Proc. 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 521-528.
- [27] T. Cheng, M. Mauch, E. Benetos, and S. Dixon, "An attack/decay model for piano transcription," in *Proc. 17th International Society for Music Information Retrieval Conference*, New York, NY, USA, 2016, pp. 584-590.
- [28] C. Hawthorne et al., "Onsets and Frames: Dual-objective piano transcription," in *Proc. 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 50-57.
- [29] S. Bock, F. Korzeniowski, J. Schluter, F. Krebs, and G. Widmer, "Madmom: A new Python audio and music signal processing library," in *Proc. 24th ACM International Conference on Multimedia*, Amsterdam, Netherlands, 2016.
- [30] R. Ambrazevičius, "The perception and transcription of the scale reconsidered: Several Lithuanian cases," *The World of Music*, vol. 47, no. 2, pp. 31-53, 2005.
- [31] P. Q. Pfordresher, S. Brown, K. M. Meier, M. Belyk, and M. Liotti, "Imprecise singing is widespread," *The Journal of the Acoustical Society of America*, vol. 128, no. 4, pp. 2182-2190, 2010.
- [32] P. Larrouy-Maestri, Y. Lévêque, D. Schön, A. Giovanni, and D. Morsomme, "The evaluation of singing voice accuracy: A comparison between subjective and objective methods," *Journal of Voice*, vol. 27, no. 2, pp. 259.e1-259.25, 2013, doi: 10.1016/j.jvoice.2012.11.003.
- [33] MuseScore: <https://musescore.org/ja> (accessed Feb. 25, 2021).
- [34] S. B. Needleman, and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443-453, 1970.
- [35] K. L. Gwet, *Handbook of Inter-Rater Reliability. The Definitive Guide to Measuring the Extent of Agreement Among Raters, 4th edition*. Gaithersburg, MD, USA: Advanced Analytics, LLC., 2015.
- [36] A. C. W. May, "Percent sequence identity: The need to be explicit," *Structure*, vol. 12, no. 5, pp. 737-738, 2004, doi: 10.1016/j.str.2004.04.001.
- [37] P. E. Savage, and Q. D. Atkinson: "Automatic tune family identification by musical sequence alignment," in *Proc. 16th International Society for Music Information Retrieval Conference*, Málaga, Spain, 2015, pp. 162-168.
- [38] J. J. Bosch, and E. Gómez, "Melody extraction in symphonic classical music: a comparative study of mutual agreement between humans and algorithms," in *Proc. 9th Conference on Interdisciplinary Musicology*, Berlin, Germany, 2014.
- [39] S. Balke, J. Abeßer, J. Driedger, C. Dittmar, and M. Müller, "Towards evaluating multiple predominant melody annotations in Jazz recordings," in *Proc. 17th International Society for Music Information Retrieval Conference*, New York City, NY, USA, 2016, pp. 246-252.
- [40] M. J. Campbell, and M. J. Gardner, "Calculating confidence intervals for some non-parametric analyses," *British Medical Journal*, vol. 296, no. 6634, pp. 1454-1456, doi: 10.1136/bmj.296.6634.1454.
- [41] J. M. McBride and T. Tlustý, "Cross-cultural data suggests musical scales evolved to maximise imperfect fifths," *arXiv Preprint*, 2020. arXiv:1906.06171.
- [42] J. Kuroyanagi et al., "Automatic comparison of human music, speech, and bird song suggests uniqueness of human scales," in *Proc. 9th International Workshop on Folk Music Analysis (FMA2019)*, Birmingham, UK, 2019, pp. 35-40.
- [43] J. R. Landis, and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159-174, 1977.

- [44] J. Sim, and C. C. Wright, "The kappa statistic in reliability studies: Use, interpretation, and sample size requirements," *Physical Therapy*, vol. 85, no. 3, pp. 257–268, 2005.
- [45] C. Seeger, "Prescriptive and descriptive music-writing," *Music. Q.*, vol. 44, no. 2, pp. 184–195, 1958.
- [46] M. Hood, *The Ethnomusicologist*. New York: McGraw-Hill, 1971.
- [47] N. Jacoby et al., "Universality and cross-cultural variation in mental representations of music revealed by global comparison of rhythm priors," *PsyArXiv Preprint*, 2021. doi: 10.31234/osf.io/b879v.
- [48] S. Rosenzweig, F. Scherbaum, and M. Müller, "Reliability assessment of singing voice F0-estimates using multiple algorithms," in *Proc. 2021 IEEE International Conference on Acoustics, Speech and Signal Processing*, Ontario, Canada, 2021, pp. 261-265.
- [49] G. Born, "Diversifying MIR: Knowledge and real-world challenges, and new interdisciplinary futures," *Trans. Int. Soc. Music Inf. Retr.*, vol. 3, no. 1, pp. 193–204, 2020.
- [50] P. E. Savage, "An overview of cross-cultural music corpus studies," in *Oxford Handbook of Music and Corpus Studies*, D. Shanahan, A. Burgoyne, and I. Quinn, Eds. Oxford University Press, in press. doi: 10.31235/osf.io/nxtbg.
- [51] A. Flexer, and T. Grill, "The problem of limited inter-rater agreement in modelling music similarity," *Journal of New Music Research*, vol. 45, no. 3, pp. 239–251, 2016, doi: 10.1080/09298215.2016.1200631.
- [52] H. Zemp, and V. Malkus, "Aspects of 'Are 'Are musical theory," *Ethnomusicology*, vol. 23, no. 1, pp. 5–48, 1979.
- [53] A. Killick, "Global notation as a tool for cross-cultural and comparative music analysis," *Analytical Approaches to World Music*, vol. 8, no. 2, pp. 235–279, 2021.
- [54] P. E. Savage, "Alan Lomax's Cantometrics Project: A comprehensive review," *Music & Science*, vol. 1, pp. 1–19, 2018.
- [55] A. Lomax, and V. Grauer, "The Cantometric coding book," in *Folk Song Style and Culture*, A. Lomax, Ed. Washington, DC, USA: American Association for the Advancement of Science, 1968, pp. 34–74.
- [56] P. E. Savage, E. Merritt, T. Rzeszutek, and S. Brown, "CantoCore: A new cross-cultural song classification scheme," *Analytical Approaches to World Music*, vol. 2, no. 1, pp. 87–137, 2012.
- [57] H. Daikoku, S. Ding, U. S. Sanne, E. Benetos, A. L. Wood, S. Fujii, and P. E. Savage: "Human and automated judgements of musical similarity in a global sample," *PsyArXiv Preprint*, 2020, doi: <https://doi.org/10.31234/osf.io/76fmq>.

AUTOMATIC RECOGNITION OF TEXTURE IN RENAISSANCE MUSIC

Emilia Parada-Cabaleiro^{1,2} Maximilian Schmitt³ Anton Batliner³
Björn Schuller^{3,4} Markus Schedl^{1,2}

¹Multimedia Mining and Search Group, Institute of Computational Perception, JKU Linz, Austria

²Human-centered AI Group, AI Lab, Linz Institute of Technology (LIT), Austria

³Chair of Embedded Intelligence for Health Care and Wellbeing, University of Augsburg, Germany

⁴ GLAM – Group on Language, Audio & Music, Imperial College London, UK

emilia.parada-cabaleiro@jku.at

ABSTRACT

Renaissance music constitutes a resource of immense richness for Western culture, as shown by its central role in digital humanities. Yet, despite the advance of computational musicology in analysing other Western repertoires, the use of computer-based methods to automatically retrieve relevant information from Renaissance music, e. g., identifying word-painting strategies such as *madrigalisms*, is still underdeveloped. To this end, we propose a score-based machine learning approach for the classification of texture in Italian madrigals of the 16th century. Our outcomes indicate that Low Level Descriptors, such as intervals, can successfully convey differences in High Level features, such as texture. Furthermore, our baseline results, particularly the ones from a Convolutional Neural Network, show that machine learning can be successfully used to automatically identify sections in madrigals associated with specific textures from symbolic sources.

1. INTRODUCTION

The ‘classical’ Italian madrigal is a secular vocal composition from the 16th century, typically for 4 to 6 vocal parts, characterised by a close relationship between music and text [1]. Due to the great historical value of madrigals for the Western cultural heritage, many initiatives aiming to preserve and investigate this repertoire through computational means have been presented, such as *The Marenzio Online Digital Edition (MODE)*¹ and the *Tasso in Music Project* [2], amongst others [3–6]. Nevertheless, in comparison to other relevant genres from Western repertoires, such as Bach’s chorales [7–9] or operas [10–12], the application of machine learning (ML) to the understanding of Renaissance music is still rare [13, 14]. Indeed, the investigation of *madrigalisms*, i. e., the word-painting strategy

¹ www.marenzio.org/index.xhtml

typical of madrigals [1], has not yet been automatised – a subject identified as of great interest [15].

In the ‘classical’ Italian madrigal, unlike the madrigal of the 17th century, the meaning of the lyrics is often expressed through textural changes. Due to the prominent role of texture in the *madrigalisms* of these specific madrigals, as a first step to approach this topic, we statistically assess which musical features are involved in different textures. Furthermore, we present baseline results for their classification. The experiments were carried out on the SEILS dataset [16], from which a variety of features related to the time, frequency, and time-frequency dimensions were extracted with the `music21` toolkit [17]. The performance of four ML models, i. e., Support Vector Machines (SVM), Multi-layer Perceptrons (MLP), Convolutional Neural Networks (CNN), and Bidirectional Long-Short Term Memory Recurrent Neural Networks (BLSTM-RNN), was evaluated for recognition of three types of texture: antiphonal (ANT), contrapuntal (CON), and homorhythmic (HOM).


The goal of our study is three-fold: (i) Identify the features characteristic of different textures through the extraction and evaluation of symbolic Low Level Descriptors and statistical functionals; (ii) initialise a research path for automatic recognition of word-painting, as a first step focussed on texture, which later should be followed by the evaluation of *madrigalisms*’ textual content; (iii) increase the interest within the ML community in applying artificial intelligence (AI) to digital humanities.

The rest of the manuscript is laid out as follows: Section 2 gives an overview of the related work; Section 3 introduces the considered repertoire; Section 4 and Section 5 outline the feature extraction and evaluation; Section 6 and Section 7 describe the experimental set-up and the ML baseline; Section 8 concludes the paper. To promote further improvements in the field, the source code which enables researchers to replicate the statistical assessment and the baseline results are freely released.²

2. RELATED WORK

With the advent of digital humanities in general and computational musicology in particular, more and more sym-

² github.com/SEILSdataset/Texture_Recognition

 © Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Björn Schuller, Markus Schedl. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Emilia Parada-Cabaleiro, Maximilian Schmitt, Anton Batliner, Björn Schuller, Markus Schedl, “Automatic Recognition of Texture in Renaissance Music”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

bolic musical corpora have been presented in the literature. Some of these are the ELVIS database,³ the *Kernscores* database [18], the MUTOPIA project⁴ database, or the *Digital Interactive Mozart Edition* [19]. The potential of preserving music in a codified syntax has prompted well-defined crowdsourcing initiatives aimed to encode and share symbolic music [20]. In parallel to these, projects focusing on the symbolic codification of Renaissance music, such as the *Tasso in Music Project* [2] have been carried out. Furthermore, given the inherent complexity of codifying early music, specific guidelines, aimed to minimise encoding inconsistencies [21] that may lead to bias in the data and therefore distort the ML outcomes [22], have been presented [23].

Nevertheless, when we consider the symbolic corpora containing annotations, these are considerably reduced [24]; therefore, systems such as *Dezrann* [25], designed to collaboratively collect analytic annotations, have been developed. Concerning Western music in general, annotated symbolic corpora have been presented, e. g., to enable the automatic analysis of harmony [24, 26] and musical structure [27, 28]. Although annotated corpora of Renaissance music have also been presented, these are still much more limited [29–31]. Similarly, computational methods aiming to investigate early music have also been developed, such as the online analysis search functionalities of the *Josquin Research Project* [32] and the SIMSSA Project [6], amongst others [14, 33, 34]. However, to the best of our knowledge, approaches to automatically extract or retrieve specific attributes typical of early music, such as *madrigalisms*, have not yet been presented.

3. DATA DESCRIPTION

The word-painting strategy used in madrigals to musically imitate the meaning of particular words is known as *madrigalism* [1]. In the ‘classical’ Italian madrigals of the 16th century, *madrigalisms* can involve rhythm or pitch but are in particular defined by changes in polyphonic texture, for example the alternation between imitative and homophonic counterpoint. Specifically, we will consider three types of texture typically associated to *madrigalisms*: antiphonal texture (ANT), i. e., alternating a musical-linguistic pattern between two parts; contrapuntal texture (CON), i. e., staggering a musical-linguistic pattern along the timeline over the different parts; and homorhythmic texture (HOM), i. e., musical-linguistic patterns occur simultaneously in the different parts. For musical examples and further details on each texture, the reader is referred to [29].

The experiments were carried out on the SEILS dataset [16], a corpus containing 30 symbolically codified madrigals from the *Il Lauro Secco* anthology. This collection is particularly suited to evaluate *madrigalisms*, since this word-painting technique is common for its composers, e. g., Luca Marenzio [1]. All the madrigals in the corpus are written for five parts: Canto, Alto, Quinto, Tenor, and Basso, from the higher to the lower. The modern notated

Group	LLD	Description
<i>Time</i>	BEAT	Note’s position (parsed into time-signature units)
	OFFSET	Note’s position (parsed into crotchet units)
	RHYTHM	Note’s rhythm (as a fraction of crotchet notes)
<i>Frequency</i>	PS	Pitch space representation (e. g., 60.0 stands for C4)
<i>Time-freq.</i>	INTERVAL	Interval between two notes (expressed in semitone units)
	MUS-TEXT	Binary music-text relationship (syllabic and melismatic)

Table 1. Description of the 6 Low Level Descriptors (LLDs) and their corresponding feature groups.

transcriptions codified in `**kern` syntax were considered. Although four kinds of texture are annotated in the corpus – CON, HOM, ANT, and COMB (combined) – the COMB one, which is a combination of the previous ones, was discarded due to its ambiguity. For simplicity, from now on we will refer to the annotated sections as *madrigalisms*.

4. FEATURE EXTRACTION

The extracted features can be grouped into three classes related to three dimensions: *Time*, *Frequency*, and *Time-freq.*, i. e., the combination of the first two. This formulation relates to the ‘standard’ 2-dimensional score representation typical of written Western music, where *Time* is encoded on the *x* axis and *Frequency* (pitch in music theory) on the *y* axis, as shown in piano-rolls [35].⁵ For each dimension, specific Low Level Descriptors (LLDs), i. e., “Unambiguously defined and objectively verifiable concepts” [36], were extracted with the `python music21` [17]. Note that other formulations of LLDs in symbolic music differing from the herein considered have also been presented [37]. Subsequently, statistical functionals were computed from the LLDs (cf. Section 4.2).

4.1 Low Level Descriptors (LLDs)

For each annotated *madrigalism*, six LLDs, chosen from those most representative of each feature group, were extracted over time considering the ‘note’ as frame unit: Three LLDs relate to *Time* (BEAT, OFFSET, and RHYTHM); one to *Frequency* (PS); two to *Time-freq.* (INTERVAL and MUS-TEXT); cf. Table 1. BEAT indicates the position of each note according to the time-signature.⁶ OFFSET gives the position of each note according to crotchets (standard length unit).⁷ RHYTHM is indicated as a fraction of crotchets (represented as 1). PS (pitch space) represents absolute pitches according to the chromatic scale.⁸ INTER-

⁵ Although this representation applies to most of the Western musical notation, exceptions should be considered, e. g., contemporary notation. Note that we are not referring to the musical syntax, e. g., Humdrum.

⁶ A 3/4, i. e., a triple simple meter, would be parsed into three crotchet units; a 6/8, i. e., a binary compound meter, into two dotted crotchet units.

⁷ For coherence with respect to BEAT and to avoid biasing the features by the score length, the OFFSET was computed within bars’ boundaries.

⁸ As in MIDI, 60 stands for C4; yet, PS contemplates also microtones and values beyond 0-127, although not present in the evaluated repertoire.

³ database.elvisproject.ca/

⁴ www.mutopiaproject.org/

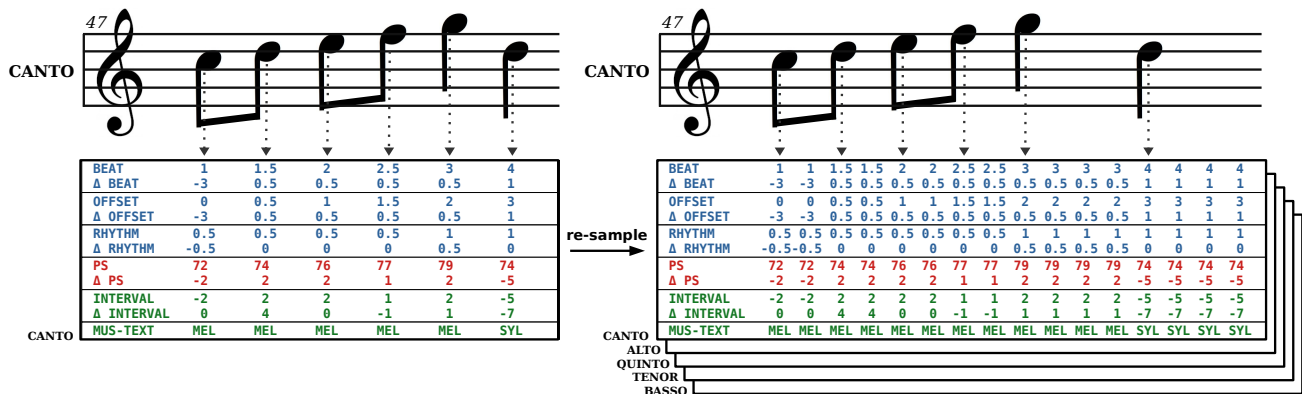


Figure 1. For each part, the Note Level Descriptor (NLD) matrix (on the left) is re-sampled and subsequently ‘assembled’ into a multi-dimensional NLD (on the right). The example corresponds to bar 47 (Canto part) of Alberti’s madrigal.

VAL indicates musical intervals in semitones, where negative values indicate downward intervals, positive upward intervals. MUS-TEXT expresses the (categorical) music-text relationship: either syllabic (one-to-one correspondence between pitches and text syllables) or melismatic (more than one pitch corresponding to each text syllable).

Although in Figure 1, MUS-TEXT is encoded categorically for comprehensibility (SYL as syllabic, MEL as melismatic), in the LLDs, it is encoded as: 0 for no text, i. e., rests; 1 for syllabic; -1 for melismatic. As standard procedure in over-time feature extraction [38], for the 5 continuous LLDs (all except for MUS-TEXT), Delta coefficients (Δ), i. e., the differences between two consecutive values, were computed. From now on, we refer to the 6 LLDs and the 5 Δ as Note Level Descriptors (NLDs). ΔPS and INTERVAL are redundant; yet, they were extracted because they belong to different groups and are thus needed in the statistical analysis. Note that $\Delta BEAT$ and $\Delta OFFSET$ are only redundant when the subdivision unit of the time-signature is a crotchet note.

The NLDs were extracted (i) for all the parts together, i. e., 1 NLD matrix per *madrigalism*; (ii) for each part individually, i. e., 5 NLD matrices per *madrigalism*. The former were extracted through the `.flat` property of `music21` (which disregards the vertical alignment across parts) with the only purpose of computing the statistical functionals;⁹ the latter are assembled together into multi-dimensional NLDs, by this preserving the correspondence between parts over time, which is relevant to musical texture, thus, also to *madrigalisms*. Since each part presents a unique note’s configuration, considering the note as frame unit leads to NLD matrices with different lengths across parts. Thus, in order to assemble them, the NLD matrices were re-sampled to the fraction of the shortest note in the corpus, i. e., a semiquaver (cf. Figure 1).

4.2 Statistical Functionals

For the 10 continuous NLDs, 16 functionals were extracted (cf. Table 2). Since for INTERVAL and Δ INTERVAL, the functionals were extracted considering positive and negative values separately, a total of 12 continuous descriptors

Category	Description
Extremes	Maximum, minimum, range
Means	Arithmetic, harmonic, geometric
Moments	Standard deviation, variance, kurtosis, skewness, coefficient of variation
Percentiles	Median, 1 st quartile, 3 rd quartile, interquartile range
Other	Mode

Table 2. Description of the 16 statistical functionals extracted from the continuous NLDs.

were used (i. e., 6 LLDs + 6 Δ). This is necessary to obtain a meaningful result, otherwise the upward and downward intervals are mutually cancelled. For the categorical NLD (MUS-TEXT), 3 functionals were extracted: N(umber) of syllabic notes (N_{syl}); N of melismatic notes (N_{mel}); and the ratio between N_{syl} and N_{mel} ($SYL-MEL_{ratio}$). All in all, 195 functionals were computed: 192 from the continuous NLDs (16 functionals \times 12 continuous descriptors); 3 from the categorical NLD (3 functionals \times 1 NLD).

5. FEATURE EVALUATION

5.1 Feature Groups Comparison

To evaluate whether the chosen feature groups are suitable to differentiate between the *madrigalism* classes (ANT, CON, HOM), Welch-ANOVA was considered, an alternative to the one-way ‘classic’ ANOVA (analysis of variance), suitable in this case: The data were normally distributed but the homogeneity assumption was violated [39]. For the pairwise comparisons across classes, the Games-Howell post-hoc test was employed. Since p -values as evaluation criteria have been repeatedly criticised [40], they will be reported in terms of effect size [41]: epsilon squared (ϵ^2) for the Welch-ANOVA and Hedge’s g for the post-hoc test. To enable the comparison across the three feature groups, Principal Component Analysis (PCA) was applied as a method for dimensionality reduction to the functionals’ vectors of each group. Although PCA implies information loss, this is a plausible method which enabled us to perform an overall assessment. The remaining variances were: 80 % for *Time*; 67 % for *Frequency* and *Time-freq*.

⁹ For the functionals the correspondence between parts is irrelevant.

group	F	df1	df2	p	ϵ^2	ANT-CON				ANT-HOM				CON-HOM			
						lwr	upr	p	g	lwr	upr	p	g	lwr	upr	p	g
<i>Time</i>	6.4	2	410	.002	.02	-1.47	1.67	.988	0.02	-3.51	0.04	.057	0.33	-3.05	-0.61	.001	0.40
<i>Frequency</i>	1.5	2	410	.216	.01	-1.08	1.28	.977	0.03	-0.60	1.98	.413	0.19	-0.24	1.42	.220	0.19
<i>Time-freq.</i>	9.9	2	410	.000	.03	-1.27	1.00	.956	0.04	0.68	3.68	.002	0.42	1.08	3.55	.000	0.52

Table 3. Welch-ANOVA and Games-Howell results for the evaluation of the *madrigalisms*: antiphonal (ANT), contrapuntal (CON), homorhythmic (HOM); for each feature group: *Time*, *Frequency*, *Time-freq.* F statistic, degrees of freedom (df1 and df2), p -values, epsilon-squared (ϵ^2), Hedge’s g , confidence intervals: lower (lwr) and upper (upr), are given.

functional	H	df1	df2	p	η^2	ANT-CON			ANT-HOM			CON-HOM		
						Z	p	d	Z	p	d	Z	p	d
N_{syl} (<i>syllabic</i>)	76.6	2	410	.000	.18	-1.57	.167	0.29	4.76	.000	0.87	8.66	.000	0.88
N_{mel} (<i>melismatic</i>)	26.7	2	410	.000	.06	-0.83	.406	0.10	2.85	.006	0.64	5.04	.000	0.59
SYL-MEL _{ratio} (<i>ratio</i>)	18.67	2	410	.000	.04	-0.98	.326	0.19	2.17	.045	0.26	4.30	.000	0.41

Table 4. Kruskal-Wallis results and Dunn pairwise comparisons for the evaluation of the *madrigalism* classes: ANT, CON, HOM; for the three MUS-TEXT statistical functionals: N(umber) of syllabic and melismatic notes, and the ratio between both. H statistic, degrees of freedom (df1 and df2), p -values, eta-squared (η^2), Z-score, and Cohen’s d , are given.

The statistical analysis shows that the *Time-freq.* features present the most prominent differences across *madrigalism* classes, as indicated by a higher (although small) effect size with respect to the other feature groups ($\epsilon^2 = .03$); cf. ϵ^2 for *Time-freq.* in Table 3. This difference is medium for CON vs HOM ($g = 0.52$), slightly lower for ANT vs HOM ($g = 0.42$), and almost no difference is displayed for ANT vs CON ($g = 0.04$); cf. g for *Time-freq.* in Table 3. The same trend is displayed to a lower extent for *Time*: higher differences are shown for CON vs HOM and ANT vs HOM ($g = 0.40$ and $g = 0.33$, respectively); almost no difference is shown for ANT vs CON ($g = 0.02$); cf. g for *Time* in Table 3. Conversely, all the differences between classes are small for the feature group *Frequency* ($g \leq 0.19$) which indicates that there is no relationship between *madrigalisms*’ texture and specific vocal registers. Nevertheless, the role of frequency-related features should be further investigated by considering the meaning and relevance of the words used in each *madrigalism* class.

Overall, the statistical evaluation indicates that HOM and CON are the *madrigalisms* with the highest dissimilarity, while ANT and CON are the most similar ones. This might seem obvious if we think of the *madrigalisms*’ texture, i. e., by evaluating them from a High Level perspective: Contrapuntal and homorhythmic textures are dissimilar; contrapuntal and antiphonal textures are similar. Yet, our analysis indicates that the functionals related to the *Time* and *Time-freq.* dimensions capture relevant properties in the definition of the evaluated classes; consequently, their NLDs are also representative of *madrigalisms*’ inherent texture. This shows a direct relationship between Low Level and High Level musical descriptors, meaning that measuring the former may enable us to predict the latter.

5.2 Music-Text Relationships

Since the relationships between music and lyrics are crucial in *madrigalisms*, the functionals extracted from the MUS-TEXT NLD (N_{syl} , N_{mel} , and SYL-MEL_{ratio}) are evaluated individually. Note that these are already vectors, thus, PCA was not performed. Since the assumptions for normality and homogeneity were both rejected, the rank-

based non-parametric Kruskal-Wallis test was carried out. For the pairwise comparisons across classes, the Dunn post-hoc test with Benjamini-Hochberg (BH) p -value adjustment was applied [42]. Again, the statistical outcomes will be evaluated in terms of effect size: eta-squared (η^2) for Kruskal-Wallis and Cohen’s d for the Dunn test.

Our analysis shows that the functionals related to the counts of each MUS-TEXT relationship (syllabic and melismatic) are relevant to differentiate between *madrigalism* classes, as shown by the large and moderate effect sizes, respectively: $\eta^2 = .18$ (for N_{syl}); $\eta^2 = .06$ (for N_{mel}); cf. η^2 in Table 4. Differences between classes are less prominent for the ratio, as shown by a lower effect size ($\eta^2 = .04$); cf. η^2 for SYL-MEL_{ratio} in Table 4. Similarly to the outcomes from the overall evaluation (cf. Section 5.1), HOM shows generally noticeable differences with respect to the other two classes. The pairwise comparisons for CON vs HOM and ANT vs HOM indicate big differences for N_{syl} ($d = 0.88$ and $d = 0.87$); moderate for N_{mel} ($d = 0.59$ and $d = 0.64$); smaller (as expected) for SYL-MEL_{ratio} ($d = 0.41$ and $d = 0.26$); cf. Cohen’s d for CON-HOM and ANT-HOM, respectively, in Table 4. Again, ANT vs CON yielded small differences for all the functionals ($0.10 \leq d \leq 0.29$); cf. d for ANT-CON in Table 4.

Since the music-text relationships might particularly vary across the different parts, statistical functionals were also extracted from the MUS-TEXT NLD, considering each part individually;¹⁰ then, the same evaluation was carried out. The results of the statistical analysis for the individual parts, although showing smaller effects, display the same overall trend as described for the parts together. For N_{syl} and N_{mel} (in all the parts), HOM vs the other two classes yielded a moderate effect size ($0.41 \leq d \leq 0.71$), for ANT vs CON a small one ($0.03 \leq d \leq 0.34$). Similarly, for SYL-MEL_{ratio} (in all the parts), all the pairwise comparisons yielded $d \leq 0.40$, except for *Canto*, which showed a slightly higher difference for HOM vs the other two classes ($0.45 \leq d \leq 0.56$). This is due to the *Canto*’s prominent use of melismas in CON and ANT, which – contrasting

¹⁰ The functionals were again extracted before the re-sampling, but in this case, processing the 5 NLD matrices of each *madrigalism* separately. Note that due to space constraints these results are not displayed in a table.

Sp.	Fold A			Fold B			Fold C		
	ANT	CON	HOM	ANT	CON	HOM	ANT	CON	HOM
I	19	58	47	24	79	39	21	64	62
II	22	69	33	26	70	60	16	62	55
III	16	71	42	19	73	58	29	57	48
IV	21	72	51	25	60	49	18	69	48

Table 5. Number of *madrigalisms* per class (ANT, CON, HOM), in the four splittings (Sp.), performed according to the 3-fold composer independent partitioning (A, B, C).

with the syllabism typical of HOM – makes the differences in SYL-MEL_{ratio} across classes much more prominent for this part. This suggests that the role of specific features might be more clearly displayed in some parts, indicating that a particular weight should be attributed to them.

6. EXPERIMENTAL SET-UP

To create the baseline for the automatic recognition of *madrigalisms*' texture, four ML models were considered: a Support Vector Machine (SVM) classifier, a Multi-layer Perceptron (MLP), a Convolutional Neural Network (CNN), and a Bidirectional Long-Short Term Memory Recurrent Neural Network (BLSTM-RNN). The SVM and the MLP were fed with the statistical functionals (cf. Section 4.2), the CNN and the BLSTM-RNN with the multi-dimensional NLDs (cf. Section 4.1). Both feature representations were z-score normalised according to the mean and variance, estimated from the respective training set. In addition, a FUSION approach considering the NLDs and functionals was investigated (cf. Figure 2).

6.1 Partitions, Experiments, and Evaluation Metrics

The experiments were carried out on a 3-fold composer independent partitioning, i. e., the *madrigalisms* were split into 3 disjunct sub-sets (A, B, C), and no *madrigalisms* by the same composer appeared across sub-sets. Note that each madrigal is by a unique composer. The 30 composers were randomly¹¹ assigned to the 3 sub-sets (10 for each), which were considered alternately as training, validation, and test sets. To generalise the outcomes, the 3-fold partitioning was performed 4 times (cf. Table 5), and the experiments were carried out for each of the 4 splits individually. Furthermore, the 6 possible permutations between sub-sets were considered per split; thus, a total of 24 experiments was carried out: 6 permutations \times 4 splits.

As the features from the *Frequency* group proved not to be relevant in the statistical analysis, the following experiments were performed: (i) using the whole feature set (*All*), i. e., 195 functionals (for SVM and MLP) and 11 NLDs (for CNN and BLSTM-RNN); (ii) excluding the features from the *Frequency* group (*Selected*), i. e., 163 functionals and 9 NLDs. Finally, given the high similarity between ANT and CON, two classification problems were addressed: (i) 3C(lass), i. e., considering the three types of *madrigalisms*; (ii) 2C, i. e., excluding ANT (the minority class). Thus, the 24 experiments were performed in four set-ups: 3C with *All* features; 3C with the *Selected* ones;

¹¹ A fixed random seed was chosen to guarantee reproducibility.

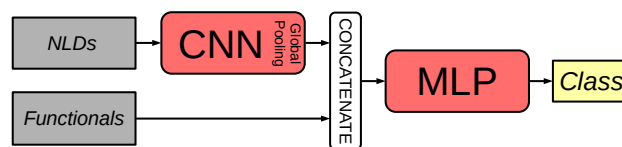


Figure 2. FUSION model: NLDs are fed into a CNN (1 or 2 convolutional blocks and global pooling over time), concatenated with the functionals, and then fed into an MLP.

2C with *All* features; 2C with the *Selected* ones; i. e., 96 experiments (4 splits \times 6 permutations \times 4 set-ups) were carried out per model. To enable a fair comparison, the models were optimised individually for each set-up.

Since the distribution of *madrigalisms* across composers is unbalanced (cf. Table 5), the samples belonging to the minority classes were up-sampled in training by randomly¹¹ duplicating *madrigalisms* until matching the size of the majority class, i. e., CON. Unweighted Average Recall (UAR) was considered as appropriate metric to evaluate the models' performance due to the unbalanced classes in the test set; furthermore, the recall for each class will be discussed. To report the overall results, mean (μ) and standard deviation (σ) across the 24 experiments per set-up and model will be indicated for both UAR and recall.

6.2 Model Optimisation

We employed an SVM classifier with linear kernel built on the python library `scikit-learn` [43]. For the optimisation, five different complexities (C) from 0.00001 to 0.1 (on a logarithmic scale, with a factor of 10 between steps) were considered. The C which yielded the highest UAR on the validation set was chosen to re-train the SVM considering the samples from the training and validation sets together. The MLP, CNN, and BLSTM-RNN were built on `TensorFlow 2.3` [44] through the API `Keras` [45]. For all of them, Adam optimiser, Softmax activation function in the output layer, a maximum of 200 epochs, and early stopping with a patience of 15 were used.

For the MLP, an architecture of two hidden layers with an optimised number of neurons for each given as 25, 75, or 175 and Sigmoid activation, with a dropout of 20% after the first hidden layer, was considered. The batch size was optimised choosing the optimum of 10, 25, and 75; the learning rate was fixed as 0.001. All optimisations and the early stopping were done on the validation set. For the CNN and BLSTM-RNN models, the multi-dimensional NLDs (time \times part \times NLD) were first reshaped fusing the part and NLD dimensions.¹² Both front-ends were followed by a fully-connected network where the same architecture and hyperparameters as for the MLP were used.

The CNN front-end consisted of convolutional blocks with a 1D-convolutional layer of 150 filters followed by batch normalisation, ReLU, and a max-pooling layer. For the convolutional layer, the filter length was 3 and the shift 2; for the max-pooling layer, both the filter length and the shift were 2. The number of convolutional blocks was optimised between one and two. The sequential represen-

¹² Using different *heads* for each part was also tried in initial experiments, but the performance was found to be worse.

3C	<i>All (Time + Freq. + Time-freq.)</i>				<i>Selected (Time + Time-freq.)</i>				
	SVM	MLP	CNN	BLSTM-RNN	SVM	MLP	CNN	BLSTM-RNN	FUSION
ANT	13.0 ± 8.5	14.3 ± 10.4	8.2 ± 7.5	24.0 ± 20.8	13.5 ± 7.0	9.4 ± 9.4	9.6 ± 7.1	19.7 ± 14.8	10.2 ± 8.5
CON	61.5 ± 7.2	57.0 ± 11.8	74.3 ± 11.3	51.1 ± 16.5	62.3 ± 7.5	65.5 ± 11.9	72.4 ± 9.6	55.9 ± 13.3	68.1 ± 8.1
HOM	58.4 ± 5.9	59.4 ± 7.9	70.5 ± 11.8	69.7 ± 16.7	59.7 ± 5.9	59.9 ± 8.8	70.8 ± 12.0	69.8 ± 9.7	67.1 ± 9.0
UAR	44.3 ± 3.0	43.6 ± 4.0	51.0 ± 4.6	48.3 ± 5.6	45.2 ± 3.3	44.9 ± 4.0	50.8 ± 4.4	48.4 ± 5.0	48.5 ± 3.9
2C	<i>All (Time + Freq. + Time-freq.)</i>				<i>Selected (Time + Time-freq.)</i>				
	SVM	MLP	CNN	BLSTM-RNN	SVM	MLP	CNN	BLSTM-RNN	FUSION
CON	74.4 ± 6.0	72.8 ± 6.1	80.0 ± 10.3	74.0 ± 11.6	75.9 ± 6.7	75.0 ± 5.9	77.5 ± 10.0	73.0 ± 11.2	80.3 ± 8.6
HOM	61.4 ± 8.2	62.1 ± 8.2	71.0 ± 9.6	69.3 ± 11.5	62.7 ± 8.4	61.3 ± 8.1	74.2 ± 9.2	70.3 ± 8.0	68.1 ± 10.2
UAR	67.9 ± 3.6	67.4 ± 3.8	75.5 ± 4.5	71.7 ± 6.3	69.3 ± 3.5	68.2 ± 4.6	75.9 ± 4.4	71.7 ± 5.6	74.2 ± 5.0

Table 6. Baseline results for the 3C(lass) and 2C classification of *madrigalisms* (ANT, CON, HOM) for *All* and *Selected* features. Recall per class and Unweighted Average Recall (UAR) are given (highest values marked in bold) for SVM, MLP, CNN, BLSTM-RNN, and FUSION approaches. Mean and standard deviation ($\mu \pm \sigma$) [%] across experiments are indicated.

tations extracted by the CNN front-end are subject to a global max-pooling (over time). The BLSTM-RNN front-end consisted of BLSTM layers of 150 units, a dropout of 20 %, and Tanh activation. The number of layers was again optimised between one and two. When using two layers, the first layer returned a sequential output, followed by self-attention (SeqSelfAttention layer).

7. BASELINE RESULTS

In Table 6, the baseline results are given. Generally, the experiments with *Selected* features present a higher UAR than those with *All* features, which confirms the outcomes of the statistical evaluation (cf. Section 5.1); still, the differences for *Selected* vs *All* for any classifier are small ($\leq 1.4\%$). The model reaching the highest UAR was the CNN (cf. UAR for CNN in Table 6), generally showing a statistically significant difference with respect to the others. Pairwise comparisons with Tukey post-hoc for CNN vs SVM and CNN vs MLP, in all the set-ups, yielded: $p < .0001$, Cohen’s $d > 1.3$; for CNN vs BLSTM-RNN, in 2C: $p < .05$, Cohen’s $d > 0.7$; while in 3C, no significant difference was shown: $p > .05$, Cohen’s $d < 0.5$.

The higher performance of the CNN, and to some extent BLSTM-RNN, might be due to the use of the NLDs, which unlike the functionals contain the correspondences across parts over time, which is relevant in *madrigalisms*. While the CNN generally performed best with one convolutional block (chosen in 84 out of the 96 experiments), the BLSTM-RNN performed best with two layers (65 out of 96). Yet, the BLSTM-RNN generally shows more unstable results across experiments, as displayed by a higher std. dev. (cf. σ in Table 6), which indicates that a simpler architecture can more reliably model the considered data.

The class recognised worst was, as expected, ANT, showing a recall, for all the models, at chance level (cf. μ for ANT in Table 6). This is due to ANT *madrigalisms* being much fewer than the others and very similar to CON, as shown in the features evaluation. The confusion between both classes particularly reduces the recall of CON in the 3C problem, whose improvement is much more prominent than the one shown for HOM when comparing the 3C and the 2C experiments (cf. the upper with respect to lower half of Table 6 for CON and HOM): Across all the models and feature sets, the average recall difference for CON between 2C and 3C is 12.85 %, while for HOM, it is only 1.76 %.

To evaluate whether a FUSION between the multi-dimensional NLDs and the functionals might yield a better performance, the best performing architecture, i. e., the CNN, was concatenated with the functionals, using the same architecture for the MLP as before (cf. Figure 2). The whole network was trained from scratch to enable the model to learn complementary representations. Experiments were carried out considering the same hyperparameter optimisation as previously described and the same four set-ups: 3C and 2C, for *All* and *Selected* features. The same pairwise combinations between functionals and NLDs were used: *Selected* functionals were concatenated with the output of the CNN trained with *Selected* multi-dimensional NLDs, and correspondingly for *All*. In Table 6, the best results for 3C and 2C problems with the FUSION model, i. e., considering the *Selected* features, are given. While FUSION’s recall on CON (2C) increased over the one from the CNN, no consistent improvement is shown, which indicates that multi-dimensional NLDs are a good representation of *madrigalisms’* texture on their own.

8. LIMITATIONS & CONCLUDING REMARKS

Our research outcomes indicate that symbolic features and ML methods are both appropriate to further investigate word-painting strategies in madrigals. They also highlight the potential of applying AI in the study of Renaissance music. Yet, since this study was the first of its kind, at this stage we evaluated the lyrics only in terms of syllabic and melismatic relationship, while the importance of specific words, which might be given by their meaning (both linguistic/metaphorical) within and across madrigals, typically highlighted through specific word-painting strategies, was not yet considered. A deeper evaluation of the lyrics is indeed one of the next priorities in our future work, by this systematically identifying the connections between music and poetry in the Italian madrigal. Furthermore, we will also compare the ML outcomes from the feature-based methods with those achieved through humdrum-based end-to-end approaches already presented in the literature [46].

Our work shows that symbolic Low Level Descriptors are suitable to automatically identify different textures in Italian madrigals. In addition, the presented baseline will hopefully stimulate further research advances in the application of ML to early music, by this promoting a deeper understanding of the Renaissance musical heritage.

9. REFERENCES

- [1] W. Apel, *The Harvard dictionary of music*. Cambridge, MA, USA: Harvard University Press, 2003.
- [2] E. Ricciardi and C. Sapp, “Editing Italian madrigals in the digital world: The Tasso in Music Project,” in *Proc. of Music Encoding Conference*, Virtual event, 2020.
- [3] L. Pugin, “Editing Renaissance music: The Aruspix Project,” in *Beihefte zur Editio: Internationales Jahrbuch für Editionswissenschaften*. Tübingen: Max Niemeyer, 2009, pp. 94–103.
- [4] F. J. Castellanos, J. Calvo-Zaragoza, and J. M. Inesta, “A neural approach for full-page optical music recognition of mensural documents,” in *Proc. of the International Society for Music Information Retrieval Conference*. Virtual event: ISMIR, 2020, pp. 23–27.
- [5] D. Rizo, J. Calvo-Zaragoza, and J. M. Inesta, “Muret: A music recognition, encoding, and transcription tool,” in *Proc. of the International Conference on Digital Libraries for Musicology*, Paris, France, 2018, pp. 52–56.
- [6] G. Vigliensoni, A. Daigle, E. Liu, J. Calvo-Zaragoza, J. Regimbal, M. A. Nguyen, N. Baxter, Z. McLennam, and I. Fujinaga, “From image to encoding: Full optical music recognition of Medieval and Renaissance music,” in *Proc. of Music Encoding Conference*, Vienna, Austria, 2019.
- [7] Y. Ju, S. Howes, C. McKay, N. Condit-Schultz, J. Calvo-Zaragoza, and I. Fujinaga, “An interactive workflow for generating chord labels for homorhythmic music in symbolic formats,” in *Proc. of the International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, 2019, pp. 862–869.
- [8] A. Leemhuis, S. Waloschek, and A. Hadjakos, “Bacher than Bach? On musicologically informed AI-based Bach chorale harmonization,” in *Proc. of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Würzburg, Germany: Springer, 2019, pp. 462–469.
- [9] H. Hild, J. Feulner, and W. Menzel, “Harmonet: A neural net for harmonizing chorales in the style of J. S. Bach,” in *Advances in Neural Information Processing Systems*, 1992, pp. 267–274.
- [10] M. Krause, F. Zalkow, J. Zalkow, C. Weiß, and M. Müller, “Classifying leitmotifs in recordings of Operas by Richard Wagner,” in *Proc. of the International Society for Music Information Retrieval Conference*. Virtual event: ISMIR, 2020, pp. 473–480.
- [11] E. Parada-Cabaleiro, M. Schmitt, A. Batliner, S. Hantke, G. Costantini, K. Scherer, and B. Schuller, “Identifying emotions in Opera singing: Implications of adverse acoustic conditions,” in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 276–382.
- [12] C. Brazier and G. Widmer, “Addressing the recitative problem in real-time Opera tracking,” *arXiv preprint arXiv:2010.11013*, 2020.
- [13] C. Antila and J. Cumming, “The VIS framework: Analyzing counterpoint in large datasets,” in *Proc. of the International Society for Music Information Retrieval Conference*. Taipei, Taiwan: ISMIR, 2014, pp. 71–76.
- [14] A. Brinkman, D. Shanahan, and C. Sapp, “Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of Josquin,” in *Proc. of the Biennial International Conference on Music Perception and Cognition*, San Francisco, CA, USA, 2016, pp. 91–97.
- [15] C. Sapp, “Suggestions for future corpus-based text painting analyses: A response to Strykowski,” *Empirical Musicology Review*, vol. 11, no. 2, pp. 120–123, 2017.
- [16] E. Parada-Cabaleiro, A. Batliner, A. E. Baird, and B. Schuller, “The SEILS dataset: Symbolically encoded scores in modern-early notation for computational musicology,” in *Proc. of the International Society for Music Information Retrieval Conference*. Suzhou, China: ISMIR, 2017, pp. 575–581.
- [17] M. S. Cuthbert and C. Ariza, “Music21: A toolkit for computer-aided musicology and symbolic music data,” in *Proc. of the International Society for Music Information Retrieval Conference*. Utrecht, Netherlands: ISMIR, 2010, pp. 637–642.
- [18] C. Sapp, “Online database of scores in the Humdrum file format,” in *Proc. of the International Society for Music Information Retrieval Conference*. London, UK: ISMIR, 2005, pp. 664–665.
- [19] I. Cividini, “Zwischen klassischer Musikphilologie und angewandter Informatik: Die Digitale Mozart-Edition (DME) der Stiftung Mozarteum Salzburg,” in *Jahrestagung der Gesellschaft für Musikforschung*, Paderborn / Detmold, Germany, 2019.
- [20] M. Gotham, P. Jonas, B. Bower, W. Bosworth, D. Rootham, and L. VanHandel, “Scores of scores: An openscore project to encode and share sheet music,” in *Proc. of the International Conference on Digital Libraries for Musicology*, Paris, France, 2018, pp. 87–95.
- [21] N. Nápoles, G. Vigliensoni, and I. Fujinaga, “Encoding matters,” in *Proc. of the International Conference on Digital Libraries for Musicology*, Paris, France, 2018, pp. 69–73.
- [22] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, “A survey on bias and fairness in machine learning,” *arXiv preprint arXiv:1908.09635*, 2019.
- [23] J. Cumming, C. McKay, J. Stuchbery, and I. Fujinaga, “Methodologies for creating symbolic corpora of

- Western music before 1600,” in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 491–498.
- [24] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, “Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis,” in *Proc. of the International Society for Music Information Retrieval Conference*. Málaga, Spain: ISMIR, 2015, pp. 721–734.
- [25] M. Giraud, R. Groult, and E. Leguy, “Dezrann, a web framework to share music analysis,” in *Proc. of the International Conference on Technologies for Music Notation and Representation*. Montreal, QC, Canada: TENOR, 2018, pp. 104–110.
- [26] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, “The annotated Beethoven corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets,” *Frontiers in Digital Humanities*, vol. 5, 2018.
- [27] P. Allegraud, L. Bigo, L. Feisthauer, M. Giraud, R. Groult, E. Leguy, and F. Levé, “Learning sonata form structure on Mozart’s string quartets,” *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 82–96, 2019.
- [28] M. Giraud, R. Groult, and F. Levé, “Computational analysis of musical form,” in *Computational Music Analysis*. Springer, 2016, pp. 113–136.
- [29] E. Parada-Cabaleiro, M. Schmitt, A. Batliner, and B. W. Schuller, “Musical-linguistic annotations of Il Lauro Secco,” in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 461–467.
- [30] R. de Valk, R. Ahmed, and T. Crawford, “JosquIntab: A dataset for content-based computational analysis of music in lute tablature,” in *Proc. of the International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, 2019, pp. 431–438.
- [31] E. Parada-Cabaleiro, A. Batliner, and B. Schuller, “A diplomatic edition of il Lauro Secco: Ground truth for OMR of white mensural notation,” in *Proc. of the International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, 2019, pp. 557–564.
- [32] A. Kirkman, “Review: The Josquin research project by Jesse Rodin and Craig Sapp,” *Journal of the American Musicological Society*, vol. 68, no. 2, pp. 455–465, 2015.
- [33] P. van Kranenburg and G. Maessen, “Comparing offertory melodies of five Medieval Christian traditions,” in *Proc. of the International Society for Music Information Retrieval Conference*. Suzhou, China: ISMIR, 2017, pp. 204–210.
- [34] R. de Valk and T. Weyde, “Bringing ‘musicque into the tableture’: Machine-learning models for polyphonic transcription of 16th-century lute tablature,” *Early Music*, vol. 43, no. 4, pp. 563–576, 2015.
- [35] M. Müller, *Fundamentals of music processing*. Cham, Switzerland: Springer Verlag, 2015.
- [36] A. Aljanaki and M. Soleymani, “A data-driven approach to mid-level perceptual musical feature modeling,” in *Proc. of the International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, 2018, pp. 615–621.
- [37] D. C. Corrêa and F. A. Rodrigues, “A survey on symbolic data-based music genre classification,” *Expert Systems with Applications*, vol. 60, pp. 190–210, 2016.
- [38] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile: The Munich versatile and fast open-source audio feature extractor,” in *Proc. of ACM Multimedia*. ACM, 2010, pp. 1459–1462.
- [39] M. B. Brown and A. B. Forsythe, “372: The ANOVA and multiple comparisons for data with heterogeneous variances,” *Biometrics*, pp. 719–724, 1974.
- [40] R. L. Wasserstein and N. A. Lazar, “The ASA’s statement on p-values: Context, process, and purpose,” *The American Statistician*, vol. 70, pp. 129–133, 2016.
- [41] D. Lakens, “Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and ANOVAs,” *Frontiers in Psychology*, vol. 4, 2013.
- [42] Y. Benjamini and Y. Hochberg, “Controlling the false discovery rate: A practical and powerful approach to multiple testing,” *Journal of the Royal Statistical Society*, vol. 57, pp. 289–300, 1995.
- [43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *Proc. of the {USENIX} Symposium on Operating Systems Design and Implementation*, 2016, pp. 265–283.
- [45] F. Chollet *et al.*, “Keras,” <https://github.com/fchollet/keras>, 2015.
- [46] H. Verma and J. Thickstun, “Convolutional composer classification,” in *Proc. of the International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, 2019, pp. 549–556.

IS DISENTANGLEMENT ENOUGH? ON LATENT REPRESENTATIONS FOR CONTROLLABLE MUSIC GENERATION

Ashis Pati

Center for Music Technology
Georgia Institute of Technology, USA
ashis.pati@gatech.edu

Alexander Lerch

Center for Music Technology
Georgia Institute of Technology, USA
alexander.lerch@gatech.edu

ABSTRACT

Improving *controllability* or the ability to manipulate one or more attributes of the generated data has become a topic of interest in the context of deep generative models of music. Recent attempts in this direction have relied on learning disentangled representations from data such that the underlying factors of variation are well separated. In this paper, we focus on the relationship between disentanglement and controllability by conducting a systematic study using different supervised disentanglement learning algorithms based on the Variational Auto-Encoder (VAE) architecture. Our experiments show that a high degree of disentanglement can be achieved by using different forms of supervision to train a strong discriminative encoder. However, in the absence of a strong generative decoder, disentanglement does not necessarily imply controllability. The structure of the latent space with respect to the VAE-decoder plays an important role in boosting the ability of a generative model to manipulate different attributes. To this end, we also propose methods and metrics to help evaluate the quality of a latent space with respect to the afforded degree of controllability.

1. INTRODUCTION

Automatic music generation using machine learning has seen significant improvements over the last decade. Deep generative models relying on neural networks have been successfully applied to several different music generation tasks, e.g., monophonic music generation consisting of a single melodic line [1–3], polyphonic music generation involving several different parts or instruments [4, 5], and creating musical renditions with expressive timing and dynamics [6, 7]. However, such models are usually found lacking in two critical aspects: *controllability* and *interactivity* [8]. Most of the models typically work as black-boxes, i.e., the intended end-user has little to no control over the generation process. Additionally, they do not allow any modes for interaction, i.e., the user cannot selectively modify the generated music or some of its parts based on desired musical

characteristics. Consequently, there have been considerable efforts focusing on controllable music generation [9–11] in interactive settings [12–14]. One promising avenue for enabling controllable music generation stems from the field of representation learning.

Representation learning involves automatic extraction of the underlying factors of variation in given data [15]. The majority of the current state-of-the-art machine learning-based methods aim at learning compact and useful representations [16, 17]. These have been used for solving different types of discriminative or generative tasks spanning several domains such as images, text, speech, audio, and music. A special case of representation learning deals with *disentangled* representations, where individual factors of variation are clearly separated such that changes to a single underlying factor in the data lead to changes in a single factor of the learned disentangled representation [18]. Specifically, in the context of music, disentangled representations have been used for a wide variety of music generation tasks such as rhythm transfer [10, 19], genre transfer [20], instrument rearrangement [21], timbre synthesis [22], and manipulating low-level musical attributes [23–25].

Disentangled representation learning has been an active area of research in the context of deep generative models for music. Previous methods have focused on different types of musical attributes (e.g., note density [23], rhythm [10], timbre [22], genre [20], and arousal [25]) and have achieved promising results. However, contrary to other fields such as computer vision [18, 26], research on disentanglement learning in the context of music has been task-specific and ad-hoc. Consequently, the degree to which disentangled representations can aid controllable music generation remains largely unexplored. While we have shown that unsupervised disentanglement learning methods are not suitable for music-based tasks [27], the use of supervised learning methods has not been systematically evaluated.

In this paper, we conduct a systematic study on controllable generation by using supervised methods to learn disentangled representations. We compare the performance of several supervised methods and conduct a series of experiments to objectively evaluate their performance in terms of disentanglement and controllability for music generation. In the context of this paper, *controllability* is defined as the ability of a generative model to selectively, independently, and predictably manipulate one or more attributes (for instance, rhythm, scale) of the generated data. We show



that while supervised learning methods can achieve a high degree of disentanglement in the learned representation, not all methods are equally useful from the perspective of controllable generation. The degree of controllability depends not only on the learning methods but also on the musical attribute to be controlled. In order to foster reproducibility, the code for the conducted experiments is available online.¹

2. METHOD & EXPERIMENTAL SETUP

The primary goal of this paper is to investigate the degree to which learning disentangled representations can provide control over manipulating different attributes of the generated music. To this end, we train generative models based on Variational Auto-Encoders (VAEs) [28] to map high-dimensional data in \mathcal{X} to a low-dimensional latent space \mathcal{Z} by approximating the posterior distribution $q(\mathbf{z}|\mathbf{x})$ (encoder). The latent vectors $\mathbf{z} \in \mathcal{Z}$ can then be sampled to generate new data in \mathcal{X} using the learned likelihood $p(\mathbf{x}|\mathbf{z})$ (decoder). We use different supervised learning methods to enforce disentanglement in the latent space by regularizing specific attributes of interest along certain dimensions of the latent space. These attributes can then be manipulated by using simple traversals across the regularized dimensions. Once the models are trained, different experiments are conducted to evaluate disentanglement and controllability.

2.1 Learning Methods

Three different disentanglement learning methods are considered. Each method adds a supervised regularization loss to the VAE-training objective

$$L = L_{\text{VAE}} + \gamma L_{\text{reg}}, \quad (1)$$

where L , L_{VAE} , L_{reg} correspond to the overall loss, the VAE-loss [28], and the regularization loss respectively. The hyperparameter γ is called the regularization strength.

The first method, referred to as I-VAE, is based on the regularization proposed by Adel et al. [29]. It uses a separate linear classifier attached to each regularized dimension to predict the attribute classes. Note that while Adel et al. use this regularization while learning a non-linear transformation of a latent space, we apply it during training of the latent space itself. This is a suitable choice for categorical attributes and is similar to the regularizer used in MIDI-VAE [20]. The second method is the S2-VAE [26]. This regularization, designed for continuous attributes, uses a binary cross-entropy loss to match attribute values to the regularized dimension. The third method is the AR-VAE [30], which uses a batch-dependent regularization loss to encode continuous-valued attributes along specific dimensions of the latent space. This method is effective at regularizing note density and rhythm-based musical attributes [25]. For comparison, baseline results obtained using the unsupervised β -VAE method [31] are also provided.

¹ https://github.com/ashishpati/dmelodies_controllability
last accessed: 1st Aug 2021

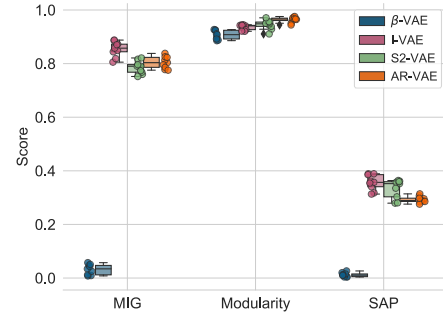


Figure 1: Overall disentanglement performance (higher is better) of different supervised methods on dMelodies. Individual points denote results for different hyperparameter and random seed combinations.

2.2 Dataset & Data Representation

To conduct a systematic study and objectively evaluate the different methods, not only do we need to be able to measure the degree of disentanglement in the learned representations, but we should also be able to measure the attribute values in the generated data. Considering this, we use the dMelodies dataset [27] which is an algorithmically constructed dataset with well-defined factors of variation specifically designed to enable objective evaluation of disentanglement learning methods for musical data. This dataset consists of simple 2-bar monophonic melodies which are based on arpeggiations over the standard I-IV-V-I cadence chord pattern. The dataset has the following factors of variation: *Tonic*, *Octave*, *Scale*, *Rhythm* for bars 1 and 2, and the *Arpeggiation* directions for each of four chords. We use the tokenized data representation used by dMelodies [27].

2.3 Model Architectures & Training Specifications

The VAE architecture is based on a hierarchical RNN model [27], which is inspired by the MusicVAE model [1]. Additional experiments using a CNN-based architecture are omitted here for brevity but provided in the supplementary material.¹ Since both S2-VAE and AR-VAE are designed for continuous attributes, the factors of variation are treated as continuous values by considering the index of the category as the attribute value and then normalizing them to $[0, 1]$. For instance, the *Scale* attribute has 3 distinct options and hence, the normalized continuous values are $[0, \frac{1}{2}, 1]$ corresponding to the major, harmonic minor, and blues scales, respectively. Three different values of regularization strength $\gamma \in \{0.1, 1.0, 10.0\}$ are used.

For each of the above methods and hyperparameter combinations, three models with different random seeds are trained. The dataset is divided into training, validation, and test set using a 70%-20%-10% split. To ensure consistency across training, all models are trained with a batch size of 512 for 100 epochs. The ADAM optimizer [32] is used with a fixed learning rate of $1e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-8$.

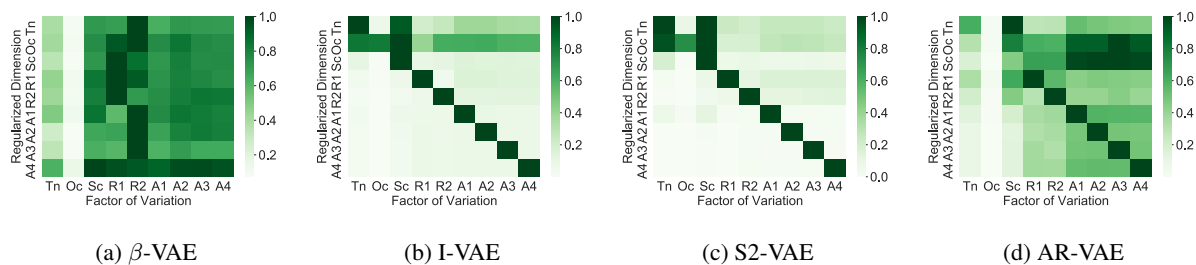


Figure 2: Attribute-change matrices for different methods. Tn: Tonic, Oc: Octave, Sc: Scale, R1 and R2: rhythm for bars 1 and 2 respectively, A1-A4: arpeggiation direction for the four chords.

3. RESULTS AND DISCUSSION

We now present and discuss the results of the different experiments conducted. The first experiment objectively measures the degree of disentanglement in the representations learned using the different methods. The second experiment evaluates the degree to which each method allows independent control over the different attributes. The third experiment throws additional light into the behavior by visualizing the latent spaces with respect to the different attributes. Then, we introduce a new metric to evaluate the quality of latent spaces with respect to the decoder. Finally, we present a qualitative inspection of the data generated by traversals along different regularized dimensions to further illustrate the key findings.

3.1 Attribute Disentanglement

In order to objectively measure disentanglement, we rely on commonly used metrics: (a) Mutual Information Gap (MIG) [33], which measures the difference of mutual information between a given attribute and the top two dimensions of the latent space that share maximum mutual information with the attribute, (b) Modularity [34], which measures if each dimension of the latent space depends on only one attribute, and (c) Separated Attribute Predictability (SAP) [35], which measures the difference in the prediction error of the two most predictive dimensions of the latent space for a given attribute. For each metric, the mean across all attributes is used for aggregation. For consistency, standard implementations are used [18].

The disentanglement performance of the three supervised methods on the held-out test set is compared against the β -VAE model in Figure 1. Unsurprisingly, all three supervised methods outperform the β -VAE across the three disentanglement metrics. The improvement is much higher for the *MIG* and *SAP* score which both measure the degree to which each attribute is encoded only along a single dimension of the latent space.

Using supervision, therefore, leads to better overall disentanglement. Note that this superior performance is achieved without sacrificing the reconstruction quality. All three supervised methods achieve a reconstruction accuracy $> 90\%$.¹ This is a considerable improvement over the unsupervised learning methods seen in the dMelodies benchmarking experiments (average accuracy of $\approx 50\%$ [30]).

3.2 Independent Control during Generation

Considering that supervised methods can obtain better disentanglement along with good reconstruction accuracy, we now look at how effective these methods are for independently controlling different attributes. To measure this quantitatively, we propose the following protocol. Given a data-point with latent vector \mathbf{z} , 6 different variations are generated by uniformly interpolating along the dimension r_l , where r_l is the regularized dimension for attribute a_l . The limits of interpolation are chosen based on the maximum and minimum latent code values obtained during encoding the validation data. For the β -VAE model, the dimension with the highest mutual information with the attribute is considered as the regularized dimension. An attribute change matrix $A \in \mathbb{R}^{L \times L}$, where L is the number of attributes, is computed using the following formulation:

$$A(m, n) = \sum_{i=1}^6 [0 \neq |a_n(\mathbf{z}_i^m) - a_n(\mathbf{z})|], \quad (2)$$

where $A(m, n)$ computes the net change in the n^{th} attribute as one traverses the dimension r_m (which regularizes the m^{th} attribute), $[\cdot]$ represents the inverse Kronecker delta function, $a_n(\cdot)$ is the value of the n^{th} attribute, and \mathbf{z}_i^m is the i^{th} interpolation of \mathbf{z} obtained by traversing along the r_m dimension. This attribute change matrix is computed for each model type by averaging over a total of 1024 data-points in the test-set and across all 3 random seeds (regularization hyperparameters are fixed at $\beta = 0.2$, $\gamma = 1.0$). The matrix is also normalized so that the maximum value across each row corresponds to one. Independent control over attributes should result in the matrix A having high values along the diagonal and low values on the off-diagonal entries which would denote that traversing a regularized dimension only affects the regularized attribute.

The following observations can be made from the matrices visualized in Figure 2. First, β -VAE performs the worst as traversals along different dimensions change multiple attributes simultaneously. Second, among the supervised methods, I-VAE and S2-VAE seem to perform better than AR-VAE. This can be seen from the lighter shades of the off-diagonal elements in the plots for I-VAE and S2-VAE. While the better performance of I-VAE is expected since it is designed for categorical attributes, the poorer performance of AR-VAE in comparison to S2-VAE needs further investigation. Finally, the *scale* attribute (3rd column) changes

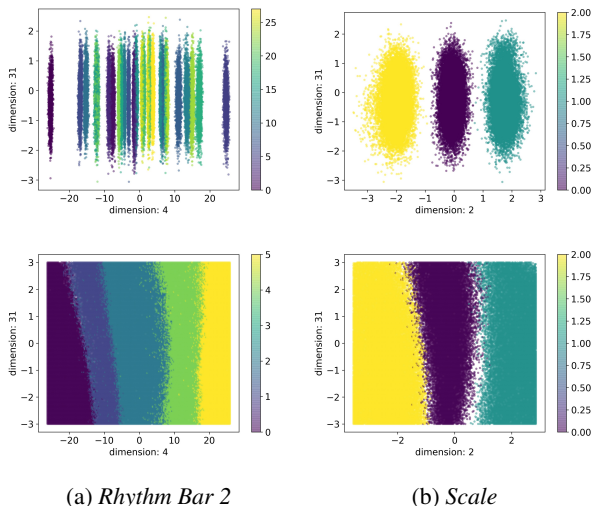


Figure 3: Data distribution (top row) and surface plots (bottom row) for I-VAE.

the most while traversing the regularized dimensions for the supervised methods. This indicates that all supervised methods struggle in generating notes conforming to particular scales. One explanation for this could be that the *scale* is the most complex among all the attributes. Note that while there is no considerable difference between the disentanglement performance of the three methods (compare Figure 1), I-VAE and S2-VAE show much better performance compared to AR-VAE in this experiment which shows that disentanglement does not ensure better controllability.

3.3 Latent Space Visualization

To better understand the difference between disentanglement and controllability of attributes, we try to visualize the structure of the latent space with respect to the different attributes. This is done using 2-dimensional *data distribution* and *latent surface* plots. Both plots show the variance of a given attribute (using different colors for different attribute values) with respect to the regularized dimension (shown on the *x*-axis) and a randomly chosen non-regularized dimension (shown on the *y*-axis).

For the data distribution plots, first, latent representations are obtained for data in the held-out test set using the VAE-encoder. Then, for each attribute, these representations are projected onto a 2-dimensional plane where the *x*-axis corresponds to the regularized dimension and the *y*-axis corresponds to a non-regularized dimension. To generate the surface plots, for a given attribute, a 2-dimensional plane on the latent space is considered which comprises the regularized dimension for the attribute and a non-regularized dimension. The latent code for the other dimensions is drawn from a normal distribution and kept fixed. The latent vectors thus obtained are passed through the VAE decoder and the attributes of the generated data are plotted.

Figures 3, 4, and 5 show the results for I-VAE, S2-VAE, and AR-VAE respectively. In each figure, the top row corresponds to the data distribution plots, and the bottom row shows the latent surface plots. For the surface plots, the

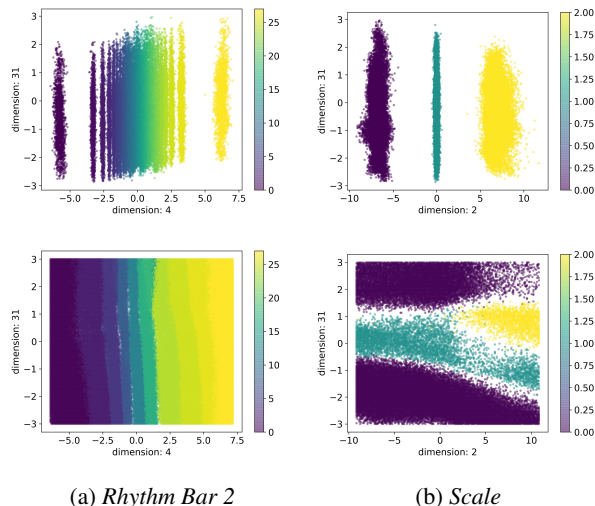


Figure 4: Data distribution (top row) and surface plots (bottom row) for S2-VAE.

generated data-points sometimes have attribute values that are either not present in the training set or cannot be determined (e.g., the generated melody might not conform to any of the 3 possible scales in the dataset, or the arpeggiation direction might be neither up nor down). These *undefined* or out-of-distribution attribute values are shown as empty spaces in the latent surface plots.

For all three methods, the data distribution plots (top rows) show a clear separation of attribute values along the regularized dimension which explains the high disentanglement performance seen in Section 3.1. However, the methods differ considerably when the latent surface plots (bottom rows) are compared. I-VAE (see Figure 3) shows good performance where moving along the regularized dimension (*x*-axis) changes the corresponding attribute, while traversals along the non-regularized dimension (*y*-axis) have little effect. However, the manner of change is unpredictable. For instance, in Figure 3(a)(bottom), only 5 out of the 28 possible rhythms are generated. In addition, the order of the generated rhythms is different from the encoder distribution in Figure 3(a)(top). In contrast, for S2-VAE, the gradual change of color in Figure 4(a)(bottom) shows a high degree of controllability for the rhythm attribute. However, it struggles to control the *scale* attribute. Traversing along the non-regularized dimension in Figure 4(b)(bottom) results in an undesirable change in the *scale* of the generated melody. The latent space of AR-VAE (see Figure 5) has the most discrepancies. Not only is the latent space not centered around the origin (see the top row of Figure 5(b)) for the *scale* attribute, but the degree of controllability is also poor. For instance, the *scale* attribute does not change at all along the regularized dimension (see Figure 5(b)(bottom)). In addition, the empty spaces in the surface plots show that many of the generated data-points have an out-of-distribution attribute value. Results for all other attributes are provided in the supplementary material.¹

The empty regions in the latent spaces show that while these methods can train strong discriminative encoders

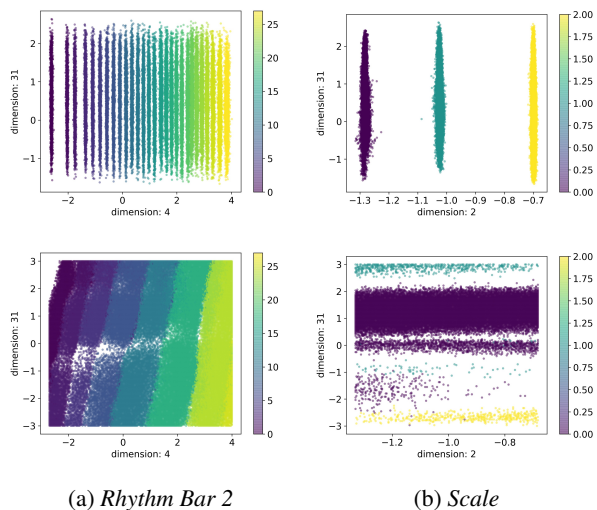


Figure 5: Data distribution (top row) and surface plots (bottom row) for AR-VAE.

which are good for disentanglement, they tend to have weak generative decoders which are incapable of utilizing the learned disentangled representations thereby resulting in *holes* or vacant regions in the latent space where the behavior of the decoder is unpredictable.

3.4 Latent Density Ratio

From the perspective of the VAE-decoder, holes in the latent space can have a significant impact on controllability. Yet, established metrics do not capture this phenomenon properly. To help quantify this, we propose the Latent Density Ratio (LDR) metric. We first sample a set of N ($=10k$) points in the latent space, pass them through the VAE decoder, and compute the percentage of data-points with valid attribute values out of the total number N . The overall LDR is obtained by averaging this metric across all attributes. The results in Table 1 show that both S2-VAE and I-VAE have a lower degree of holes (higher LDR value) in comparison to AR-VAE which is in line with observations in the previous experiments.

3.5 Qualitative Inspection of Latent Interpolations

Finally, we take a qualitative look at the data generated by the different methods while traversing the latent space along the regularized dimensions. Ideally, traversals along a regularized dimension should only cause changes in the corresponding attribute while leaving the other attributes unchanged. In addition, the regularized attribute should also change in a predictable manner. Figure 6 shows the results for the I-VAE method. For each sub-figure, different rows correspond to melodies generated by traversing along the regularized dimension for the attribute in the sub-figure caption. Results for S2-VAE and AR-VAE are shown in Figures 8 and 7, respectively.

Across methods, most of the time, the melodies generated by traversing along regularized dimensions show changes in the corresponding attribute only. For instance,

Learning Method	LDR
I-VAE	0.448
S2-VAE	0.544
AR-VAE	0.244

Table 1: LDR metric (higher is better) for different methods

in Figures 6(a) and 7(a), only the rhythm of the second bar changes while the rest of the melody stays intact. In Figure 6(c,d), the arpeggiation directions of the third and fourth chords are flipped, respectively. Also, in Figure 6(b), all the other attributes remain constant (rhythm, arpeggiation directions) while the pitches of the generated notes change to reflect different scales. While this is desirable, there are a few important things to note.

First, the *scale* attribute seems hard to control. For instance, in Figure 6(b), for I-VAE, some of the generated melodies (the first two rows) do not conform to any of the scales present in the dataset. In Figure 7(b), for AR-VAE, the *scale* does not change at all. This difficulty in controlling the *scale* attribute was also observed in Section 3.2. Second, depending on the holes in the latent space, traversals along regularized dimensions sometimes create melodies with attributes that are unseen in the training data. This happens also for attributes other than *scale*. For instance, in Figure 7(c), row 2, the third chord has an unseen arpeggiation direction. Finally, for I-VAE, the direction of change for arpeggiation factors (see Figure 6(c,d)) is unpredictable. While the arpeggiation direction (of the third chord) goes from up to down in Figure 6(c), the direction (for the fourth chord) is flipped from down to up in Figure 6(d). This is due to the I-VAE regularization formulation which is agnostic to the order of the categorical attributes. Contrast this to AR-VAE and S2-VAE, where the nature of the change in the attribute values is predictable. The direction of arpeggiation will always go from up to down for these methods (see Figures 8(a,b) and 7(c,d)).

3.6 Discussion

The results of the experiments in this section show that supervised methods for disentanglement perform significantly better than unsupervised methods. This is expected since the former use attribute-specific information during training to guide the model towards learning better representations. Among the supervised methods, there are no major differences in terms of the disentanglement metrics in Section 3.1. However, controllability during data generation (discussed in Sections 3.2, and 3.5) differs considerably between the methods. These differences suggest that while disentanglement is closely related to a strong encoder (learning the posterior $q(\mathbf{z}|\mathbf{x})$), improving controllability requires a strong decoder (learning the likelihood $p(\mathbf{x}|\mathbf{z})$). This explains the often better performance of conditioning-based methods relying on adversarial training of decoders [36,37].

Visualizing the latent spaces (in Section 3.3) with respect to the attribute values highlights that the presence or

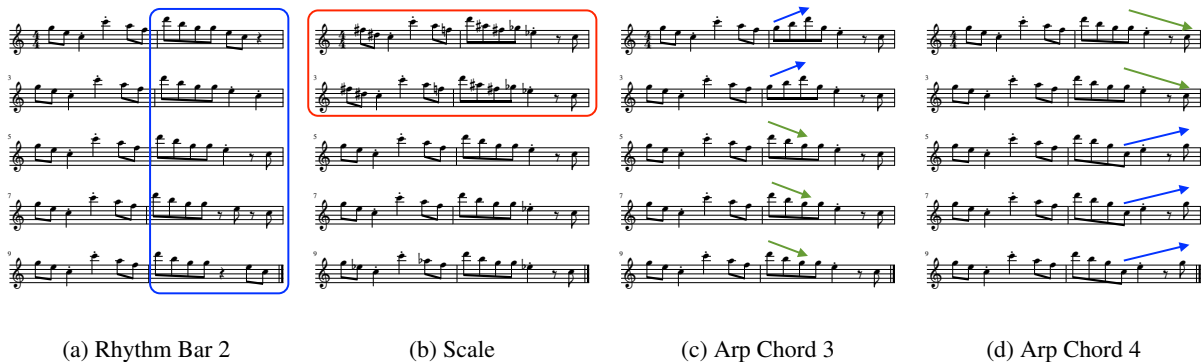


Figure 6: Generated data by traversing along regularized dimensions for I-VAE.

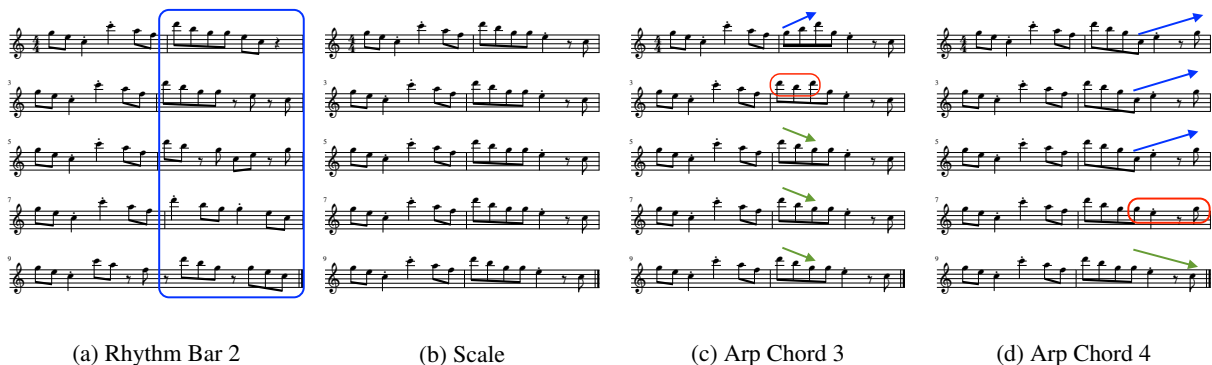


Figure 7: Generated data by traversing along regularized dimensions for AR-VAE.

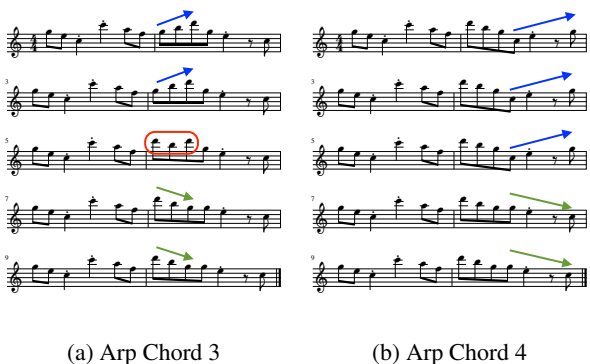


Figure 8: Generated data by traversing along regularized dimensions for S2-VAE.

absence of holes in the learned latent space plays a crucial role in the degree of controllability afforded by a model. The LDR metric proposed in Section 3.4 is an attempt to quantify this behavior. Note that other factors can be considered while evaluating controllability that have been left out of this study. For instance, for continuous-valued attributes, one would prefer the regularized dimension having a positive correlation with the attribute value [30].

4. CONCLUSION

In this paper, we present a systematic investigation of the relationship between attribute disentanglement and controllability in the context of symbolic music. Through a diverse

set of experiments using different methods, we show that even though different supervised learning techniques can force effective disentanglement in the learned representations to a comparable extent, not all methods are equally effective at allowing control over the attributes during the data generation process. This distinction is important because controllability is paramount for generative models [8] and is often not taken into account while evaluating disentanglement learning methods.

An important observation is the issue of holes in latent spaces. It should be noted this has also been seen in other data domains relying on discrete data such as text [38]. There are a few promising directions to address this problem. One option is to constrain the latent space to conform to a specific manifold and perform manipulations within this manifold [38, 39]. An alternative direction could be to learn specific transformation paths within the existing latent manifold to avoid these holes [40].

The experiments in this paper have used labels from the entire training set. Another interesting direction for future studies could be to extend these experiments to a semi-supervised paradigm by using a limited number of labels obtained from only a fraction of the training set [26]. This would increase the confidence in applying these methods to real-world data where obtaining label information for the entire dataset might be either too costly or simply impossible.

5. ACKNOWLEDGMENTS

The authors would like to thank NVIDIA Corporation (Santa Clara, CA, United States) for supporting this research via the NVIDIA GPU Grant program.

6. REFERENCES

- [1] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," in *Proc. of 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018.
- [2] F. Colombo, S. Muscinelli, A. Seeholzer, J. Brea, and W. Gerstner, "Algorithmic composition of melodies with deep recurrent neural networks," in *Proc. of 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, Huddersfield, UK, 2016.
- [3] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proc. of 1st Conference on Computer Simulation of Musical Creativity (CSMC)*, Huddersfield, UK, 2016.
- [4] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. of 18th International Society of Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 324–331.
- [5] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. of 29th International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.
- [6] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *Proc. of International Conference of Learning Representations (ICLR)*, New Orleans, USA, 2019.
- [7] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, pp. 1–13, 2018.
- [8] J.-P. Briot and F. Pachet, "Deep learning for music generation: Challenges and directions," *Neural Computing and Applications*, 2018.
- [9] A. Pati, A. Lerch, and G. Hadjeres, "Learning to Traverse Latent Spaces for Musical Score Inpainting," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.
- [10] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.
- [11] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proc. of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [12] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: A steerable model for Bach chorales generation," in *Proc. of 34th International Conference on Machine Learning (ICML)*, Sydney, Australia, 2017, pp. 1362–1371.
- [13] C. Donahue, I. Simon, and S. Dieleman, "Piano genie," in *Proc. of 24th International Conference on Intelligent User Interfaces (IUI)*, Los Angeles, USA, 2019, pp. 160–164.
- [14] T. Bazin and G. Hadjeres, "Nonoto: A model-agnostic web interface for interactive music composition by inpainting," in *Proc. of 10th International Conference on Computational Creativity (ICCC)*, UNC Charlotte, NC, USA, 2019.
- [15] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013.
- [16] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. of 37th International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 1597–1607.
- [17] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," in *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020.
- [18] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations," in *Proc. of 36th International Conference on Machine Learning (ICML)*, Long Beach, California, USA, 2019.
- [19] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer VAE: A Hierarchical Model for Structure-Aware and Interpretable Music Representation Learning," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 516–520.
- [20] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer," in *Proc.*

of 19th International Society for Music Information Retrieval Conference (ISMIR), Paris, France, 2018.

- [21] Y.-N. Hung, I.-T. Chiang, Y.-A. Chen, and Y.-H. Yang, “Musical composition style transfer via disentangled timbre representations,” in *Proc. of 28th International Joint Conference on Artificial Intelligence (IJCAI)*, Macao, China, 2020.
- [22] Y.-J. Luo, K. Agres, and D. Herremans, “Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders,” in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.
- [23] G. Hadjeres, F. Nielsen, and F. Pachet, “GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures,” in *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI)*, Hawaii, USA, 2017, pp. 1–7.
- [24] A. Pati and A. Lerch, “Latent space regularization for explicit control of musical attributes,” in *Proc. of ICML Workshop on Machine Learning for Music Discovery Workshop (MLAMD), Extended Abstract*, Long Beach, California, USA, 2019.
- [25] H. H. Tan and D. Herremans, “Music FaderNets: Controllable music generation based on high-level features via low-level feature modelling,” in *Proc. of 20th International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.
- [26] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, “Disentangling factors of variations using few labels,” in *Proc. of 8th International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.
- [27] A. Pati, S. Gururani, and A. Lerch, “dMelodies: A Music Dataset for Disentanglement Learning,” in *Proc. of 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.
- [28] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” in *Proc. of 2nd International Conference on Learning Representations (ICLR)*, Banff, Canada, 2014.
- [29] T. Adel, Z. Ghahramani, and A. Weller, “Discovering Interpretable Representations for Both Deep Generative and Discriminative Models,” in *Proc. of 35th International Conference on Machine Learning (ICML)*, Stockholm, Sweden, 2018, pp. 50–59.
- [30] A. Pati and A. Lerch, “Attribute-based Regularization of Latent Spaces for Variational Auto-Encoders,” *Neural Computing and Applications*, 2020. [Online]. Available: <https://doi.org/10.1007/s00521-020-05270-2>
- [31] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, “beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework,” in *Proc. of 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.
- [32] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. of 3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, 2015.
- [33] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, “Isolating Sources of Disentanglement in Variational Autoencoders,” in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Montréal, Canada, 2018.
- [34] K. Ridgeway and M. C. Mozer, “Learning Deep Disentangled Embeddings With the F-Statistic Loss,” in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, Montréal, Canada, 2018, pp. 185–194.
- [35] A. Kumar, P. Sattigeri, and A. Balakrishnan, “Variational Inference of Disentangled Latent Concepts from Unlabeled Observations,” in *Proc. of 5th International Conference of Learning Representations (ICLR)*, Toulon, France, 2017.
- [36] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, “Fader Networks: Manipulating Images by Sliding Attributes,” in *Advances in Neural Information Processing Systems 31 (NeurIPS)*, Long Beach, California, USA, 2017, pp. 5967–5976.
- [37] L. Kawai, P. Esling, and T. Harada, “Attributes-aware deep music transformation,” in *Proc. of 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020.
- [38] P. Xu, J. C. K. Cheung, and Y. Cao, “On variational learning of controllable representations for text without supervision,” in *Proc. of 37th International Conference on Machine Learning (ICML)*, 2020.
- [39] M. Connor and C. Rozell, “Representing Closed Transformation Paths in Encoded Network Latent Space,” in *Proc. of 34th AAAI Conference on Artificial Intelligence*, New York, USA, 2020.
- [40] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, “Understanding and improving interpolation in autoencoders via an adversarial regularizer,” in *Proc. of 7th International Conference on Learning Representations (ICLR)*, New Orleans, USA, 2019.

PULSE CLARITY METRICS DEVELOPED FROM A DEEP LEARNING BEAT TRACKING MODEL

Nicolás Pironio¹

Diego Fernández Slezak^{1,2}

Martín A. Miguel^{1,2}

¹ Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales.

Departamento de Computación. Buenos Aires, Argentina.

² CONICET-Universidad de Buenos Aires. Instituto de Investigación en Ciencias de la Computación (ICC). Buenos Aires, Argentina.

npironio@dc.uba.ar

ABSTRACT

In this paper we present novel pulse clarity metrics based on different sections of a state-of-the-art beat tracking model. Said model consists of two sections: a recurrent neural network that estimates beat probabilities for audio and a dynamic Bayesian network (DBN) that determines beat moments from the neural network's output. We obtained pulse clarity metrics by analyzing periodical behavior from neuron activation values and we interpreted the probability distribution computed by the DBN as the model's certainty. To analyze whether the inner workings of the model provide new insight into pulse clarity, we also proposed reference metrics using the output of both networks. We evaluated the pulse clarity metrics over a wide range of stimulus types such as songs and mono-tonal rhythms, obtaining comparable results to previous models. These results suggest that adapting a model from a related task is feasible for the pulse clarity problem. Additionally, results of the evaluation of pulse clarity models on multiple datasets showed that, with some variability, both ours and previous work generalized well beyond their original training datasets.

1. INTRODUCTION

In music, the pulse refers to the underlying regular rhythmic pattern in a song, usually expressed by listeners by tapping their foot. In western notation, the pulse takes on an especially relevant role, given that location and duration of rhythmic events are described with respect to it. Listeners can extract a pulse from the acoustic surface and infer a meter structure that may enable them to adjust their own behaviour to it (e.g. dance accordingly to a song) [1].

The strength with which the feeling of the pulse

emerges in a listener is not necessarily the same for all music. The concept of pulse clarity refers to such subjective experience. As the pulse is relevant for temporal organization, pulse clarity facilitates a listener's understanding of a song, affecting the musical experience. Musical cognition experiments have used pulse clarity as a high-level musical feature, usually correlating it to human responses. For example, it has been related with degree and variability of movement [2, 3], as well as with specific neural responses to different musical stimuli [4]. Pulse clarity has been seen to be influenced by different rhythmic structure characteristics (e.g. syncopation), as it affects participant's beat tapping variability [5–7].

In the mentioned experiments, pulse clarity is estimated from the musical input using the model from Lartillot et al. [8]. In their work, the authors present various descriptors for audio recordings based on the analysis of an onset detection curve and the periodicities it presents. The model consists of the best descriptor, which was selected based on experimental results where participants rated the pulse clarity of movie soundtrack excerpts. Miguel et al. [9] proposed another pulse clarity model for symbolic representations of rhythmic passages. Said model outputs a beat congruence score over time which is interpreted as a pulse clarity metric. The model's score was evaluated by comparing it to human beat tapping variability data over songs and achieved comparable results to the Lartillot et al. [8] model.

A closely related problem in the Music Information Retrieval discipline is the beat tracking task, which consists of determining the pulse moments for a musical excerpt [10]. This task has a long and varied history of models, with most recent and proficient ones making use of novel techniques such as deep learning. Since pulse clarity can be thought of as the difficulty of performing beat tracking, our work proposes a transfer learning approach using data from beat tracking models to estimate pulse clarity in musical excerpts [11]. Here we develop a methodology for interpreting the beat tracking deep learning architecture presented by Krebs et al. [12] and Bock et al. [13] which has proved its effectiveness on the beat tracking task. This



© N. Pironio, D. Fernández Slezak, and M. Miguel. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** N. Pironio, D. Fernández Slezak, and M. Miguel, "Pulse clarity metrics developed from a deep learning beat tracking model", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

family of models was selected because of its state-of-the-art performance and its use of a dynamic Bayesian network, which estimates probabilities of different beat interpretations. We theorized these estimations to be useful to approximate pulse clarity.

In this exploratory analysis, we propose a series of pulse clarity metrics based on different sections from the architecture. We evaluated the proposed pulse clarity metrics by calculating the Spearman rank correlation coefficient against pulse clarity interpretations from three empirical datasets: the movie soundtrack excerpts used in [8], where pulse clarity was self reported, the musical excerpts from the MIREX Beat tracking train dataset, where pulse clarity was calculated from variability in the tapping data, and the rhythmic passages from Miguel et al. [14] where pulse clarity was both reported by participants and calculated from the tapping data. We also present results for the Lartillot et al. [8] and Miguel et al. [9] models over the datasets as reference. Evaluation results show that there is relevant information for the pulse clarity problem in the associated beat tracking task as our metrics performed as well as previous pulse clarity models.

In the next section the general architecture from Krebs et al. [12] and Bock et al. [13] is presented, as well as the corresponding developed interpretations. The evaluation section describes the datasets in detail, explains how pulse clarity was estimated from tapping data and shows the evaluation methodology and the obtained results. We conclude by arguing that, based on our results, repurposing a related task model is reasonable for pulse clarity estimation and suggest further evaluation of the existing models.

2. PULSE CLARITY MODEL

In this section we briefly review the model architectures presented by Krebs et al. [12] and Bock et al. [13] for the beat tracking task, highlighting the key features relevant for the pulse clarity metrics we derived. We then describe each metric proposed, categorized by which aspects of the beat tracking model were considered for its definition.

2.1 Beat tracking model

The beat tracking architecture consists of three steps: audio preprocessing, estimation of beat probability for a given audio frame, and selecting beat moments given the beat probabilities. Bock et al. [13] presents a modification to the original architecture from Krebs et al. [12] to also take into account downbeat tracking. As these are fairly similar in respect to their architecture, all metrics developed are implemented from both the beat and downbeat models. Here we describe the downbeat model from Bock et al. [13] (as it is implemented by the authors in the `madmom` package [15], version 0.16.1) and clarify the significant differences between the two models when necessary.

In the audio preprocessing stage, different magnitude spectrograms are computed from the audio signal. These in turn are used as the input for a neural network ensemble. Each network in the ensemble is a recurrent neu-

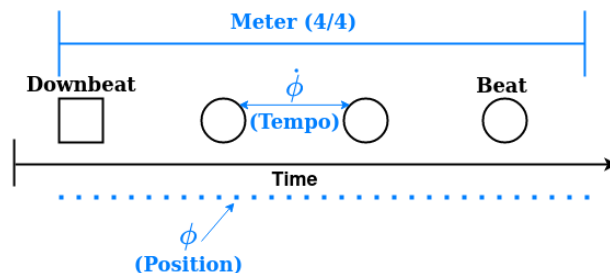


Figure 1. Example 4/4 bar with the position, tempo and meter state space variables associated.

ral network (RNN) that estimates the beat probability of an audio frame. These RNN have three hidden recurrent bidirectional layers, each with 25 long-short term memory (LSTM) cells [16]. The network’s output consists of three neurons with a softmax activation function, representing the probability distribution over the *beat*, *downbeat* and *no beat* classes for a given audio frame. In the case of the Krebs et al. model, the probability distribution represents only the *beat* and *no beat* classes. The final beat activation function of the ensemble is computed as the average between each individual network’s output.

Lastly, a dynamic Bayesian network (DBN) is used to determine the sets of beat and downbeat moments, given the probabilities output by the RNN ensemble. Conceptually, the sequence of audio frames is associated with a Markov chain of latent variables and the output of the RNN is used as the observations. The latent variables state space consists of a set of possible bar positions, tempi and time signatures - only the first two variables are considered in the Krebs et al. [12] model. Using the Viterbi algorithm, the most probable sequence of variables is determined and from it the sets of beats and downbeats moments are extracted.

2.2 Dynamic Bayesian Network based metrics

Compared to a deep learning architecture, the Bayesian dynamic network has a clearer interpretation. This is most notable by the use of a state space and transition model that encodes knowledge of the task at hand. In the analyzed model, the state space S of the DBN encodes the position within the bar (ϕ), the tempo ($\dot{\phi}$) and the time signature for each audio frame (3/4, 4/4). In Figure 1 we depict how these variables are related to an example 4/4 bar.

In a DBN, scores proportional to the probability of the most likely state sequences are calculated using the Viterbi algorithm. We define D_s as the scores for full sequences (analyzing the entire input) for each possible ending state s . From this distribution we make two possible interpretations of pulse clarity. The first one considers the probability estimate of the most probable state, $\max(D_s)$, which we interpret as the level of confidence with which the last variable is determined. We name this metric **Viterbi max**. The second aims to capture the uncertainty over D_s by computing its entropy: $H(D_s)$. We call this metric **Viterbi entropy**. As entropy is lower for a more concentrated dis-

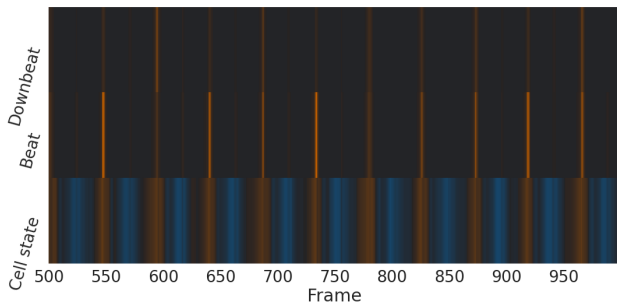


Figure 2. Average cell state activation for the last layer neurons in a 5 second excerpt of a song, contrasted with the beat activation function of the network. Orange colors are positive values, while blue colors are negative.

tribution, a low value for this metric can be interpreted as a clearer decision made by the DBN.

2.3 Recurrent network based metrics

We decided to further explore whether the activations of the recurrent neural networks could be used to estimate pulse clarity. In this section of the architecture, we lacked a clear interpretation of the inner states of the deep learning model. Yet, given that the pulse of a song is an inherently periodical pattern, we hypothesized the RNN would present periodical activations at different levels of the network. When observing the network activations, these patterns were present in the form of activation peaks. With this in mind, we analyzed these activations from three different points of view, considering only a single trained network from the ensemble for simplicity.

Firstly, we considered the cell state values, which act as an "internal memory" of the LSTM cells, for each audio frame. Specifically, for each frame, the mean cell state activation of the 50 neurons in the last hidden layer is computed, separating the positive and negative values into separate series. Then, for each series, a peak picking process is performed, where the width for every peak is obtained. Averaging these widths results in the **Cell state precision** metric. In Figure 2 we can observe how the peak values of the average cell state activations for the last layer neurons tend to align to the final output of the network. The average width was interpreted as the confidence with which the network determines the beat probability for a frame: the wider a peak is, the lower the certainty.

Now focusing on capturing periodicity at a higher level, we turn to consider the cell activations (the output of the LSTM's) through time. Figure 3 shows the output series for a subset of neurons, in which periodical activation patterns can be interpreted for each neuron independently. This behavior motivated looking into possible periodicities between the output series of different cells and each cell with itself.

In the case of considering different series, for each pair of neurons i, j , the maximum absolute correlation between the output series o_i, o_j value is computed, considering all possible lags. We define the **neurons cross correlation** as

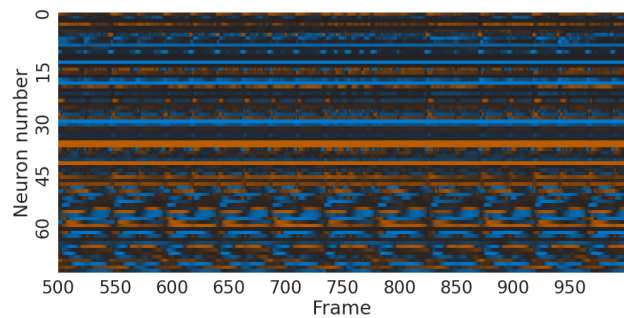


Figure 3. Sample output values of the forward layers neurons for a 5 second excerpt of a song.

the sum of each of these values, as depicted in Equation 1. With this metric we aim to determine the degree of coordination between neurons.

$$\text{NCC} = \sum_{o_i \neq o_j} \max_{lag \in [0, |o_i|]} |corr(o_i, o_j, lag)| \quad (1)$$

When considering each series against itself, we define **autocorrelation periodicity** as the average of the maximum autocorrelation values for each cell (Equation 2). These maximum autocorrelation values are obtained considering only lags representing periodicities between 40 and 330 BPM. Each autocorrelation value is divided by the size of the signal overlap, considering the lag. Compared to the neuron's cross correlation metric, the autocorrelation periodicity is less strict, as it only tries to capture if every neuron's output has a periodic pattern with itself.

$$\text{ACP} = \frac{1}{N} \sum_i \max_{lag \in L} \frac{A_{i,lag}}{\text{overlap}(o_i, lag)} \quad (2)$$

Where:

$$\begin{aligned} A_{i,lag} &= |corr(o_i, o_i, lag)| \\ \text{overlap}(o_i, lag) &= |o_i| - lag \\ L &= [40bpm, 330bpm] \end{aligned} \quad (3)$$

2.4 Output-based metrics

As the intention was to determine if there was relevant information from within the model for the pulse clarity task, we chose to develop reference metrics based on the output both from the RNN and the DBN. Subsequently, we analyze if the metrics derived from the inner workings of the networks surpass the performance of the output-based metrics. Using the beat activation function output by the RNN, two interpretations of pulse clarity were computed. First, we consider the average probability for beat moments as an indication of the overall certainty of the output. The **peak average** is obtained by applying a peak picking process to the beat probability signal and then averaging the peak probabilities. Second, using the previous calculated peaks as beat moments, we define the **RNN entropy** as the entropy of the inter-beat interval distribution. This concept is the same as the one used in the calculations of tapping variability presented in the Evaluation section. Analogously,

using the beat moments outputted by the DBN, we compute the **DBN entropy** as the entropy over the inter-beat interval distribution.

3. EVALUATION

In this section we evaluate the performance of the proposed metrics. We will present the considered datasets in detail, which vary in their type of stimuli and annotations and, as such, we clarify the pulse clarity interpretations of the annotated data specific to each one. Then the evaluation process is described, in which for each pulse clarity metric the absolute Spearman rank correlation coefficient is computed against each dataset. Using this coefficient as a performance score, a ranking of metrics is obtained.

3.1 Datasets

Three datasets were used for evaluation, which we named the MIREX, rhythms and soundtracks datasets. The MIREX dataset [17] has its origins in the beat tracking task, developed for the evaluation of models, with the intent to compile difficult-to-track musical excerpts. It consists of 30-second excerpts of 20 varied style songs. These excerpts have a stable tempo and present a varied distribution of tempi values. 40 beat annotations are available for each song.

The *rhythms* dataset was developed with the purpose of capturing the subjective pulse experience. To this end, we carried out an experiment where participants listened to 33 rhythmic passages of varying rhythmic complexity and were instructed to tap to a self selected beat. Participants were allowed to stop tapping if the beat was not clear enough or change their selected beat mid-trial. After each trial, participants rated how difficult the tapping task was with values between 1 (easy) and 5 (hard). The stimuli consisted of 11 rhythms from [14], 7 from [15], 5 were isochronous beats at 150, 200, 250, 500, 800 ms inter-beat intervals and 10 were new. 7 of the new stimuli were presented in increasing complexity order at the beginning of the experiment to familiarize the participants with the task. All other stimuli were randomized. With the exception of the isochronous stimuli, presentation inter-beat intervals varied between 450 and 550 ms avoiding having the same IBI in two consecutive trials. Each stimulus consisted of repeating a short rhythm the number of times required to last a minimum of 24 seconds. From 35 total participants, 30 remained after filtering participants that were deemed to not understand the concept of beat. They were selected as participants who replicated the stimulus instead of defining a beat in more than three trials. 6 participants were female, and 26 were male. Overall average age was 28.27 (sd = 7.94) and overall mean musical training was 4.85 years (sd = 3.90). For our evaluation, we will not consider the 5 isochronous stimuli in the dataset as these were not intended to evaluate rhythmic complexity.

Lastly, we use the *soundtracks* (ST) dataset used in [8], which is composed of 100 five-second excerpts of movie soundtracks, selected to cover a wide range of pulse clar-

ity scenarios. From these, 15 excerpts were discarded as some metrics couldn't be computed for them because they provided too few beat events. Each track was rated by 25 musically trained participants in its beat clarity on a scale from 1 to 9, labeled from "unclear" to "clear". The mean clarity score is provided in the dataset [18].

Tracks in the MIREX and *rhythms* datasets have more than one annotation for each track. To obtain a single value for each track and category, the empiric pulse clarity value for a track is considered as the mean response of the subjects. Previously, pulse clarity values (tapping variability and self-reported) were z-standardized within participants.

As there are various types of annotations, we consider different interpretations of pulse clarity for each dataset. For the *rhythms* dataset we consider the answers to the tapping difficulty question and in the *soundtracks* dataset we use the "pulse clarity" reported answers. For both the MIREX and Experimental datasets, human tapping annotations are available. These consist of a list of moments in time where the person felt the underlying pulse. Using this information, we propose a tapping variability metric, "inter-tap-interval entropy" (ITI-E), to act as a proxy for pulse clarity. The computation is as follows: first the difference between subsequent taps is obtained. These differences are considered samples from the underlying subject's inter-tap interval distribution. A Gaussian kernel density estimator with a bandwidth of 5ms is fitted over the samples and 400 equidistant points considered from the range of 8 and 320 BPM are evaluated to obtain the density estimation. Lastly, the entropy over the density estimation is calculated as a means to capture its variability. In trials where less than five taps were produced, the metric was not considered reliable. For these cases, it was considered the participant had an unclear pulse precept and decided not to tap. The entropy value was replaced with the maximum entropy found for the participant. This methodology was verified by correlating the obtained values for the *rhythms* dataset and the reported tapping difficulty answers, obtaining an r coefficient of 0.81 with $p < 0.001$.

Calculating the entropy over the distribution as opposed to an approach based on standard deviation calculation was chosen because we consider the possibility that the distribution could be multi-modal, meaning that a subject tapped to two or more possible interpretations of the pulse in the same stimulus. In said case, the standard deviation would result in high variability, when in fact it could be argued that the tapping was precise for more than one pulse interpretation.

3.2 Model selection

For the evaluation of the proposed metrics we classify them in categories and select the best scoring metric in each one. The categories considered were DBN metrics, RNN metrics, output metrics and comparison metrics. This last one is comprised of the pulse clarity model proposed by Lartillot in [8] (`mirpulseclarity` function from MIR-Toolbox 1.7.2 [19] with parameters for `model=1`), the congruency score from the THT model presented in [9]

	MIREX	Rythms		ST	Avg
	ITI-E	Conf	ITI-E	PC	
Comp	0.550 [†]	0.783	0.690	0.570	0.648
DBN	0.412 [‡]	0.912	0.775	0.860	0.740
Out	0.791	0.858	0.769	0.388 [†]	0.701
RNN	0.632	0.627	0.621	0.372 [†]	0.563

Table 1. Test set scores for each category in every dataset. The selected metrics in the training process were: **THT-congruency** (Comp), **Viterbi max** (beat version) (DBN), **DBN entropy** (beat version) (Out), **cell state precision** (RNN). All correlations have p-values below 0.001, except for those with [†] where $0.05 > p > 0.001$, and [‡] where $0.1 > p > 0.05$.

and an additional interpretation of it called **THT entropy**, which is the same as **DBN entropy** but using the beat moments outputted by the THT model. These categories allow the evaluation of the two main questions this work proposed. The first one being if the interpretations made of the beat tracking model are comparable to the previous pulse clarity models. Furthermore it is of interest to know if the metrics derived from internal behaviour surpass the ones obtained by interpreting the output of the deep learning model.

We separate the data into train and test subsets to select the best metric per category. The train set is obtained by randomly selecting half of the *soundtracks* dataset. Selecting train data only from this dataset is motivated by the fact that the other two have few stimuli and partitioning them would increase the probability of losing representability in the subsets. In the training process we selected one metric from each category as the one that scored consistently higher when considering the correlation on 10 subsets of 80% of the training data. In Table 1, we report the test set scores for each metric selected in the training process.

From Table 1 we can observe that there is not a best model overall. Nonetheless it is remarkable that in two of the three datasets the DBN metrics section has the highest score, achieving r values over 0.77 with the **Viterbi Max** metric (2nd row on Table 1). Averaging all test set scores, this metric performed best. Comparing against the Comparison section, no proposed metric provides better results over all datasets. When comparing the metrics from the inner behaviour of the model versus those calculated using the output, neither the RNN or DBN categories surpass the Output metric selected in all datasets. This category was the second with highest average score with the **DBN entropy** metric, providing similar results to the **Viterbi max** metric.

4. CONCLUSIONS

In this paper we showed possible interpretations of pulse clarity from the inner workings of a beat tracking model. The developed metrics achieved comparable results with

respect to previous works over distinct datasets, showing that there is relevant information in the analyzed beat tracking models. Specifically, the intuitive DBN based metrics performed considerably better compared to the RNN metrics. Nevertheless, when comparing the inner calculations of the model with simple transformations of the model’s output, the inner calculations did not consistently yield better results. This indicates that, although useful, inspecting the inner behavior of the model may not be strictly necessary.

We evaluated the pulse clarity models on very different stimulus types: songs, rhythms and movie soundtracks. Results showed that both the developed and comparison models performed well on all datasets, even on those that had different types of stimulus than their original training data. This indicates that models can generalize well from one type of stimulus to another. Yet, some variability was present across datasets, inviting further evaluation of pulse clarity models on a broader spectrum of musical stimuli.

The pulse clarity metrics obtained from the beat tracking model presented correlations comparable with those of previous work. Given a more extensive dataset, with more musical styles and exploring more dimensions of pulse clarity, relevant pulse clarity predictors can be obtained. These would provide new tools for the musical psychology community. Overall, we propose that developments in models for music perception tasks can be repurposed for related tasks.

5. OPEN PRACTICES STATEMENT

A reference implementation written in Python is publicly available at <https://github.com/nPironio/maipc>, including all the metrics presented in this work.

6. REFERENCES

- [1] W. T. Fitch, “Rhythmic cognition in humans and animals: distinguishing meter and pulse perception,” *Frontiers in Systems Neuroscience*, vol. 7, 2013. [Online]. Available: <https://doi.org/10.3389%2Ffnys.2013.00068>
- [2] V. E. Gonzalez-Sanchez, A. Zelechowska, and A. R. Jensenius, “Correspondences between music and involuntary human micromotion during standstill,” *Frontiers in Psychology*, vol. 9, p. 1382, 2018. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fpsyg.2018.01382>
- [3] B. Burger, M. R. Thompson, G. Luck, S. Saarikallio, and P. Toiviainen, “Music Moves Us: Beat-Related Musical Features Influence Regularity of Music-Induced Movement,” no. July, 2012, pp. 183–187. [Online]. Available: http://icmpc-escom2012.web.auth.gr/sites/default/files/papers/183_Proc.pdf
- [4] W. Trost, S. Frühholz, T. Cochrane, Y. Cojan, and P. Vuilleumier, “Temporal dynamics of musical emotions examined through intersubject synchrony

- of brain activity,” *Social Cognitive and Affective Neuroscience*, vol. 10, no. 12, pp. 1705–1721, 05 2015. [Online]. Available: <https://doi.org/10.1093/scan/nsv060>
- [5] W. T. Fitch and A. J. Rosenfeld, “Perception and production of syncopated rhythms,” *Music Perception: An Interdisciplinary Journal*, vol. 25, no. 1, pp. 43–58, 2007.
- [6] B. H. Repp and Y.-H. Su, “Sensorimotor synchronization: A review of recent research (2006–2012),” *Psychonomic Bulletin & Review*, vol. 20, no. 3, pp. 403–452, Jun 2013. [Online]. Available: <https://doi.org/10.3758/s13423-012-0371-2>
- [7] A. D. Patel, J. R. Iversen, Y. Chen, and B. H. Repp, “The influence of metricality and modality on synchronization with a beat,” *Experimental Brain Research*, vol. 163, no. 2, pp. 226–238, May 2005. [Online]. Available: <https://doi.org/10.1007/s00221-004-2159-8>
- [8] O. Lartillot, T. Eerola, P. Toiviainen, and J. Fornari, “Multi-feature modeling of pulse clarity: Design, validation, and optimization,” in *Proceedings of the International Symposium on Music Information Retrieval*, 2008.
- [9] M. A. Miguel, M. Sigman, and D. Fernandez Slezak, “From beat tracking to beat expectation: Cognitive-based beat tracking for capturing pulse clarity through time,” *PLOS ONE*, vol. 15, no. 11, pp. 1–22, 11 2020. [Online]. Available: <https://doi.org/10.1371/journal.pone.0242207>
- [10] M. MCKINNEY, D. Moelants, M. DAVIES, and A. KLAPURI, “Evaluation of audio beat tracking and music tempo extraction algorithms,” *JOURNAL OF NEW MUSIC RESEARCH*, vol. 36, no. 1, pp. 1–16, 2007. [Online]. Available: <http://dx.doi.org/10.1080/09298210701653252>
- [11] K. Weiss, T. M. Khoshgoftaar, and D. Wang, “A survey of transfer learning,” *Journal of Big Data*, vol. 3, no. 1, p. 9, May 2016. [Online]. Available: <https://doi.org/10.1186/s40537-016-0043-6>
- [12] F. Krebs, S. Böck, and G. Widmer, “An Efficient State-Space Model for Joint Tempo and Meter Tracking.” in *Proceedings of the 16th International Society for Music Information Retrieval Conference*. Málaga, Spain: ISMIR, Oct. 2015, pp. 72–78. [Online]. Available: <https://doi.org/10.5281/zenodo.1414966>
- [13] S. Böck, F. Krebs, and G. Widmer, “Joint beat and downbeat tracking with recurrent neural networks,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, M. I. Mandel, J. Devaney, D. Turnbull, and G. Tzanetakis, Eds., 2016, pp. 255–261. [Online]. Available: https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/186_Paper.pdf
- [14] M. Miguel, M. Sigman, and D. F. Slezak, “Tapping to your own beat: experimental setup for exploring subjective tacti distribution and pulse clarity,” Oct 2019. [Online]. Available: osf.io/7sqaw
- [15] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new python audio and music signal processing library,” in *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016, pp. 1174–1178.
- [16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [17] “Mirex beat tracking training dataset.” 2006, https://www.music-ir.org/mirex/wiki/2019:Audio_Beat_Tracking.
- [18] T. Eerola, “Ground-truth data for selected perceptual features,” Jul 2020. [Online]. Available: osf.io/6wqh5
- [19] O. Lartillot and P. Toiviainen, “A matlab toolbox for musical feature extraction from audio,” in *International conference on digital audio effects*, vol. 237. Bordeaux, 2007, p. 244.

ON THE VERACITY OF LOCAL, MODEL-AGNOSTIC EXPLANATIONS IN AUDIO CLASSIFICATION: TARGETED INVESTIGATIONS WITH ADVERSARIAL EXAMPLES

Verena Praher^{1*}

Katharina Prinz^{1*}

Arthur Flexer¹

Gerhard Widmer^{1,2}

¹ Johannes Kepler University Linz, Austria

² LIT AI Lab, Linz Institute of Technology, Austria

{verena.praher, katharina.prinz}@jku.at

ABSTRACT

Local explanation methods such as LIME have become popular in MIR as tools for generating post-hoc, model-agnostic explanations of a model’s classification decisions. The basic idea is to identify a small set of human-understandable features of the classified example that are most influential on the classifier’s prediction. These are then presented as an explanation. Evaluation of such explanations in publications often resorts to accepting what matches the expectation of a human without actually being able to verify if what the explanation shows is what really caused the model’s prediction. This paper reports on targeted investigations where we try to get more insight into the actual veracity of LIME’s explanations in an audio classification task. We deliberately design adversarial examples for the classifier, in a way that gives us knowledge about which parts of the input are potentially responsible for the model’s (wrong) prediction. Asking LIME to explain the predictions for these adversaries permits us to study whether local explanations do indeed detect these regions of interest. We also look at whether LIME is more successful in finding perturbations that are more prominent and easily noticeable for a human. Our results suggest that LIME does not necessarily manage to identify the most relevant input features and hence it remains unclear whether explanations are useful or even misleading.

1. INTRODUCTION

With the rise of deep learning methods used in Music Information Retrieval (MIR), also the desire for explaining the decisions made by such models has increased. A plethora of explanation methods (“explainers”) have been originally developed for text or image data and adapted to the audio domain [1, 2], or specifically introduced for MIR

systems [3]. Most notably, different versions of Local Interpretable Model-agnostic Explanations (LIME), a post-hoc explainer [4], have been used to explain models in a variety of MIR tasks [5–11].

Evaluation of explanations is known to be a hard task, due to a lack of agreement on what constitutes a “good explanation” [12] and, consequently, what may serve as ground truth for measuring the quality of an explanation. Explanations are often evaluated visually and the applied “metric” is whether the highlighted input parts appear relevant to the human observer (“confirmation bias”) [13].

We propose evaluating local explanations by exploiting a known weakness of deep neural networks: adversarial examples. By adversarially perturbing examples in a way that a system’s original prediction is changed, we know exactly what in the input caused the erroneous (*adversarial*) prediction and can use this as the “ground truth” that the explanation method should recover.

We are not the first to investigate the relationship between adversarial examples and interpretability. Slack et al. [14] show that a racially biased model can be attacked in a way that the (biased) prediction remains unchanged but the explanation appears as if the model did not base the prediction on sensitive attributes. This work is related in the sense that it highlights weaknesses of post-hoc explanation methods (including LIME).

Closest to our work are Göpfert et al. [15], who use “localised” adversarial attacks that target only selected segments of an image, and investigate the ability of different explainers (including LIME) to recover the affected part. In their experiments LIME outperforms the other explainers, which supports our choice of LIME as an explainer that deserves a more careful and critical investigation.

We perform four experiments with purposefully designed adversarial examples in order to obtain more insight about strengths and weaknesses of LIME-based explanations in an audio classification task. As a test bed, we have chosen a DNN-based singing voice detection model [16] that currently defines the state of the art on that task, and is suited to study the quality of explanations for several reasons: it addresses a clearly defined task (as opposed to genre classification for example), it has been used for demonstrating explanations before [5, 11], and it has the interesting previously documented property that it can be confused by directly drawing on the spectrogram [17].

* Equal contribution.



2. LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS

LIME is an algorithm for explaining individual predictions (*local*) for any black box machine learning model (it is *model-agnostic*). The general idea of the algorithm is to train a simpler model g that approximates the neighborhood of the prediction $f(x)$ that we want to explain. The first step is to derive an interpretable representation depending on the input domain, e.g. the presence or absence of super pixels¹ in the image domain.

Let $x \in \mathbb{R}^d$ (in the case of spectrograms, $d = T \times F$) be the representation in the original input domain and $x' \in \{0, 1\}^{d'}$ be the interpretable representation where 0 and 1 denote the absence and presence of an interpretable feature, respectively. In the next step we sample N_s instances around x' by randomly generating vectors of length d' containing 0's and 1's. Each generated instance z' is mapped back to the input domain and fed through the original model f . $z \in \mathbb{R}^d$ looks like the original input example with all parts that are 0 in z' occluded (e.g. gray values in the image domain). The N_s instances z' and the corresponding predictions $f(z)$ constitute the input to the explanation model g [4]. The most commonly used explainers are linear models of the form²

$$g(z') = b + w_g z'. \tag{1}$$

When training the linear explanation model, the generated instances are weighted via an exponential kernel decaying with increasing distance to the input example. This distance (and based on it the weight) is computed between the binary representation of the examples being explained (all 1's) and each of the instances in the neighborhood³.

A requirement for an explanation is that it be *locally faithful*: we expect the explainer g to approximate f in the neighborhood of x . The faithfulness is measured by the fidelity score, which is computed by the coefficient of determination (R^2) of the linear model's predictions [4], and ranges from 0 to a perfect score of 1, with the exact interpretation of the range of values being somewhat unclear.

Different flavours of LIME have been used for explaining predictions in a variety of audio classification and tagging tasks. The general LIME algorithm was kept the same and the difference is the type of interpretable features. Mishra et al. [5] proposed the first adaption of LIME for MIR, termed SoundLIME (SLIME), which segments the spectrogram into time, frequency, or time-frequency segments. SLIME was demonstrated on the task of singing voice detection [18] and used for analysing a replay spoofing detection system [6]. Haunschmid et al. used other types of interpretable features (super pixels [7], source sep-

aration estimates [8, 9]) for explaining the predictions of a variety of models, including music taggers [8, 9] and a content-based music recommender system [10]. Mishra et al. [11] proposed different content types for replacing the “grayed out” segments (e.g. *zero*, the *min* and the *mean* value of the spectrogram) and found the mean value to work best in their setting. For our experiments we will be using time-frequency segments [5, 11]. The hyperparameters for the LIME algorithm are described in Section 4.4.

3. ADVERSARIAL ATTACKS

In this section, we briefly describe a way to compute adversarial examples for a singing voice detection system. The goal is to obtain perturbations which do not affect how a human perceives a particular example, yet simultaneously change the prediction of the system, i.e., transform “Singing Voice” to “No Singing Voice” or vice versa. To realise this, we use an adversarial attack that was originally proposed for audio [19], and was previously applied in MIR to attack an instrument classification system and a music recommender [20]. The attack, subsequently denoted by Carlini & Wagner (C&W), assumes a white-box scenario, i.e. full knowledge about a model and its parameters.

As C&W is an iterative targeted attack, we use it to decrease the loss of a system with respect to a new (wrong) target prediction, in multiple iterations. Following the notation in [20], let f be a system and $x \in \mathbb{R}^d$ its input, i.e., a specific prediction is denoted by $f(x)$. Also, let $\delta_{ep} \in \mathbb{R}^d$ be an adversarial perturbation at iteration ep , and an adversarial example $x + \delta_{ep}$. Furthermore, we denote the system loss by L_{sys} , and the target prediction by t . To restrict the adversarial perturbation, C&W minimises a weighted sum of the squared L2-norm of δ and a system loss, resulting in an optimisation objective as follows [19, 20]:

$$L_{total} = \|\delta_{ep}\|_2^2 + \alpha * L_{sys}(f(x + \delta_{ep}), t),$$

$$\delta_{ep+1} = \text{clip}_\epsilon(\delta_{ep} - \eta * \text{sign}(\nabla_{\delta_{ep}} L_{total})). \tag{2}$$

Note that $\nabla_{\delta_{ep}}$ is the gradient w.r.t. to δ_{ep} , and updates are performed based on the sign of the gradient and the multiplicative factor η . By applying clip_ϵ , an adversarial perturbation is always in the range of $[-\epsilon, \epsilon]$, and α is used to balance the magnitude of the perturbation and the system loss w.r.t. the target prediction.

In the following experiments, we will compute adversarial examples both for raw audio waveforms and for input spectrograms. In either case, we will always analyse the resulting perturbations in the time-frequency domain. For adversaries computed for the waveform we compute the time-frequency representation and subtract the original example to obtain the adversarial perturbation in the time-frequency domain. For adversarial examples computed for the spectrogram directly, time-frequency information is already available.

¹ (perceptually) grouped pixels

² In the original paper [4], the explainer had the form $g(z') = w_g z'$, mistakenly omitting the intercept b . Inspecting the source code shows that the intercept was trained as well: <https://github.com/marcotcr/lime/>

³ The original paper [4] stated that the distance for weighting the examples was computed between the images directly, but the source code shows that it is actually computed between the binary representations, which was later confirmed by the author.

4. EXPERIMENTAL SETUP

This section describes the experimental setup, including the data we use, details about the singing voice detection system and hyperparameters for both the adversarial attacks and computing the explanations.

4.1 Data

To train the singing voice detection system proposed by Schlüter and Lehner, we use a subset of the data used in their work [16], namely the openly available *Jamendo* dataset [21]. The dataset consists of 93 songs totalling around 6 hours of music, with a proposed training / validation / test split of 61 / 16 / 16 files respectively. Each audio file has a sampling rate of 44.1kHz.

The annotations for the Jamendo dataset are provided with sub-second granularity [18], and indicate the presence / absence of singing voice. The proportion of singing voice (“sing”) in comparison to non-singing voice (“no sing”) is close to 50 : 50 in all three data splits.

4.2 Singing Voice Detector

For subsequent experiments, we adapt the singing voice detection system introduced by Schlüter and Lehner [16]. We use the proposed architecture of the Convolutional Neural Network (CNN) and deploy the training routine with unaltered hyperparameters on the Jamendo dataset. Prior to training, the data is resampled to 22.05kHz and then used to compute magnitude Mel spectrograms (frame length 1024, hop size 315 samples and 80 Mel bands) [16]. The Mel spectrograms are logarithmically scaled and normalised per frequency band to have zero mean and unit variance over the training data.

The CNN is trained on spectrogram excerpts with a length of 115 frames, for which the target prediction corresponds to the ground-truth annotation for the central frame [16]. During training, we use a mini-batch size of 32 and Adam to optimise the Binary Cross Entropy loss for 40.000 update steps. We start with a learning rate of 0.001 and scale it by 0.85 every epoch. To support generalisation, dropout and data augmentation [18] is used; for a more detailed description of the training procedure, hyperparameters and the CNN architecture, we refer to [16].

The final classification error of the singing voice detector (in percent) on the Jamendo test data, given as the mean \pm standard deviation over 5 different runs, is 11.54 ± 0.96 . Furthermore, recall and specificity over 5 runs are 89.61 ± 1.71 and 87.46 ± 1.00 respectively. The metrics are computed based on binary predictions of the network, which are obtained after applying a median filter and a threshold tuned on validation data (cf. [16]). Note that exact reproduction of previously reported errors, namely 8.0 in [18] (with a slightly different architecture) and 5.5 in [16], is difficult as Schlüter and Lehner train on a larger (in-house) dataset, whereas we only use publicly available data. Additionally, non-determinism (e.g., via data augmentation or initial weights) can influence the performance of a model noticeably.

In the following experiments, let the singing voice detection system be f , and $f(x)$ the numeric model output for class “sing”. The final classification is made by checking whether $f(x)$ is above (“sing”) or below (“no sing”) a threshold that is optimised on the validation set, and is equal to 0.51 for our system.

4.3 Hyperparameters for the Adversarial Attack

Equation (2) shows the hyperparameters we need to determine before attacking the singing voice detector. The maximum number of iterations in which we try to find adversarial perturbations that change the prediction of an excerpt is set to 1000 in our experiments. Other hyperparameters – clipping factor ϵ , update factor η and weight factor α – are tuned on the validation set of Jamendo, and chosen as the setting that results in the highest number of successful adversarial perturbations, i.e., changed the most predictions out of all audio excerpts in the validation data. For the attack on raw audio we use $\epsilon = 0.01$, $\eta = 0.0003$ and $\alpha = 2$; for the attack on the spectrograms we take $\epsilon = 0.1$, $\eta = 0.0005$ and $\alpha = 15$. Due to the binary nature of the singing voice detection task, the target t for each excerpt is chosen to be the opposite of its original prediction. Note that we find successful perturbations for 28.4% of all excerpts for attacks on the spectrogram, and for 60.5% of excerpts for raw audio.

4.4 Hyperparameters for the Explanations

As mentioned above the LIME algorithm has a set of hyperparameters that have to be chosen. We segment the spectrogram (80 frequency bins (F) \times 115 time frames (T)) into 5 time and 4 frequency segments, respectively, resulting in 20 “human-interpretable” features of size 20 frequency bins \times 23 time frames. Repeating the calculation of an explanation might lead to different results due to the randomness in the neighborhood generation when the number of generated instances is too small. A preliminary experiment as described in [5, 11] on a subset of the explanations, suggests that $2^{13} = 8192$ instances are sufficient to generate stable explanations, which is why we set $N_s = 8192$ for all our experiments. We also investigated different ways of replacing explanation segments [11] (*zero*, *min*, and *mean*), but as the results appeared similar, we will focus on reporting the results for the *mean* content type. For weighting the instances to train the explainer we use the cosine distance function and an exponential kernel with a kernel width of 0.25.

5. EVALUATION OF EXPLANATIONS

To investigate to what extent local explanation methods can find the underlying reason for a particular classification by a system, we perform four experiments. In the first, we demonstrate how LIME can help detecting rather obvious causes for a prediction that are due to manually altered spectrograms. The goal of the remaining experiments is to quantitatively evaluate the ability of LIME to detect adversarial perturbations: in the second experiment

we try to evaluate whether the segments that LIME highlights actually caused a prediction; in the third, we analyse if the explanations can detect the correct location of the adversarial perturbation if it only affects few segments of the input spectrogram; and in the fourth, we compare the fidelity scores for correct and incorrect explanations.

Due to the computation time needed for obtaining explanations, we subsequently limit the number of analysed excerpts for each song in the test set. For the first experiment, we manually alter a single random excerpt per song with an original classification of “no sing”. In the setting of the second, third and fourth experiment, for each of the 16 test songs, we randomly select 10 adversarial excerpts which were originally classified as “sing” and another 10 excerpts originally classified as “no sing”. We conduct all experiments with adversarial attacks computed both on raw audio and on spectrograms, as we want to account for the fact that attacks on raw audio might use implicit constraints that are difficult to detect for LIME (which operates on spectrograms).

To allow reproducibility, we publish code⁴ for all subsequent experiments.

5.1 Explaining Manually Altered Spectrograms

In our first experiment, we exploit the fact that the predecessor of the singing voice detector we use (cf. [18]) could easily be fooled by drawing on the input spectrogram⁵ [17]. After choosing a set of excerpts originally classified as “no sing”, we adapt drawings on the respective spectrograms until the prediction is changed to “sing”. Then we ask LIME to explain these new predictions. Figure 1 shows that the explainer can correctly identify the manually altered parts of the spectrogram as the cause for the wrong prediction. Examples like these, supported by relatively high fidelity scores, might make us think that LIME knows what is going on and make us trust these explanations. We could stop our evaluation here and put trust in our model because the explanations highlight seemingly correct behaviour [5], look meaningful [7], or match the expected behavior [9], but we will continue with a more careful investigation.

5.2 Explaining Predictions on Adversarial Examples

In this set of experiments, we first compute explanations for the predictions obtained for each of the selected adversarial excerpts. Every explanation consists of a list of interpretable features (time-frequency segments) and their corresponding weights. The weight of a particular feature should indicate the importance of the feature for making a certain prediction [4, 5]. To evaluate whether the explanation actually selects the features that are responsible for the (now wrong) prediction, we could naively check if the segments selected by LIME correspond to regions of the adversarial perturbation that have the highest magnitudes (defined via the L2 norm).

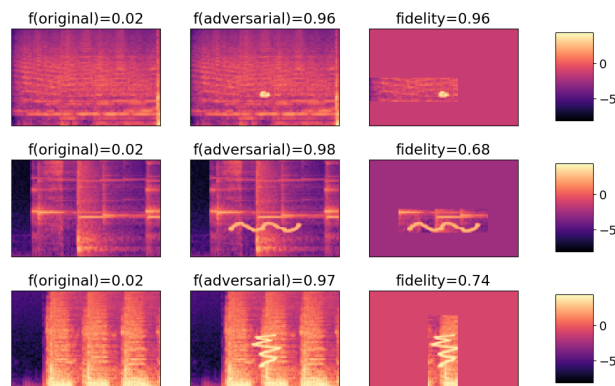


Figure 1: Different examples of spectrograms for which the model’s original prediction is “no sing” (left column) but drawing certain symbols (middle) leads to a change to “sing” with a model output f . The rightmost column shows the top 3 interpretable features (which may be directly adjacent and thus look like one segment) picked by the LIME algorithm, and their fidelity.

	Waveform	Spectrogram
“no sing” → “sing”	53 %	41 %
“sing” → “no sing”	21 %	32 %

Table 1: Percentage of label flips when using only $k = 3$ segments of an adversarial perturbation, based on the most relevant features chosen by LIME. Columns denote whether perturbations were computed for the waveform / spectrogram, rows show original → target prediction.

However, as we do not know which part of a perturbation really led to a misclassification, i.e., whether parts of a perturbation with the highest magnitude have the most influence or not, this could lead to false conclusions.

To circumvent this, we follow another approach: we split the adversarial perturbation into the same time-frequency segments that LIME is using as interpretable features; we then selectively add those k segments of the perturbation to the input spectrogram that coincide with the features identified as an explanation. If the selected features are actually explaining the prediction, we expect the partial perturbation to be sufficient to achieve the same change of classification, or what we call *label flip*, as the full perturbation did. However, using $k = 3$ (which seems a reasonable number of segments to present to a user [5]) flips only 21 – 53% of the predictions (see Table 1).

We therefore next look at whether larger numbers k of segments are able to flip more of the predictions. We use either LIME for selection of segments or choose the k perturbation segments with the highest magnitudes. We do this for $k \in [1, \dots, 20]$ and report the results in Figure 2. For almost all k and across different settings we are more often successful in flipping the label when segment selection is based on segment magnitude rather than on LIME results. For LIME-based segments it is necessary to choose almost all 20 available segments to achieve flip rates close to 100%. This result suggests that it is indeed possible to

⁴ <https://github.com/CPJKU/veracity>
⁵ https://github.com/f0k/singing_horse

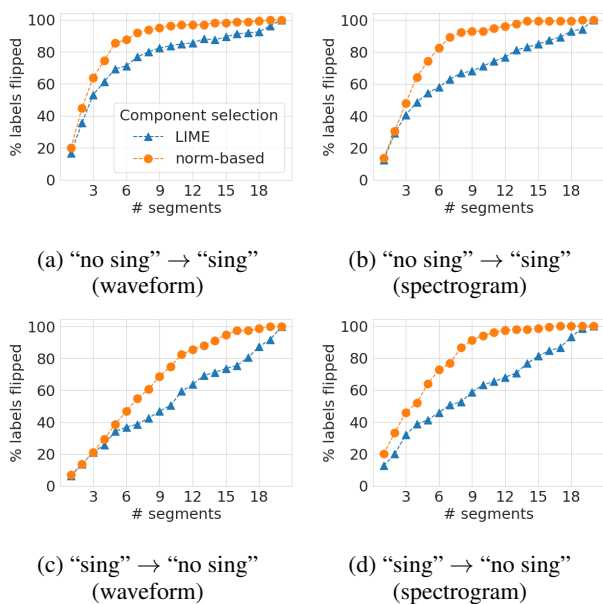


Figure 2: Percentage of classifications we can flip (y-axis) by adding only k segments (x-axis) of an adversarial perturbation to an input. Choosing which k segments are used is either based on LIME explanations (triangles), or on the norm of a perturbation (circles).

	Waveform		Spectrogram	
	LIME	norm	LIME	norm
“no sing” → “sing”	86 %	99 %	91 %	98 %
“sing” → “no sing”	49 %	74 %	44 %	55 %

Table 2: Percentage of label flips when adding all perturbation segments that contributed positively according to LIME, compared to using the same number of segments k selected based on the norm.

explain predictions with larger numbers of k interpretable features, but only when using the norm of a perturbation (which in a real application we would not know) to select the features and not when using LIME for the selection.

Finally, we analyse whether taking all segments that receive a positive weight from LIME can flip a prediction, as positive weights should correlate positively with the explained class [5]. For this analysis, we no longer add a fixed number of segments of a perturbation to each excerpt, but instead select all k segments of a perturbation for which the corresponding interpretable feature received a positive weight. This number of segments k can be different for each excerpt we look at. Additionally, we choose the same number of k segments based on the largest magnitude, and compare how often this results in label flips. The results of this experiment are summarised in Table 2. Again, we observe that taking partial perturbations is more successful when selecting the segments with the highest magnitude, as opposed to the segments most relevant for LIME.

These results suggest that LIME does not pick the input features that are most relevant for making a particular prediction.

5.3 Explaining “Localised” Perturbations

Considering the results in the previous section, one might argue that an adversarial perturbation can be present in the whole spectrogram, which could make it hard for LIME to decide which segments are most relevant. We make use of the finding that it is often sufficient to add only a small number of selected segments of the perturbation to flip a label, and conduct an additional experiment where we analyse how often LIME detects those segments.

This is similar to [15], where adversarial perturbations were constrained beforehand such that only selected regions of the input could be modified, but in contrast we do not restrict the perturbations directly. Instead we refine our adversarial perturbations by first splitting them into segments that correspond exactly to the time-frequency segments that LIME uses. We then use only k such segments of a perturbation, where the segments themselves are chosen based on the highest magnitude.

We then identify the subset of adversarial excerpts whose label can be flipped by only adding $k \in 1, 3, 5$ segments of the perturbation. These subsets are created for the four different settings, namely the two types of adversarial attacks (waveform / spectrogram), and for the two possible label flips (“sing” to “no sing” and vice versa). This leads to $3 * 2 * 2 = 12$ subsets of adversarial excerpts.

After determining this subset of adversarial excerpts with partial (“localised”) perturbations, we once again compute explanations with LIME. Here we set the number of interpretable features that LIME should display to k , i.e. the number of segments we previously added as localised perturbations. Since the time-frequency segments of LIME and the partial perturbations align, we can then examine how often LIME correctly identifies the regions causing a particular prediction. Figure 3 shows the result of this experiment, for all 12 adversarial subsets. Each plot depicts how many of the k segments that were added as adversarial perturbation, and that hence were crucial for a particular classification, are correctly detected by LIME. It is easiest to interpret results with only one modified segment (left column), since we know exactly what the correct explanation should look like. Overall, for less than half of the excerpts is the correct segment presented as an explanation. For 3 and 5 modified segments we can also check how many segments are correctly identified, and we can observe that it is rarely all of them and for some settings none of them. This result suggests that even when the cause is quite localised and aligns with the segments that we are using as interpretable features, LIME is rarely able to recover all affected features.

5.4 Are Fidelity Scores Reliable?

As Figure 1 in Section 5.1 suggested previously, “correct” explanations are often accompanied by relatively high fidelity values. We do not know, however, whether we can use fidelity to determine if an explanation highlights the segments of the input that are really the most relevant for a prediction, and it has been demonstrated that a high-fidelity explanation of a black box model might not reflect

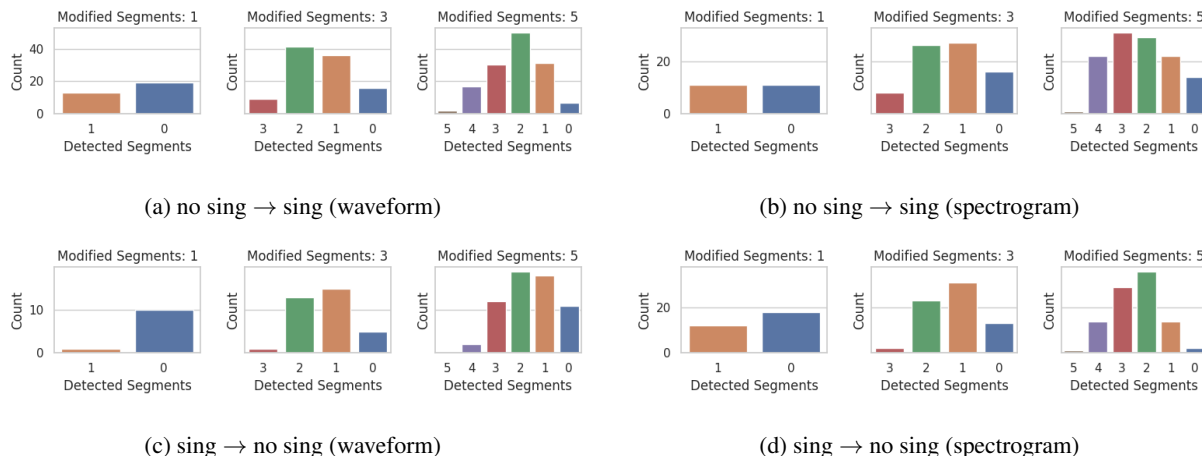


Figure 3: The number of segments that are correctly identified when adding k perturbed segments to the original input and asking LIME for an explanation containing the same k number of interpretable features.

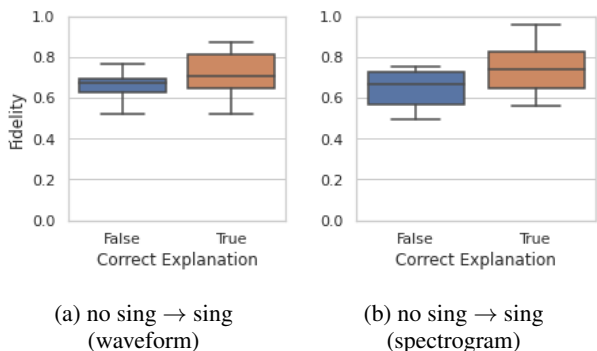


Figure 4: Fidelity scores for explanations that correctly and incorrectly identified the most relevant segment.

the actual cause for a prediction [22].

In the last experiment described above we have created a setting with $k = 1$ where we exactly know which segment is relevant for the prediction, and we can use this information to compare whether the fidelity is different for “correct” and “wrong” explanations, i.e. explanations which match the added segments of the perturbation as opposed to explanations which do not. As can be seen from Figure 4, the fidelity for “correct” explanations is on average only slightly higher than for “wrong” predictions, with ranges of fidelity values strongly overlapping. Based on the fidelity score alone, a user hence cannot decide whether an explanation is “correct” and whether it shows the most relevant segment(s) for a prediction.

6. DISCUSSION AND CONCLUSION

In a nutshell, the central results of our investigations can be summarised as follows:

- Figure 1: “Obvious” causes for predictions can be detected with high fidelity.
- Table 1: When computing the 3 most relevant interpretable features with LIME, we detect segments

that are able to flip the label for only 21-53% of the excerpts.

- Figure 2: When using a varying number of interpretable features, we detect fewer “correct” segments than when simply taking the same number of segments with the highest magnitude.
- Table 2: When using all interpretable features that received a positive weight, we detect fewer “correct” segments than when taking the same number of segments with the highest magnitude.
- Figure 3: In a setting where we only add partial perturbations, only few segments are correctly detected.
- Figure 4: Based on the fidelity score it is impossible to judge the quality of an explanation.

Taken together, we believe that these results support the following conclusions: (1) local model-agnostic explanations cannot reliably detect the input regions most relevant for a prediction unless they are rather obvious; (2) evaluation based on “what looks reasonable” leads to accepting explanations that do not reveal the real cause of a prediction; (3) the fidelity score may give a false sense of security about the quality of explanations.

In [15], Göpfert et al. write, “It might be tempting to judge explanatory methods on whether they succeed in identifying features that a human observer thinks should be relevant to the classification, but the existence of adversarial examples shows that the reasoning of humans and neural networks can differ dramatically.” This observation points to a fundamental dilemma between “veridical” explanations that faithfully reflect the workings of the classification model (and may not be accessible to model-agnostic explanation methods such as LIME), and “human-interpretable” explanations that would connect a machine decision to concepts familiar to us. Our results confirm that one should be careful in interpreting model-agnostic explanations as explanations of the underlying model, and that experiments with carefully crafted examples are important to get more detailed insights into the properties of such methods.

7. ACKNOWLEDGEMENTS

This work is supported by the Austrian National Science Fund (FWF, project P31988).

8. REFERENCES

- [1] S. Mishra, B. L. Sturm, and S. Dixon, “Understanding a Deep Machine Listening Model Through Feature Inversion,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, 2018, pp. 755–762.
- [2] —, ““What are You Listening to?” Explaining Predictions of Deep Machine Listening Systems,” in *Proceedings of the 26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018*. IEEE, 2018, pp. 2260–2264.
- [3] S. Mishra, D. Stoller, E. Benetos, B. L. Sturm, and S. Dixon, “GAN-based Generation and Automatic Selection of Explanations for Neural Networks,” *CoRR*, vol. abs/1904.09533, 2019.
- [4] M. T. Ribeiro, S. Singh, and C. Guestrin, ““Why Should I Trust You?”: Explaining the Predictions of Any Classifier,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 2016, pp. 1135–1144.
- [5] S. Mishra, B. L. Sturm, and S. Dixon, “Local Interpretable Model-agnostic Explanations for Music Content Analysis,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, 2017, pp. 537–543.
- [6] B. Chettri, B. L. Sturm, and E. Benetos, “Analysing Replay Spoofing Countermeasure Performance under Varied Conditions,” in *28th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2018, Aalborg, Denmark, September 17-20, 2018*. IEEE, 2018, pp. 1–6.
- [7] V. Haunschmid, S. Chowdhury, and G. Widmer, “Two-level Explanations in Music Emotion Recognition,” *Machine Learning for Music Discovery Workshop, MLAMD at ICML2019*, 2019.
- [8] V. Haunschmid, E. Manilow, and G. Widmer, “Towards Musically Meaningful Explanations Using Source Separation,” *CoRR*, vol. abs/2009.02051, 2020.
- [9] —, “audioLIME: Listenable Explanations Using Source Separation,” in *13th International Workshop on Machine Learning and Music*, 2020, pp. 20–24.
- [10] A. B. Melchiorre, V. Haunschmid, M. Schedl, and G. Widmer, “LEMONS: Listenable Explanations for Music recOmmeNder Systems,” in *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 12657. Springer, 2021, pp. 531–536.
- [11] S. Mishra, E. Benetos, B. L. Sturm, and S. Dixon, “Reliable Local Explanations for Machine Listening,” in *2020 International Joint Conference on Neural Networks, IJCNN 2020, Glasgow, United Kingdom, July 19-24, 2020*. IEEE, 2020, pp. 1–8.
- [12] Z. C. Lipton, “The Mythos of Model Interpretability,” *Communications of the ACM*, vol. 61, no. 10, pp. 36–43, 2018.
- [13] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim, “Sanity Checks for Saliency Maps,” in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 2018, pp. 9525–9536.
- [14] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, “Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods,” in *AIES ’20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*. ACM, 2020, pp. 180–186.
- [15] J. P. Göpfert, H. Wersing, and B. Hammer, “Recovering Localized Adversarial Attacks,” in *Artificial Neural Networks and Machine Learning - ICANN 2019: Theoretical Neural Computation - 28th International Conference on Artificial Neural Networks, Munich, Germany, September 17-19, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 11727. Springer, 2019, pp. 302–311.
- [16] J. Schlüter and B. Lehner, “Zero-Mean Convolutions for Level-Invariant Singing Voice Detection,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, 2018, pp. 321–326.
- [17] J. Schlüter, “Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals,” Ph.D. dissertation, Johannes Kepler University Linz, Austria, Jul. 2017.
- [18] J. Schlüter and T. Grill, “Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, 2015, pp. 121–126.
- [19] N. Carlini and D. A. Wagner, “Audio Adversarial Examples: Targeted Attacks on Speech-to-Text,” in *Proceedings of the IEEE Security and Privacy Workshops, SP Workshops*. IEEE, 2018, pp. 1–7.

- [20] K. Prinz, A. Flexer, and G. Widmer, “On End-to-End White-Box Adversarial Attacks in Music Information Retrieval,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 93–104, 2021.
- [21] M. Ramona, G. Richard, and B. David, “Vocal Detection in Music with Support Vector Machines,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2008, Las Vegas, Nevada, USA*. IEEE, 2008, pp. 1885–1888.
- [22] H. Lakkaraju and O. Bastani, ““How do I fool you?”: Manipulating User Trust via Misleading Black Box Explanations,” in *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*. ACM, 2020, pp. 79–85.

IS THERE A "LANGUAGE OF MUSIC-VIDEO CLIPS" ? A QUALITATIVE AND QUANTITATIVE STUDY

Laure Prétet
LTCI, Télécom Paris
Bridge.audio, Paris, France

Gaël Richard
LTCI, Télécom Paris
IP Paris, France

Geoffroy Peeters
LTCI, Télécom Paris
IP Paris, France

ABSTRACT

Recommending automatically a video given a music or a music given a video has become an important asset for the audiovisual industry - with user-generated or professional content. While both music and video have specific temporal organizations, most current works do not consider those and only focus on globally recommending a media. As a first step toward the improvement of these recommendation systems, we study in this paper the relationship between music and video temporal organization. We do this for the case of official music videos, with a quantitative and a qualitative approach. Our assumption is that the movement in the music are correlated to the ones in the video. To validate this, we first interview a set of internationally recognized music video experts. We then perform a large-scale analysis of official music-video clips (which we manually annotated into video genres) using MIR description tools (downbeats and functional segments estimation) and Computer Vision tools (shot detection). Our study confirms that a "language of music-video clips" exists; i.e. editors favor the co-occurrence of music and video events using strategies such as anticipation. It also highlights that the amount of co-occurrence depends on the music and video genres.

1. INTRODUCTION

Each day, an ever-growing quantity of videos is created by professionals (for advertisement, movies, series, etc) and individuals (for Instagram, TikTok, YouTube, etc). Finding an appropriate soundtrack to emphasize the video content is therefore a common exercise, which can be time-consuming if done manually. This explains the success of commercial systems such as MatchTune or of research papers such as "Look, Listen and Learn" [1]. While such systems are very good at recommending music based on the video content, the temporal synchronization between both modalities is rarely taken into account. In order to develop synchronization-aware recommendation systems, some domain knowledge is required on how the synchronization is performed in real videos that feature music. In

this work, we attempt at bridging this knowledge gap by performing a fine-grained cross-modal analysis of the synchronization between audio and video content. We hypothesize that better understanding professionally produced music videos helps designing better models for music-video synchronization. This has applications in automatic music-video recommendation [2–6] and generation [7–9].

Temporal structure (at the beat, bar or functional segment level) is one of the dominant characteristics of music. For this reason, its automatic estimation has received a lot of attention in the Music Information Retrieval (MIR) community [10]. Temporal structure in video (cuts, scenes, chapters) has similarly received a lot of attention in the Computer Vision community (for example with the goal of creating video summary [11]). Our fine-grained analysis will be using these structural elements.

Our cross-modal analysis could be performed on any type of video that features a musical soundtrack (eg commercials, movies). We focus here on the special case of Official Music Videos (OMV). We call OMV an audiovisual document where the audio part consists in a music track, and which aims at promoting said track and its performing artists. As a result, the music track is generally the only source of audio in OMVs. This makes OMVs good prototypes for a study on music-video synchronisation. We do not consider user-generated videos, because we assume that analyzing professionally produced OMVs is more likely to provide reusable insights.

In the specific case of OMVs, the editing team will often arrange the video rushes based on the structure of the music track [12]. In some cases, the music track can also be adapted from the studio version for narrative purposes. Therefore, music and video structure are de facto associated. However, the level of synchronicity is not always the same, depending on the considered OMV. This is not only due to artistic choices but also depends on the music genre and video genre, as we will see in our study.

Proposal and paper organization. In this paper, we study the relationship between music and video temporal organization using a qualitative and a quantitative approach. The *qualitative* study is based on a set of interviews with three renowned specialists of official music videos. We interview them in order to find out if and how they consider the relationship between music and video structure in their work. The *quantitative* analysis is based on a detailed analysis of music and video structural events in OMV using MIR and Computer Vision



© Laure Prétet, Gaël Richard, Geoffroy Peeters. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Laure Prétet, Gaël Richard, Geoffroy Peeters, "Is there a "language of music-video clips" ? A qualitative and quantitative study", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

tools. The OMVs correspond to a subset of the Harmonix dataset [13]. We study specifically the relationship between the duration of music and video segments and between the positions of their respective boundaries. We highlight the dependency of those according to the OMV music and video genre (for which we annotated the data).

The paper is organized as follows. Section 2 discusses the related literature. Section 3 describes the qualitative study and summarizes the interviews of three music video experts: Jack Bartman (composer), Alexandre Courtès (director) and Maxime Pozzi (editor). Section 4 describes the quantitative study: the dataset creation (4.2), the analysis of the music and video segment duration (4.3.1) and of the music and video segment position (4.3.2). Section 5 concludes and discusses the perspectives of this work.

2. RELATED WORK

2.1 Music-Video Synchronization: A Short Review

Music supervision is an industry that aims specifically at synchronizing music to video. Music supervision experts are dedicated to proposing the best soundtrack to all types of videos, ranging from commercials to movies and series. As of today, this recommendation and synchronization work still features a large amount of manual work. Inskip et. al. [14] interviewed music supervision experts and described their workflow. The authors mention that "the clearest briefs appear to be moving images", suggesting that other types of data (emotion, textual description, reference soundtracks) are not necessary to perform the task.

At the same period, Gillet et al. [15] proposed a system that can automate part of the music supervision task. Their system relies on the synchronization of music structure (onsets and functional segments) and video structure (motion intensity and cuts) to perform music-video recommendation, without external data. Yang [16] and Mulhem [17] proposed similar approaches.

More recently, Alexander Schindler gathered a dataset of OMVs (the Music Video Dataset) and performed an in-depth analysis of this specific media [18]. In [19], Schindler and Rauber explain how shot boundary detection is an ill-defined task in music videos, as shot transitions are used in a complex and artistic way. By analyzing the clips, they observe that the music videos present characteristic editing styles (number of shots per second, types of transition) for certain music genres or moods. But they do not quantify this correlation. In [12], the same authors analyze the correlation between visual contents (objects present in the scene) and music genre. For example, cowboy hats are almost systematic in country music videos.

In our study, we propose a joint approach. We analyze the correlation between the music/video *structure* and music/video *genres*.

2.2 Audiovisual Structure Estimation Tools

Our quantitative study (Section 4) relies both on MIR to analyze the music structure and on Computer Vision to

analyze the video structure. More specifically, we estimate the downbeat positions, functional segments and shot boundaries from the OMV of our dataset. In the following, we describe the tools we have used for our analysis.

Downbeat tracking is a popular MIR task [20]. As a result, several ready-to-use libraries are available to estimate downbeat positions from audio files [21, 22]. The state-of-the-art algorithm of Böck et al. [23] consists in two steps. First, a `RNNDownBeatProcessor`, which relies on multiple Recurrent Neural Networks (LSTMs), estimates jointly beat and downbeat activation functions. The output of the neural networks represents the probability of each frame of being a beat or downbeat position. These activation functions are then fed as observations to a `DBNDownBeatTrackingProcessor`, which relies on a Dynamic Bayesian network (DBN). The DBN outputs the beat positions of highest likelihood, along with their position inside the bar.

At a larger timescale, the automatic detection of boundaries between functional segments (choruses, verses and bridges) has also received a lot of attention from the MIR community. The Ordinal Linear Discriminant Analysis (OLDA) algorithm by McFee et al. [24] relies on supervised learning to perform this task. This method adapts the linear discriminant analysis projection by only attempting to separate adjacent segments. Then, the obtained features are clustered with a temporal constraint: only similar successive segments are merged together.

Similar to music, videos can be divided into segments of various duration, from shots to scenes to chapters and longer sequences. In this study, we focus on a segmentation into shots. The TransNet system [25], by Souček et al., is a Convolutional Neural Network which employs dilated 3D convolutions and which is trained in a supervised way on a shot boundary detection task.

3. QUALITATIVE ANALYSIS: INTERVIEWS

3.1 Methodology

In order to gather intuition on the synchronization of music and video, we conducted a series of semi-structured face-to-face interviews. We selected three music video experts from different professions: composition, direction and editing. Following Inskip et. al. [14], we selected the respondents using a snowball sampling technique.

Interviews were performed using the Zoom video conferencing software, lasting up to one and a half hours. The interviews were transcribed manually by the researcher, and transcripts were sent back to the respondents for validation. Areas of discussion included the participant's day-to-day workflow and technical tools, their interactions with the other professions of the industry, and their opinion on example music videos prepared by the researcher.

3.2 Interviews Summary

3.2.1 Jack Bartman, Composer

As a composer (for commercials such as Nike, Apple or UbiSoft), Bartman has to adopt both a global and a pre-

cise local approach: the content of the music has to match the visual atmosphere, and its temporal structure must be aligned both globally at clip level and locally at frame level. In some cases, the editing follows the structure of the music. But in other cases, typically for advertisement, it is the opposite, and the composer has to adapt the music to an existing movie. Most of the time, when music has to be edited on an existing movie, the slicing operation is privileged.

"Slicing can happen at unconventional moments, like the first or last beat of a bar! I simply add a sound effect to make it work."

Time stretching and accelerations can be employed too, but are far less usual. Bartman stresses that synchronizing cuts to audio events is especially important around emotional climaxes of the video. Finally, for some projects, an exact synchronization is not the golden rule:

"This year, I worked on a short movie about psychological aspects of the Covid-19 lockdown. After getting used to an imperfectly synchronized mockup soundtrack, the director did not want to use the final version, as the mockup would better suit the intended "madness" atmosphere."

3.2.2 Alexandre Courtès, Director

As a director (such as for U2, Phoenix, Cassius, Franz Ferdinand or Jamiroquai), Courtès generally has a lot of freedom when it comes to the temporal organization of a music video. Directors often come up with their own concept and they have little constraint about the content of the video. At large temporal scale, their mission is to emphasize the music climaxes by the appropriate video content.

"The music video will often show a performance, so it is similar to a musical comedy: it has to feature costumes, chapters, sets, acts."

Directors are not responsible for placing the cuts, but they can introduce diversity in the video transitions (explosions, large objects passing in front of the camera; see [19] for a more exhaustive list).

"Cuts have to follow the music's rhythm, even though they might not always co-occur with beats."

3.2.3 Maxime Pozzi, Editor

As an editor (such as for Rihanna, Taylor Swift, Foals or Woodkid), Pozzi has to combine both a local, frame-level approach to the design of a global emotional trajectory.

"Editors and musicians have a similar job, we all want the same thing: rhythm, narration, climaxes."

For chorus and verses, the editing will follow the rhythm and typically accelerate near climaxes. During bridges, it will often be slower and poetic. This can be illustrated for example by Katy Perry's *Firework* music video (Figure 1). In this clip, we can see some functional segments where cuts happen very frequently (several times in each bar) and segments where they happen less frequently, for example on the downbeats only.

Editing can be used as an element of narration. For example, in Adele's *Rolling in the deep* music video, starting at timestamp 02:20, the cuts are systematically placed just before the downbeat (see Figure 2).

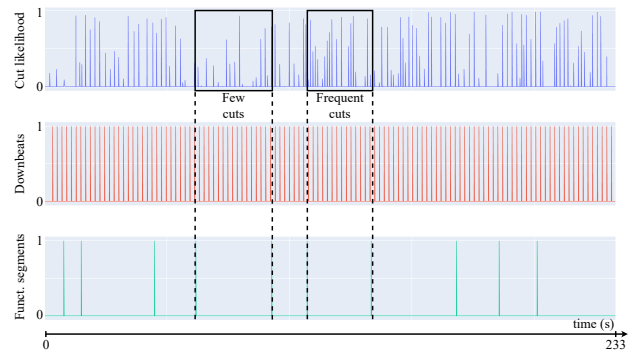


Figure 1: Audiovisual structure of Katy Perry, *Firework*, full clip. Horizontal axis: time. Cuts: TransNet estimates. Downbeats: Madmom estimates. Music functional segments: OLDA estimates.

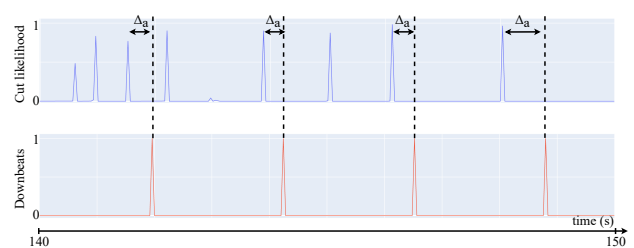


Figure 2: Audiovisual structure of Adele, *Rolling in the deep*, timestamps 02:20 to 02:30. Horizontal axis: time. Cuts: TransNet estimates. Downbeats: Madmom estimates. Δ_a : anticipation of cuts with respect to downbeat.

"Off-beat cuts are used to create dynamics: to surprise the viewer, and illustrate the music's emotional climax. It makes the video direction appear more "indie" as well, this can be required by the performing artists."

3.3 Summary

These three interviews provide us with a series of intuitions and hypotheses about the way audio and video are synchronized in music videos. First, musical structure such as chorus and verses are taken into account when directing a music video. Second, audio events such as rhythm, beat and downbeat are taken into account when editing a music video. Finally, according to the desired atmosphere, the audio and video structural events can be more or less perfectly synchronized.

4. QUANTITATIVE ANALYSIS

4.1 Methodology

In the following, we conduct a set of quantitative experiments on how the Structural Events (SE) of the music and of the video are synchronized in time. We do so using Official Music Videos (OMVs). We therefore first collect a dataset of OMVs, along with music and video genre annotations (Section 4.2). For each of them we use MIR tools to estimate music SE (downbeats and functional segments) and Computer Vision tools to estimate video SE (shot boundaries). In our first experiment, we study the

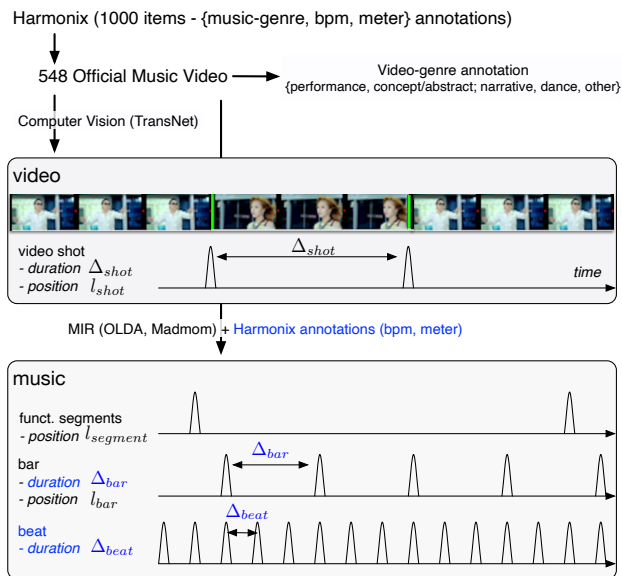


Figure 3: Schematic view of the different audiovisual structural events considered: shots (Δ_{shot}, l_{shot}), functional music segments ($l_{segment}$), bars/downbeats (Δ_{bar}, l_{bar}) and beats (Δ_{beat}, l_{beat}). Illustration music video: Psy, *Gangnam Style*.

correlation between the duration of the shots and the various musical SEs (beat and bar duration). In our second experiment, we study the temporal co-occurrence of the shot boundaries and the various musical SEs (bar and functional segment boundaries). We analyze the results of those for each music genre and each video genre.

4.2 Dataset

For our quantitative study, we consider a subset of the Harmonix dataset [13]. Harmonix was initially released for automatic estimation of beat, downbeat and functional music segments. It features popular (mostly hits) Western music tracks for which there is a high probability of having an associated music video. From the list of 1,000 YouTube video links provided, 899 were successfully retrieved, of which 40% contained only still images and 2.4% were duplicates. As a contribution of this work we provide the list and URLs of the remaining 548 OMVs as well as the genre annotations described below¹.

4.2.1 Annotations into Structural Events

We consider here two types of Structural Events (SE): those based on the music content -audio-, and those based on video content -image frames over time (see Figure 3).

Music SE. We consider three types of music SEs. At the smallest temporal scale we consider the beats and downbeats; at the largest temporal scale we consider the functional music segment boundaries (between the verses, bridges, choruses). Harmonix features a set of manual annotations². However, these annotations correspond to studio versions of the tracks which can, in some cases,

be largely different from the version used in the OMV. For this reason, we only used the annotations into bpm and meter of the Harmonix dataset to get the beat duration $\Delta_{beat} = \frac{60}{bpm}$ and bar duration $\Delta_{bar} = 4$ or $3\Delta_{beat}$ (which is computed as a multiple of the bar duration using the time signature). For the downbeat positions, we used the algorithm of Böck et al. [23], implemented in the Madmom library [21]. In the following, we denote by l_{bar} the list of downbeat positions for a given track. For the functional music segments, we used the implementation of OLDA from the MSAF library [26]. In the following, we denote by $l_{segment}$ the list of boundary positions between the segments for a given track. For our dataset, the average duration of functional music segments is 19.73 s. and the average bar duration is 2.30 s.

Video SE. We consider only the least ambiguous video SE, the shot boundaries (or cuts). To detect boundaries between shots, we use the TransNet system [25] and the associated library, available on GitHub³. The TransNet output is a continuous function of time $f_{shot}(t) \in [0, 1]$ representing the likelihood of a boundary at time t . f_{shot} has a sampling rate of 25 Hz.

Also, for each OMV, we compute the histogram of its shot duration. We do so by first estimating the list of shot boundary positions l_{shot} by thresholding $f_{shot}(t)$ with $\tau = 0.5$. The resulting segments have an average duration Δ_{shot} of 4.76s. We then compute the histogram of these durations. We denote by Δ_{shot}^{max} the position of the maximum of this histogram (in seconds).

We sum up the various SE in Table 1.

Table 1: Notation associated to each SE considered.

Music		
genre		Harmonix annotations
funct. segments positions	$l_{segment}$	OLDA/MSAF
bar duration	Δ_{bar}	Harmonix annotations
bar/downbeat positions	l_{bar}	Madmom
beat duration	Δ_{beat}	Harmonix annotations
Video		
genre		Manual annotations
shot boundary probability	$f_{shot}(t)$	TransNet
shot boundary positions	l_{shot}	
most common shot duration	Δ_{shot}^{max}	

4.2.2 Annotations into genre

We consider both the genre associated to the music and the one associated to the video.

Music genre. While still controversial in its exact definition [27], music genre is a convenient way to describe musical content. For this reason, it has been and it is still a widely studied topic⁴. For our experiments, we use the music genre annotations provided by the Harmonix dataset metadata.

Video genre. Video genre classification is a much less studied topic. Existing studies focus on a much smaller sets of video genres [32–34]. Only Gillet et al. [2] and

¹ Our list is accessible at: https://gitlab.com/creaminall/publications/ismir-2021-language-of-clips/-/blob/master/video_genres.csv.

² into functional segments, downbeat and beat.

³ <https://github.com/soCzech/TransNet>

⁴ It has dedicated challenges [28], and large datasets featuring hundreds of categories [29–31].

Schindler [18] studied the case of OMVs and there is no consensus on their taxonomy of video genres. There is also no annotated dataset for this task.

We merge [2] and [18] to obtain a set of 5 video categories and a corresponding single-label dataset. Maxime Pozzi, a professional music video editor, validated our taxonomy during our preliminary interview (see part 3.2.3). One author then manually annotated all 548 video clips of Harmonix into the five following video genres:

- **Performance videos (P)**: The artist or band are presented performing the song. 74 videos; example: Iron Maiden, *Powerslave*.
- **Concept/Abstract videos (C)**: The video illustrates the music metaphorically via a series of abstract shots related to semantics or atmosphere of the song. 227 videos; example: Lady Gaga, *Poker Face*.
- **Narrative videos (N)**: The music video has a strong narrative content, with identifiable characters and an explicit chronology. 160 videos; example: Taylor Swift, *Teardrops on My Guitar*.
- **Dance videos (D)**: Artists present a rehearsed dance choreography in sync with the music. 62 videos; examples: Sean Paul, *Get Busy*.
- **Other (O)**: Other types of music videos, including lyrics videos, animated music videos, etc. 25 videos; example: Train, *Hey*, *Soul Sister*.

4.3 Experiments

We hypothesize that the music structural events play an important role for the placement of cuts during the video editing. We check this assumption by measuring:

- if their segment duration are correlated in Section 4.3.1;
- if their position co-occur in Section 4.3.2.

According to Gillet [2], the performance of alignment-based music-video recommendation systems are strongly correlated to the video genre. We therefore differentiate our results by music and video genre.

4.3.1 Comparison between events duration

Our first experiment aims at evaluating to which extent the musical and video events have similar durations.

To measure this, we compare Δ_{shot}^{\max} (the most common shot duration) with the beat duration Δ_{beat} and bar duration Δ_{bar} obtained from the Harmonix annotations. When Δ_{shot}^{\max} is close to Δ_{bar} , this indicates that a systematic change of shots occurs with the same speed as the bar changes. This however does not mean that the changes occur simultaneously (we study this in Section 4.3.2).

This is for example the case of "Heartless" by Kanye West (see Figure 4 [top]) where the large peak at $\Delta_{shot}^{\max}=2.72$ s can be explained by the tempo at 88 bpm; or "Firework" by Katy Perry (see Figure 4 [bottom]) where the large peak at $\Delta_{shot}^{\max}=1.93$ s can be explained by the tempo at 124 bpm.

In our dataset, a synchronization at the bar level ($0.5\Delta_{bar} < \Delta_{shot}^{\max} < 1.5\Delta_{bar}$) occurs for one fifth of the clips (95 music videos). Synchronization may also occur at other levels: at the beat level Δ_{beat} , or the pattern

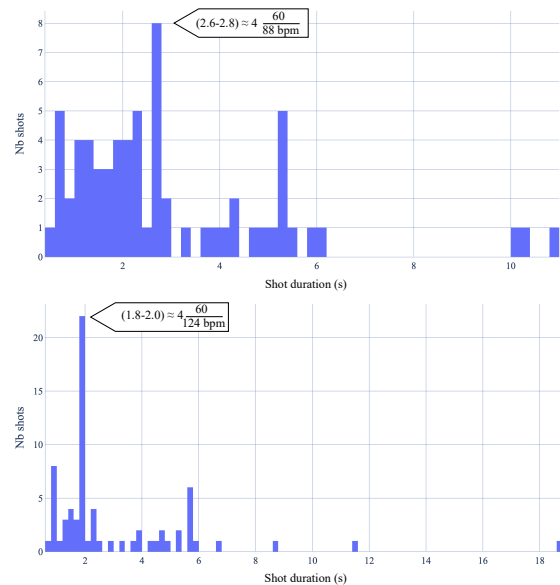


Figure 4: [top] Histogram of shot duration in the music video of *Heartless* by Kanye West. The tempo is 88 bpm. [bottom] Histogram of shot duration in the music video of *Firework* by Katy Perry. The tempo is 124 bpm.

level $\Delta_{pattern}$ (usually an even multiple of the bar duration). In our dataset, a synchronization at the beat level ($0.5\Delta_{beat} < \Delta_{shot}^{\max} < 1.5\Delta_{beat}$) occurs for two thirds of the clips (329 music videos). However, synchronization at pattern level $\Delta_{pattern} = 4\Delta_{bar}$ almost never occurs (2 music videos).

In Table 2, we indicate for each music genre and video genre, the number of tracks for which the Δ_{shot}^{\max} correspond to Δ_{bar} or Δ_{beat} . We only focus here on the most represented genres, i.e. which appear at least 10 times. We observe a strong correspondence between Δ_{shot}^{\max} and Δ_{bar} for the music genres Country, Dance/Electro and Rock (one fourth of the tracks). We observe a strong correspondence between Δ_{shot}^{\max} and Δ_{beat} for the music genres Alternative and Reggaeton (three quarters of the tracks). This may imply, for example, that music video professionals favor more dynamic editing styles (using shorter shots on average) for Reggaeton than for Country music. We observe a strong correspondence between Δ_{shot}^{\max} and Δ_{bar} for the video genre Performance (one fourth of the tracks). On the contrary, we observe a low correspondence between Δ_{shot}^{\max} and Δ_{beat} for the video genre Other (one third of the tracks). It is likely that music videos in the Other category favor experimental editing styles, with shots of more diverse duration.

As we see, there is a strong relationship between the video events and musical events duration. This however does not mean that the changes occur simultaneously. We study this in the next section.

4.3.2 Comparison between events position

Our second experiment aims at evaluating to what extent the musical events l_{seg} , l_{bar} and video events $f_{shot}(t)$ happen simultaneously. To measure this, we compute for each audio boundary i ($t_i \in l_{seg}$ or $t_i \in l_{bar}$) a score $S_i \in [0, 1]$.

Table 2: Agreement of musical structure (bar Δ_{bar} and beat Δ_{beat} level) and dominant shot duration Δ_{shot}^{max} according to the **music genre** [top table] and according to the **video genre** [bottom table]. Highest values are highlighted in bold, lowest values in italic.

Music Genre	$\Delta_{shot}^{max} \simeq \Delta_{bar}$		$\Delta_{shot}^{max} \simeq \Delta_{beat}$	
	# tracks	%	# tracks	%
Alternative	2	8.3	19	79.2
Country	10	29.4	16	47.1
Dance/Electro	12	24.5	28	57.1
Hip-Hop	12	12.6	69	72.6
Pop	40	14.8	158	58.3
R&B	1	5.3	13	68.4
Reggaeton	1	8.3	9	75.0
Rock	4	23.5	10	58.8

Video Genre	$\Delta_{shot}^{max} \simeq \Delta_{bar}$		$\Delta_{shot}^{max} \simeq \Delta_{beat}$	
	# tracks	%	# tracks	%
Concept	33	14.5	148	65.2
Dance	11	17.7	40	64.5
Narration	28	17.5	93	58.1
Performance	19	25.7	41	55.4
Other	4	16.0	8	32.0

S_i is defined as the integral over time of the shot boundary likelihood $f_{shot}(t)$ tapered by a non-normalized Gaussian window $w(t)$. $w(t)$ is centered on t_i , with $\sigma = 2$ (such that the effective duration of the window is approximately 0.5s at a frame rate of 25Hz) and with $w(0) = 1$.

$$S_i = \int_t w(t - t_i) f_{shot}(t) dt, \quad \forall t_i \in \{l_{seg}, l_{bar}\}$$

A large value of S_i indicates that the t_i position (the music structural event) corresponds to a large probability of shot boundary. We then average S_i for all audio boundaries i to get S . S might be considered as a measure of precision, since it provides information on *how many audio boundaries are explained by a video boundary*. It should be noted that the number of video boundaries is larger than the number of audio boundaries (as seen in Figures 1 and 2). S is also close to the measure proposed by [35] to evaluate the performances of beat-tracking algorithms. A large value of S indicates that the shot boundaries are located at the same positions as the music structural events l_{seg} or l_{bar} . We compute S separately using the t_i from l_{seg} or from l_{bar} . To check if the amount of music-video event synchronization depends on the music and video genre, we average S over all tracks of a given genre (music or video).

Co-occurrence of music/video events by music genre.

Table 3 [top part] shows the co-occurrence scores S aggregated over music genres. We observe variations of the values of S according to the music genre. For **Pop**, $S(l_{seg})$ is large (0.36) indicating that many shot transitions occur at the functional segment boundaries positions. For **R&B** and **Reggaeton**, $S(l_{bar})$ is large (0.31 and 0.28) indicating that many shot transitions occur at the downbeat positions. We also observe that the value of $S(l_{seg})$ and $S(l_{bar})$ vary according to the music genre with very small values for **Dance/Electronic**, **Hip-Hop** and **Rock**. This comes as a surprise especially for **Dance/Electronic**, because in the previous experiment, we observed a strong correspondence between the duration of shots and bars for this music genre. This shows that even though bars and

Table 3: Shot transition intensity S around music boundaries (either functional segments boundaries l_{seg} or bar boundaries l_{bar}) according to **music genre** [top table] and according to the **video genre** [bottom table]. Mean values and confidence intervals at 95% are displayed. Highest values are highlighted in bold, lowest values in italic.

Music Genre	$S(l_{seg})$	$S(l_{bar})$	# tracks
Alternative	0.22 ± 0.08	0.23 ± 0.02	24
Country	0.20 ± 0.06	0.21 ± 0.02	34
Dance/Electro	<i>0.18 ± 0.05</i>	0.21 ± 0.02	49
Hip-Hop	0.19 ± 0.03	0.25 ± 0.01	95
Pop	0.36 ± 0.02	0.21 ± 0.01	271
R&B	0.29 ± 0.10	0.31 ± 0.03	19
Reggaeton	0.24 ± 0.11	0.28 ± 0.04	12
Rock	<i>0.18 ± 0.07</i>	<i>0.19 ± 0.03</i>	17

Video Genre	$S(l_{seg})$	$S(l_{bar})$	# tracks
Concept	0.20 ± 0.02	0.23 ± 0.01	227
Dance	0.18 ± 0.04	0.24 ± 0.01	62
Narration	0.18 ± 0.03	0.23 ± 0.01	160
Performance	<i>0.15 ± 0.04</i>	0.16 ± 0.01	74
Other	<i>0.11 ± 0.06</i>	<i>0.11 ± 0.02</i>	25

shots have similar duration, their boundaries might not always co-occur.

Co-occurrence of music/video events by video genre.

Table 3 [bottom part] shows the co-occurrence scores S aggregated over video genres. We observe variations of the values of S according to the video genre. We see that the **Dance** video genre has a large value of $S(l_{bar})$ (0.24), which is not surprising given that video labeled as **Dance** actually show people dancing on the beat. We also observe large values of $S(l_{bar})$ for the **Concept** and **Narration** video genres with consistent synchronization on the downbeats. For the **Performance** video genre (the band is playing in front of the camera), we don't observe such a large correspondence ($S(l_{bar}) = 0.16$). For the **Other** video genre, the low values ($S(l_{bar}) = S(l_{seg}) = 0.11$) are not surprising, given that some videos are very experimental and may feature complex video transitions, which may be difficult to detect by the TransNet.

5. CONCLUSION

According to the professionals and to our experiments, official music videos are edited by taking into account the music structure. Although some experts mentioned that synchronization was often a matter of taste and intuition, we were able to bring out some trends. We showed that the co-occurrence of music and video structural events would vary according to the music and video genres. These elements can be reused to design or improve automatic music-video recommendation systems. For example, if the task is to recommend an illustration video for a **Pop** or **R&B** track, the system is expected to favor candidates that allow high synchronization of the structural events.

However, we have the intuition that other factors may impact the editing style of OMV. In future work, we plan to investigate the role of other metadata, such as release date, artist popularity or harmonic complexity. Although we focused on OMV for this study, we believe that a similar analysis can be conducted on other types of musical videos, e.g. movies or commercials.

6. REFERENCES

- [1] R. Arandjelovic and A. Zisserman, “Look, Listen and Learn,” in *Proceedings of IEEE ICASSP (International Conference on Computer Vision)*, Venice, Italy, 2017.
- [2] O. Gillet, S. Essid, and G. Richard, “On the correlation of automatic audio and visual segmentations of music videos,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 347–355, 2007.
- [3] J.-C. Wang, Y.-H. Yang, I.-H. Jhuo, Y.-Y. Lin, and H.-M. Wang, “The acousticvisual emotion guassians model for automatic generation of music video,” in *Proceedings of ACM Multimedia*, Nara, Japan, 2012.
- [4] F.-F. Kuo, M.-K. Shan, and S.-Y. Lee, “Background Music Recommendation for Video Based on Multimodal Latent Semantic Analysis,” in *Proceedings of ICME (International Conference on Multimedia and Expo)*, San Jose, CA, USA, 2013.
- [5] J. C. Lin, W. L. Wei, and H. M. Wang, “EMV-matchmaker: Emotional temporal course modeling and matching for automatic music video generation,” in *Proceedings of ACM Multimedia*, Brisbane, Australia, 2015, pp. 899–902.
- [6] J. C. Lin, W. L. Wei, J. Yang, H. M. Wang, and H. Y. M. Liao, “Automatic music video generation based on simultaneous soundtrack recommendation and video editing,” in *Proceedings of MMM (International Conference on Multimedia Modeling)*, Reykjavik, Iceland, 2017.
- [7] X.-S. Hua, L. Lu, and H.-J. Zhang, “Automatic music video generation based on temporal pattern analysis,” in *Proc. of ACM Multimedia*, New York City, NY, USA, 2004.
- [8] J. Wang, E. Chng, and C. Xu, “Fully and Semi-Automatic Music Sports Video Composition,” in *Proceedings of ICME (International Conference on Multimedia and Expo)*, Toronto, ON, Canada, 2006.
- [9] J. Wang, E. Chng, C. Xu, Hanqing Lu, and Q. Tian, “Generation of Personalized Music Sports Video Using Multimodal Cues,” *IEEE Transactions on Multimedia*, vol. 9, no. 3, 2007.
- [10] J. S. Downie, “Music Information Retrieval Evaluation eXchange.” [Online]. Available: http://www.music-ir.org/mirex/wiki/MIREX_HOME
- [11] A. Aner and J. R. Kender, “Video Summaries through Mosaic-Based Shot and Scene Clustering,” in *Proceedings of ECCV (European Conference on Computer Vision)*, Copenhagen, Denmark, 2002, pp. –.
- [12] A. Schindler and A. Rauber, “Harnessing music-related visual stereotypes for music information retrieval,” *ACM Transactions on Intelligent Systems and Technology*, vol. 8, no. 2, pp. 1–21, 2016.
- [13] O. Nieto, M. McCallum, M. E. P. Davies, A. Robertson, A. Stark, and E. Egozy, “The Harmonix Set: Beats, Downbeats, and Functional Segment Annotations of Western Popular Music,” in *Proceedings of ISMIR (International Conference on Music Information Retrieval)*, Delft, The Netherlands, 2019. [Online]. Available: <https://archives.ismir.net/ismir2019/paper/000068.pdf>
- [14] C. Inskip, A. Macfarlane, and P. Rafferty, “Music, Movies and Meaning: Communication in Filmmakers’ Search for Pre-existing Music, and the Implications for Music Information Retrieval,” in *Proceedings of ISMIR (International Conference on Music Information Retrieval)*, Philadelphia, PA, USA, 2008. [Online]. Available: <https://archives.ismir.net/ismir2008/paper/000117.pdf>
- [15] O. Gillet, “Transcription des signaux percussifs. Application à l’analyse de scènes musicales audiovisuelles,” Ph.D. dissertation, Ecole Nationale Supérieure des Telecommunications, 2007. [Online]. Available: <https://pastel.archives-ouvertes.fr/pastel-00002805>
- [16] Ruiduo Yang and M. Brown, “Music database query with video by synesthesia observation,” in *Proceedings of ICME (International Conference on Multimedia and Expo)*. Taipei, Taiwan: IEEE, 2004, pp. –.
- [17] P. Mulhem, M. S. Kankanhalli, J. Yi, and H. Hassan, “Pivot vector space approach for audio-video mixing,” *IEEE Multimedia*, vol. 10, no. 2, pp. 28–40, 4 2003.
- [18] A. Schindler, “Multi-Modal Music Information Retrieval: Augmenting Audio-Analysis with Visual Computing for Improved Music Video Analysis,” Ph.D. dissertation, Technische Universität Wien, 2019. [Online]. Available: <https://arxiv.org/abs/2002.00251>
- [19] A. Schindler and A. Rauber, “On the Unsolved Problem of Shot Boundary Detection for Music Videos,” in *Proceedings of MMM (International Conference on Multimedia Modeling)*, vol. 11295 LNCS. Thessaloniki, Greece: Springer Verlag, 2019. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-030-05710-7_43
- [20] B. Jia, J. Lv, and D. Liu, “Deep learning-based automatic downbeat tracking: a brief review,” *Multimedia Systems*, vol. 25, no. 6, 12 2019.
- [21] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “Madmom: A new Python audio and music signal processing library,” in *Proceedings of ACM Multimedia*, New York City, NY, USA, 2016, pp. 1174–1178.
- [22] D. Bogdanov, X. Serra, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, and J. Zapata, “ESSENTIA: an Open-Source Library for Sound and Music Analysis,” in *Proceedings of ACM Multimedia*. New York, NY, USA: ACM Press, 2013.

- [23] S. Böck, F. Krebs, and G. Widmer, “Joint Beat and Downbeat Tracking with Recurrent Neural Networks.” in *Proceedings of ISMIR (International Society for Music Information Retrieval Conference)*, New York City, NY, USA, 2016. [Online]. Available: <https://archives.ismir.net/ismir2016/paper/000186.pdf>
- [24] B. McFee and D. P. Ellis, “Learning to segment songs with ordinal linear discriminant analysis,” in *Proceedings of IEEE ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, Florence, Italy, 2014, pp. 5197–5201.
- [25] T. Souček, J. Moravec, and J. Lokoč, “TransNet: A deep network for fast detection of common shot transitions,” *arXiv preprint arXiv:1906.03363*, 2019. [Online]. Available: <https://arxiv.org/abs/1906.03363>
- [26] O. Nieto and J. P. Bello, “Systematic Exploration Of Computational Music Structure Research.” in *Proceedings of ISMIR (International Conference on Music Information Retrieval)*, New York City, NY, USA, 2016. [Online]. Available: <https://archives.ismir.net/ismir2016/paper/000043.pdf>
- [27] R. Hennequin, J. Royo-Letelier, and M. R. Moussallam Deezer, “Audio Based Disambiguation of Music Genre Tags,” in *Proceedings of ISMIR (International Conference on Music Information Retrieval)*, 2018. [Online]. Available: <https://archives.ismir.net/ismir2018/paper/000163.pdf>
- [28] M. Defferrard, S. P. Mohanty, S. F. Carroll, and M. Salathé, “Learning to Recognize Musical Genre from Audio,” in *Companion of WWW (The Web Conference)*, New York City, NY, USA, 2018, pp. –.
- [29] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The Million Song Dataset,” in *Proceedings of ISMIR (International Conference on Music Information Retrieval)*, Miami, FL, USA, 2011. [Online]. Available: <https://ismir2011.ismir.net/papers/OS6-1.pdf>
- [30] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. Channing Moore, M. Plakal, and M. Ritter, “Audio Set: An Ontology and Human-Labeled Dataset for Audio Events,” in *Proceedings of IEEE ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, New Orleans, LA, USA, 2017.
- [31] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “FMA: A Dataset for Music Analysis,” in *Proceedings of ISMIR (International Society for Music Information Retrieval Conference)*, Suzhou, China, 2017. [Online]. Available: <https://archives.ismir.net/ismir2017/paper/000075.pdf>
- [32] J. You, G. Liu, and A. Perkis, “A semantic framework for video genre classification and event analysis,” *Signal Processing: Image Communication*, vol. 25, no. 4, 2010.
- [33] J. Varghese and K. N. Ramachandran Nair, “A Novel Video Genre Classification Algorithm by Keyframe Relevance,” in *Information and Communication Technology for Intelligent Systems. Smart Innovation, Systems and Technologies*. Singapore: Springer, 2019, vol. 106. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-981-13-1742-2_68
- [34] K. Choroś, “Video Genre Classification Based on Length Analysis of Temporally Aggregated Video Shots,” in *Computational Collective Intelligence. Lecture Notes in Computer Science*. Springer, Cham, 2018, vol. 11056. [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-319-98446-9_48
- [35] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing, “On tempo tracking: Tempogram representation and Kalman filtering,” *Journal of New Music Research*, vol. 29, no. 4, pp. 259–273, 2000. [Online]. Available: <https://www.mcg.uva.nl/papers/mmm-26.pdf>

TABLA GHARĀNĀ RECOGNITION FROM AUDIO MUSIC RECORDINGS OF TABLA SOLO PERFORMANCES

Gowriprasad R¹ Venkatesh V² Hema A Murthy¹ R Aravind¹ Sri Rama Murty K²

¹ Indian Institute of Technology, Madras, ² Indian Institute of Technology, Hyderabad, India

ee19d702@smail.iitm.ac.in, ee19mtech01010@iith.ac.in, hema@cse.iitm.ac.in

aravind@ee.iitm.ac.in, ksrm@ee.iith.ac.in

ABSTRACT

Tabla is a percussion instrument in Hindustani music tradition. *Tabla* learning and performance in the Indian subcontinent is based on stylistic schools called *gharānā-s*. Each *gharānā* is characterized by its unique style of playing technique, dynamics of *tabla* strokes, repertoire, compositions, and improvisations. Identifying the *gharānā* from a *tabla* performance is hence helpful to characterize the performance. This paper addresses the task of automatic *gharānā* recognition from solo *tabla* recordings. We motivate the problem and present different facets and challenges in the task. We present a comprehensive and diverse collection of over 16 hours of *tabla* solo recordings for the task. We propose an approach using deep learning models that use a combination of convolutional neural networks (CNN) and long short-term memory (LSTM) networks. The CNNs are used to extract *gharānā* discriminative features from the raw audio data. The LSTM networks are trained to classify the *gharānā-s* by processing the sequence of extracted features from CNNs. Our experiments on *gharānā* recognition include different lengths of audio data and comparison between various aspects of the task. An evaluation demonstrates promising results with the highest recognition accuracy of 93%.

1. INTRODUCTION

With the vast availability of varied music collections on the digital platform and widespread use of personal digital devices, there is a growing interest in accessing music based on its various characteristics. The limited availability of editorial metadata and annotations led to the need for music information retrieval to automatically extract music's characteristic properties from the audio recordings. Automatic identification of metadata from audio-like, stylistic school recognition—especially in the context of the same genre—is a tough task, even for humans.

This paper addresses the automatic identification of *tabla gharānā-s*, valuable metadata from solo *tabla* recordings. The percussion instrument *tabla* is an integral part

of Hindustani music as it keeps track of rhythm. It is not only used as an accompaniment but also used in solo performances. *Tabla* solo is intricate and elaborate, with a variety of precomposed forms used for developing further elaborations based on the player's stylistic schools called *gharānā-s*. Identifying *gharānā* from a *tabla* performance provides valuable editorial metadata, which helps characterize performance and further musicological analysis. From the standpoint of Music Information Retrieval (MIR), studying and analyzing various *tabla* performance patterns is vital. It has applications in music description, auto-tagging, similarity measures, discovery, informed and enhanced music listening, music training, and computational musicology. We first discuss an overview of *tabla* and its *gharānā-s*.

1.1 *Tabla* and its *gharānā-s*

Tabla consists of a pair of drums, *dāyān* and *bāyān*, a treble drum and a bass drum respectively [1]. The *tabla* repertoire and technique are transmitted from generation to generation by guru-shishya (teacher-student) lineage [2], which is primarily an oral tradition. This guru-shishya lineage gave rise to different schools of *tabla* practice called *gharānā-s*. The word *gharānā* literally means the house of the teacher. *Tabla* solo performance showcases the percussionist's skill with *tabla* developing upon a variety of precomposed compositions such as *thēkā*, *kāyadā*, *palatā*, *rēlā*, *pēškār* and *gaṭ* within the rhythmic framework called *tāl*. Each composition has different functional and aesthetic roles in a solo performance. The developments are intricate and elaborate based on the player's *gharānā*.

There are two major playing styles (*bāj*) in *tabla*, *bandh* (closed) *bāj* and *khulā* (open) *bāj* [3]. In *bandh bāj*, the *tabla* is played more on its border area; hence the stroke resonance is controlled or subdued. In this closed style of playing, importance is given to the sound of *tabla* and speedy progressions. In *khulā bāj*, the *tabla* is played more in the middle portion with open strokes using full palm and fingers. In this style, importance is given to tonal richness of the strokes with resonance. A few open and closed strokes are illustrated in the Figure 1. Based on these two broad playing styles, there are six *gharānā-s* developed namely *Delhi*, *Ajrada*, *Lucknow*, *Banaras*, *Farukhabad*, and *Punjab*. Each *gharānā* is characterized by its unique styles of playing, strokes played on the *tabla*, improvisations, and precomposed patterns. Every *gharānā* has a different approach to technique, and repertoire [3].



© Gowriprasad R, Venkatesh V, Hema A Murthy, R Aravind and Sri Rama Murty K. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Gowriprasad R, Venkatesh V, Hema A Murthy, R Aravind and Sri Rama Murty K, "Tabla Gharānā Recognition from Audio Music Recordings of *Tabla* Solo Performances", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

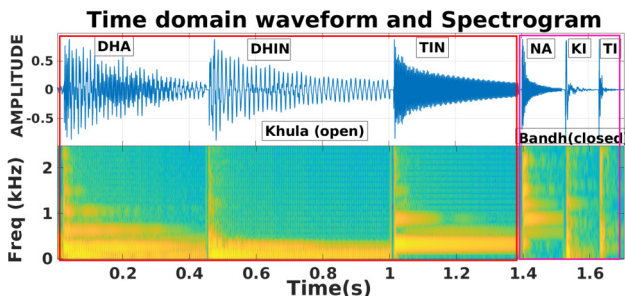


Figure 1. Illustration of a few tabla strokes.

1.2 Related Work

Most of the research related to Indian percussion focused mainly on stroke transcription and sequence modeling. Gillet *et al.* [4] focused on stroke transcription, and Chordia [1] extended the work by using additional features with larger-diverse dataset. Chordia *et al.* [5] used predictive models for tabla stroke sequence by making use of tabla syllables. Samudravijaya *et al.* [6] used hidden Markov models for recognition of tabla strokes. Kuriakose *et al.* [7] and Anantapadmanabhan *et al.* [8] worked on transcription of mridangam strokes. Chordia *et al.* [9] have worked on multiple viewpoints on modeling tabla sequences. Gupta *et al.* [10] identifying syllabic percussion patterns from the transcribed tabla audio.

The works on music style and classification are as follows. Vidwans *et al.* [11] used melodic contours to classify vocal style and classify cultural music using melodic features [12]. Agarwal *et al.* [13] did a comparative study of Indian and western music forms. Tang *et al.* [14] used hierarchical LSTMs for music genre classification. Gessle *et al.* [15] did a comparative analysis of Convolutional neural network (CNN) and Long short-term memory (LSTM) networks for music genre classification. Gogineni *et al.* [16] worked towards mridangam artiste identification from solo mridangam audio. We address a similar task of tabla gharānā recognition from solo tabla audios using the CNN-LSTM approach. As per our knowledge, identifying the tabla gharānā has not been attempted before.

The rest of the paper is organized as follows. The challenging factors influencing the task are categorically mentioned motivating the experiment. The dataset collected for the task is described. We formulate the task and explain the proposed model architecture. Multiple experiments addressing various facets of the task are described. The experimental results are analyzed and discussed.

2. TABLA GHARĀNĀ RECOGNITION

Tabla gharānā recognition is a task of identifying the stylistic schools of tabla given a solo tabla recording. We address the task by processing the composition-specific sequential information from the audio. We discuss the various aspects of the task and the collected dataset.

2.1 Motivation and Challenges

We motivate the task of tabla gharānā recognition by getting insight into the factors influencing the task. The factors include both the supporting and challenging aspects.

Gottlieb [17] mentions three factors for comparing the similarities and differences in the playing: (1) Sound production, that is, quality and the technique used, (2) Repertoires, and (3) Rhythmic practices. The technique and the rhythmic practices differ from artiste to artiste. Compositions bearing the gharānā distinctions are based on the repertoires from each gharānā-s.

To get the expert’s advice on the factors influencing the task, we consulted four tabla maestros. A few common opinions provided by the artistes on the task are mentioned here. Artistes nowadays would have learned from several teachers from different gharānā-s. Different gharānā styles will also influence their playing style [2]. Hence it is not straightforward to classify the artiste as coming from a particular gharānā in the present era. Therefore the repertoires [3] and compositions with some specific combination of certain strokes become the distinguishing factor in identifying the gharānā. This means that it is more straightforward and valid to recognize the gharānā based on the compositions rather than the artistes’ playing styles.

Considering the vast diversity of the tabla solo repertoire and its practices, we list out a few possible challenging aspects influencing the system. (1) Tonic variability - Tabla is a pitched harmonic percussive instrument tuned to a specific tonic in a concert [18, 19]. As the tonic varies, the properties of the sound like harmonics, timbre, tone, etc., also vary. Thus the feature vectors representing the same stroke with different tonic will also change, challenging the system performance. In an ideal scenario, the gharānā distinctions are independent of tonic variability. (2) Artiste variability - Each artiste has their approach to techniques, finger postures, individual nuances with extempore development. Thus there exist an invariable influence of the artiste variability. (3) Composition variability - A few gharānā-s share similarities between their compositions. This is because some gharānā-s are the offshoots of others [2], share common stroke sets and techniques. There are equally significant instances where the compositional theme is entirely different within the gharānā itself. Thus composition variability has a strong influence on gharānā distinction.

2.2 Dataset Description

To experiment with various shades of tabla gharānā requires a diverse collection of annotated audio data. As there is no dataset available for this task, we collected solo tabla recordings from commercial audio CDs, live recordings from the artistes’ archives, and online sources. This corpus consists of tabla solo from 18 different artistes with at least 20 years of tabla playing experience. The full-length tabla solo consists of different compositions played one after the another with a few cycles of ṭhēkās in between adjacent compositions, marking the start and end of a composition [3]. These compositions are usually from different gharānā-s. Thus we consider the aspect of repertoire, in so far as it has a bearing on the gharānā distinctions [2]. Kāyadā-s are the extendable compositions elaborated upon a theme from a particular gharānā through paltā-s. Paltā-s are the variations of original phrase or theme [2]. Each

Gharānā name (ID)	No. of Artists	No. of Tonics	No. of Compositions	Durations hh:mm:ss
Ajrada (A)	7	5	21	2:23:58
Banaras (B)	8	9	35	3:09:01
Delhi (D)	10	5	16	2:32:11
Farrukhabad (F)	8	9	23	2:36:50
Lucknow (L)	7	7	36	2:34:52
Punjab (P)	7	5	42	2:53:27

Table 1. Dataset Description.

kāyadā rendition usually lasts for three to five minutes. The kāyadā compositions exists in all the gharānā-s [2]. Gaṭ-s and chakradhār are preset compositions that last for more than one to two metrical cycles. Hence in these compositions, one can observe the theme of the gharānā as well as the artiste’s playing style.

Professional performers were employed to listen and extract the kāyadā, gaṭ, and chakradhār sections from the audio by marking the start and end points. In addition, the datasets from Gupta et al. [10], and Rohit et al. [20] are included in our dataset. Then four tabla maestros from different gharānā-s were requested to listen to these audio segments and give the ground truth gharānā labels. By doing so, we were able to collect around 16 hours of gharānā annotated audio. The details of the dataset are described in Table 1. The complete dataset is heterogeneous with artiste variability (18 artistes), tonic variability, tempo variability, and soft harmonium or sarangi accompaniment. Most of the audio played predominantly in ūntāl consisting of 16 beats. There are a few audios from the other tāls, such as jhaptāl and ektāl. The audios are sampled at 44.1 kHz. We use the downsampled version of the data at 16 kHz.

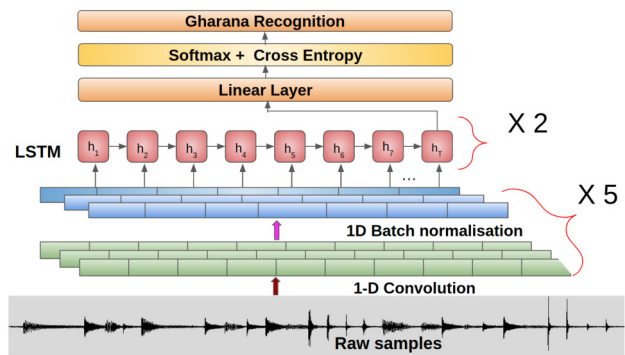
3. PROPOSED APPROACH FOR GHARĀNĀ RECOGNITION

The factors influencing the task necessitates a system that addresses various aspects and automatically perform gharānā recognition. With recent advances in Deep Neural Networks (DNNs), several tasks like music style recognition [21], music genre recognition [22] have achieved performance improvement with DNNs over the statistical methods. The success of DNNs motivates us to formulate a neural network based gharānā recognition system addressing various facets of the task.

We formulate the task by proposing a CNN-LSTM model. The model is trained by examples of different compositions from various artistes. The CNNs are trained to extract the local discriminative features pertaining to different stroke sounds from the raw audio. Tabla solos are developed, improvised, and elaborated upon a theme through a series of variations according to rhythmic practices [3]. Thus some strokes co-occur more often than others in the tabla compositions. Hence it is essential to train the models by encoding the sequence information. The LSTM networks are trained to classify the gharānā-s by processing the sequence of extracted features from CNNs.

3.1 Proposed model Architecture

The proposed CNN-LSTM method has a single training stage, which takes the raw samples and models posterior


Figure 2. Proposed model architecture.

distribution over the six classes. The overall architecture of CNN-LSTM for gharānā recognition is shown in Figure 2. It has two components: a 1-dimensional convolution neural network (1D-CNN) and a long short-term memory (LSTM) network. Five layers of 1D-CNN are used in the model. The first 1D-CNN component takes the raw audio samples as input. The CNN section acts as a feature extractor, performing convolutions on samples along the time axis. It produces a feature vector for every 10 ms with a reduced frame rate. This sequence of feature vectors of 10 ms stride acts as input for the LSTM.

Every CNN layer is followed by BatchNormalization [23] and ReLU activation [24]. The kernel size, number of kernels, stride, and padding of each convolution layer are $(1 \times 10, 256, 5, 3)$, $(256 \times 8, 256, 4, 2)$, $(256 \times 4, 256, 2, 1)$, $(256 \times 4, 256, 2, 1)$, $(256 \times 4, 256, 2, 1)$, respectively. This 1D-CNN component configuration is adapted from contrastive predictive coding encoder [25]. The LSTM component consists of two-layer LSTMs, each with 256 dimensions. The final hidden activation of 256 dimensions is fed to the log-softmax classification layer with six units to get the gharānā predictions. The model is trained using Adam optimizer [26] with a batch size of 32 and the learning rate of 0.01. We have used learning schedule of dropping the learning rate to half of current whenever the validation loss doesn’t decrease [27]. We also experimented by adding one more CNN with 5% dropout, and an LSTM layer. An implementation of the proposed architecture is available ¹.

4. EXPERIMENTS AND RESULTS

4.1 Human Assessment

For the task of gharānā recognition from audio recordings, we need to know the duration that encapsulates good gharānā specific information. From the experts feedback, we got to know that each gharānā has certain compositions practiced and played by almost all the tabla experts nowadays. Hence, if the tabla artistes were to recognize the gharānā by listening, they do it within two to three seconds if the composition is known. If they do not know the composition that is played, it takes some time to analyze the combination of strokes in the theme for a few cycles and predict the stylistic school gharānā. Thus to have a human assessment on time taken to recognize and the accuracy, a survey experiment is conducted.

¹ <https://git.io/JE0o0>

Student ID (Years of practice)	Avg Time taken (secs)	Accuracy
S1 (12)	9 - 12	85%
S2 (8)	11 - 15	76%
S3 (12)	7 - 10	87%
S4 (16)	6 - 8	97%
S5 (12)	8 - 10	92%
S6 (11)	8 - 11	89%
S7 (10)	10 - 13	83%
Average	8.4 - 11.3	87%

Table 2. Senior tabla students survey (Human baseline).

Since the four tabla maestros were also consulted for preparing the ground truth annotations, it is not proper to have the same experts for the human baseline survey experiment. Thus, we conducted a survey experiment on seven senior tabla students. These students have more than eight years of tabla practice. The students are asked to listen to the compositions from our collected dataset and predict the gharānā labels. The time taken for prediction is noted by pausing the audio at the instance of prediction. The details of the survey are tabulated in Table 2. One can observe that around 87% of the recordings were predicted correctly by the students on average. The average time taken to predict each recording was found to be eight to eleven seconds.

4.2 Baseline Experiments

As there are no previous works available on gharānā recognition, we adapted CNN maxpooling LSTM model from [22] to achieve baseline results. The model use two CNN layers and one LSTM layer. MFCC + derivatives + double derivatives (MFCC_ΔΔ) features are fed at the input. Each audio file is segmented to the 10-sec duration and treated as individual examples. MFCC_ΔΔ features are extracted with a window size of 25 ms and a frameshift of 5 ms. Each example is represented by (2000, d) feature matrix. Each feature vector is of dimension d=57. It is the sum of MFCC (19)+ Δ+ ΔΔ (d=19+19+19=57). 70% of the entire audio is used for training the models. 15% of the audio was used for development and test each. This model is considered as baseline for the following reasons: (1) The model is used for a similar task of music style classification, and a comparison with the state-of-the-art systems are made [22]. (2) This uses a simpler architecture similar to our proposed system. (3) Handcrafted MFCC_ΔΔ features are used as the input.

4.3 CNN-LSTM Experiments

We conducted seven different sets of experiments with the proposed CNN-LSTM model addressing various aspects. The training, development, and test data split is the same as the baseline experiment (70 – 15 – 15). The motivation for different experiments and their results are described.

4.3.1 Experiment with segment duration

Each theme is played initially at the speed of or double the speed of the original tempo for one or two cycles and is then played at four times the original tempo [3]. The underlying tempo is flexible in Indian music and differs from artiste to artiste. Some themes have a structure of three

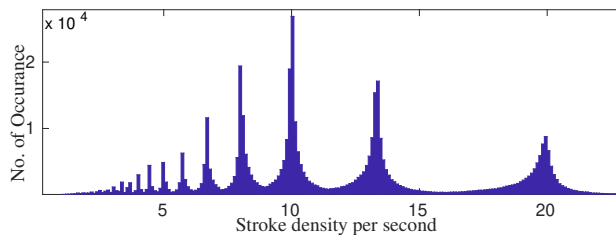


Figure 3. Histogram of stroke density per second, Mean=11.72, Median=10.25.

Trial	Weighted F1 Score					Avg
	1	2	3	4	5	
3s	0.68	0.67	0.69	0.67	0.65	0.67
5s	0.79	0.77	0.78	0.8	0.79	0.78
10s	0.85	0.86	0.85	0.84	0.86	0.85
15s	0.85	0.90	0.89	0.88	0.91	0.89
15s DA	0.89	0.95	0.91	0.93	0.93	0.92

Table 3. Cross validation scores data length experiments.

strokes per beat (triple rhythm) [3, 18], which has nearly 12 strokes per beat at the fourth speed. Thus on average, around 8-12 strokes are played per beat at the fourth speed depending on the compositions. The mean and median of stroke density per second as obtained in the histogram, Figure 3 are found to be 11.72 and 10.25 respectively.

Repetition of the phrases is commonly observed in a musical concert. While playing at the fourth speed, one can hear at least one to two cycles of the theme or paltās in around 10 seconds of duration. This is evident as around 100 strokes are played in 10 seconds, which spans two to three cycles of the composition theme length. Since almost the whole structure of a theme can be found in about 10 seconds, it is adequate to segment the audio recordings into 10 to 15 second segment and treat them as separate examples. The average time duration as needed in the human assessment is also around 10 seconds. Thus we limited the length of train examples to be 10 to 15 seconds.

The entire data is segmented into 10s and 15s segments with 10% overlap with adjacent segments and treated as separate examples. The examples are randomly shuffled, and the train-dev-test split is done. Two models are trained and tested with 10s and 15s segments. The test segments are further chunked to 3s and 5s and treated as separate test examples in the next two experiments. This is done to analyze and get an insight into how much audio length is required for the system to recognize the gharānā specific sequence information. The longer the audio, the better the recognition. These experiments are 5-fold cross-validated, and the weighted F-1 scores are reported in Table 3. F-1 score is the harmonic mean of precision and recall.

4.3.2 Data Augmentation (DA)

To increase the training data diversity, data augmentation is performed. Speed perturbation with the factor of 0.9X and 1.1X (10% variation) without altering the pitch is done on the entire data using HPSS-TSM method [28, 29]. To get the value of the speed perturbation factor that does not alter the original data structure, we referred to the histogram of

Weighted F1 Score					
Exp	Model~Variant	1	2	3	Avg
15 IA	Std. Network	0.70	0.63	0.66	0.66
	+1 LSTM	0.73	0.68	0.70	0.70
15s IC	Std. Network	0.37	0.45	0.47	0.43
	+1 LSTM	0.38	0.46	0.49	0.44
15s IT	Std. Network	0.69	0.71	0.63	0.67
	+1 CNN	0.72	0.76	0.68	0.72
	f_0 = 260.63Hz	0.77	0.80	0.76	0.77

Table 4. Cross validation scores IA, IT, IC.

the average number of strokes per second.

Figure 3 shows the histogram depicting the statistical estimate of stroke density per second over the entire dataset. To get the stroke density per second, initially, the audio was preprocessed by computing the Hilbert envelop of linear prediction residual on the raw audio as described in [30]. Then the onset locations of each stroke are computed using a spectral flux onset detector [31]. The computed onset locations are considered for further analysis without any post-processing as the ground truth onset locations for the entire data are not available. The first difference of the onset locations gives the inter-onset interval (IOI). The inverse of IOI gives the stroke density.

The modes in the histogram have sharp peaks. One can infer that the underlying original tempo is almost uniform across the recordings. Thus a larger deviation from the original tempo is not advisable. The audio did not sound realistic if the speed perturbation was large. Therefore we restricted to only a 10% deviation from the original tempo. This increased the size of the dataset to three times the original. Since better performance was observed on 15s segments of the test data earlier, we choose to experiment with the same on augmented data as well.

4.3.3 Inter Artiste(IA) and Composition(IC) Experiments

The presentation of a particular composition by different artistes differs due to varied rhythmic practices and extempore development. We conducted IA experiment where the train and test artistes are distinct. Each compositional theme is unique in its own way. Thus to have an insight into various compositions, we performed an IC experiment. We took the help of senior students to have three sets where the compositions are diverse and performed the experiment. We experimented by adding a layer of LSTM to the model. This task is performed to check if an extra LSTM layer could learn more sequence information.

4.3.4 Inter Tonic (IT) Experiment

Different tonic essentially means the tabla itself is different. The tonic value is not a differentiating factor between the gharānā-s. We performed IT analysis to explore the diversity of the dataset. The train and test data tonics (tabla) are distinct in this experiment. We also tried two variants in IT experiments. (1) Adding one more CNN layer to the model with 5% dropout. This task is done to check if a bigger model could perform better in the IT experiment setup. (2) Preprocessing the data by normalizing the tonic of all the audio to 260.63 Hz (C_4), and then perform the

Method	Models	Weighted F1	Accuracy
Proposed CNN-LSTM	10s	0.85	0.86
	15s	0.89	0.90
	15s DA	0.92	0.93
	15s IA	0.66	0.67
	15s IC	0.43	0.45
	15s IT	0.67	0.67
	15s IA + DA	0.69	0.71
	15s IC + DA	0.44	0.45
Baseline (Sec 4.2)	15s IT + DA	0.7	0.7
	CNN-LSTM	0.53	0.55
	10s-(MFCC)		

Table 5. Results with different experiments.

experiment. The length of the audio is initially varied by a factor of $(\frac{260.63}{f_0})X$ using HPSS-TSM [28, 29] method, where f_0 is the original tonic value. Then it is resampled to play at the original speed, which in turn changes the pitch. This task is done to check if the preprocessing aids the performance in the IT experiment. The IA, IC, IT experiments are performed on 15s segments. The 3-fold cross-validation scores are reported in Table 4. The "Std.Network" in Table 4 refers to the proposed model with 5-CNN and 2-LSTM layers.

5. ANALYSIS AND DISCUSSION

5.1 Performance analysis

Table 5 summarizes the experiments' average performance scores after n-fold cross-validation. One can observe that the proposed CNN-LSTM model tested with 15s data segments has the best performance of 90%. Performance dropped during IA, IC, and IT experiments. This is evident as the experiments are on the challenging aspects influencing the system. From Table 4, we can see that the addition of an extra CNN layer improved the performance of IT by 5%. An extra LSTM layer improved the performance of IA by 4% and IC by 1%. Data augmentation improved the performance of 15s, IA, IT by 3% and IC by 1%. This indicates that tuning a bigger model aids the system to be robust to tonic and artiste variation. This emphasizes that the composition variability has a larger influence on the task than the artiste variability. This is also evident from the artistes' feedback described in section 2.1.

The experiment with preprocessing the data by normalizing the tonic favored but did not add a larger value. If this preprocessing approach helped, the results should have been on par with the 15s experiment with the random train-test split. Few reasons which we are able to examine are as follows. (1) The artifacts and distortions due to time scale modification and resampling. (2) The bass drum is usually not set to any tonic. Shifting the pitch of the treble drum invariably shifts the pitch of the bass drum with the same factor. This does not sound realistic. It is to be observed that the performance improved nearly by 10% by preprocessing the audio. This indicates that suitable preprocessing aids the system performance. We observed that adding one more CNN and LSTM layer to the proposed model did not improve the performance in the random shuffle exper-

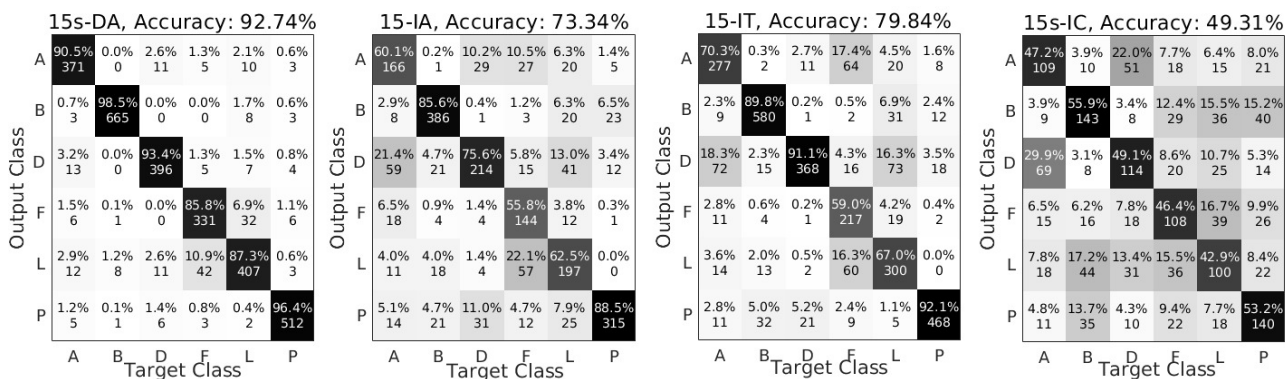


Figure 4. Confusion matrices depicting the best performing experiments on different facets.

iments with segment duration. Thus we reported the average scores from the proposed standard model in Table 5.

One can observe a clear improvement in the performance by 32% by comparing the 10s experiment with the baseline. This indicates that the features extracted from the raw audio by the CNN model have better discriminatory information than the handcrafted MFCC features.

5.2 Confusion analysis

The confusion matrices depicting the results for the major experiments are displayed in Figure 4. Figure 5 shows the t-SNE visualization of test data embedding from the CNN-LSTM model extracted from the 15s-DA samples. These embeddings are tapped at the output of the LSTM layer just before the linear layer. The visualization of embeddings and confusion matrices depicts the similarities and uniqueness of gharānā-s among each other. Different clusters are marked for the benefit of reference. One can observe that Punjab gharānā embeddings form a clear separation from the others (C1). This is evident as the Punjab gharānā has had a separate existence and is unique compared to the other ones [3].

We can observe an inevitable overlap and confusion between Lucknow and Farrukhabad embeddings (C2) as well as Delhi and Ajrada embeddings (C3). This overlap is acceptable as the Ajrada, and Farrukhabad gharānā-s are the offshoots of Delhi and Lucknow gharānā-s respectively [2]. It is also a fact that Delhi and Ajrada have similar playing styles, and both belong to bandh bāj [3]. Lucknow and Farrukhabad also lot of similarities traditionally, and both belong to khulā (open) bāj [17].

The founders of Lucknow gharānā hailed from Delhi gharānā [2]. Both Farrukhabad and Banaras styles originated from Lucknow gharānā. Thus we can observe the Lucknow gharānā embeddings getting overlapped with others. This confusion is also observed in various experiments addressing different facets. One can also observe the clear separation of Banaras gharānā embeddings (C4). This is evident as Banaras gharānā has a lot of changes to Lucknow gharānā getting influenced by Pakhawaj bōls [2, 17] and has a unique way of playing the bass drum [3]. Farrukhabad repertoire exploited entire vocabulary of the instrument [3]. Thus Farrukhabad embeddings can be seen confused with others.

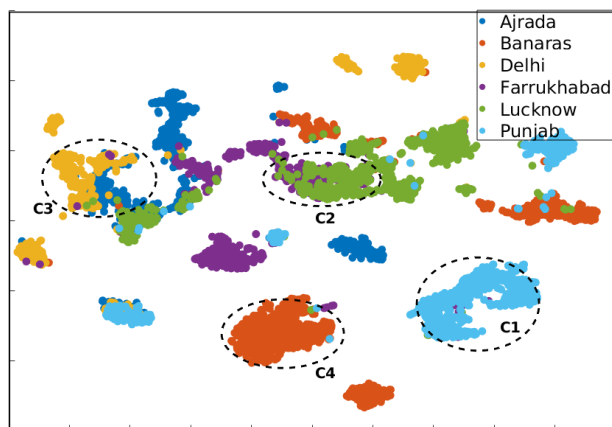


Figure 5. t-SNE visualization of test data embedding extracted from the 15s CNN-LSTM Model.

By visualizing the embeddings, and confusion matrices, one can verify many traditional similarities and differences between the gharānā-s as depicted and explained in the sources [2, 3, 17, 18]. Thus one can claim that the model has been trained in a positive way.

6. CONCLUSION

We addressed an unexplored problem of recognizing tabla gharānā, the stylistic schools of tabla, by proposing a deep learning model. The task used around 16 hours of gharānā class annotated data consisting of various compositions played by contemporary artistes. We motivate the problem and present different facets and challenges in the task. CNN and LSTMs in tandem are trained to extract gharānā discriminative features from the raw audio data and classify gharānā-s by processing the sequence information. Different experiments addressing various aspects of the task are performed. Additionally, proposed variants improved the performance in respective experiments. The system performance is comparable with the human assessment. The proposed CNN-LSTM model delivered promising results with the highest accuracy of 93% and a relative improvement of 31% over the considered baseline.

As a first attempt, we started by proposing a CNN-LSTM model for the task. Incorporating the tonic information during training will be beneficial. We aim to extend the dataset by incorporating other tabla compositions such as pēškār, tukda, etc., and perform the experiments.

7. ACKNOWLEDGMENTS

The authors are grateful to the tabla maestros Ramesh Dhannur, Kiran Yavagal, Aneesh Pradhan, Aditya Kalyankur, and Anirudh Sharma for their support and help. We are thankful to Ajay Srinivasamurthy for his support and timely guidance. We thank Prof. Preeti Rao and Swapnil Gupta for sharing their dataset with us.

8. REFERENCES

- [1] P. Chordia, "Segmentation and recognition of tabla strokes." in *Proc. 6th International Society for Music Information Retrieval (ISMIR), 2005*.
- [2] S. Bagchee, *Nād: Understanding rāga music*. Eeshwar, 1998.
- [3] A. Pradhan, *Tabla: A Performer's Perspective*. Book-Baby, 2011.
- [4] O. Gillet and Richard, "Automatic labelling of tabla signals," in *Proc. 4th International Society for Music Information Retrieval (ISMIR), 2003, Baltimore, USA*.
- [5] P. Chordia, A. Sastry, and S. Şentürk, "Predictive tabla modelling using variable-length markov and hidden markov models," *Journal of New Music Research*, vol. 40, no. 2, pp. 105–118, 2011.
- [6] K. Samudravijaya, S. Shah, and P. Pandya, "Computer recognition of tabla bols," Technical report, Tata Institute of Fundamental Research, Tech. Rep., 2004.
- [7] J. Kuriakose, J. C. Kumar, P. Sarala, H. A. Murthy, and U. K. Sivaraman, "Akshara transcription of mridangam strokes in carnatic music," in *Twenty First National Conference on Communications (NCC) 2015*.
- [8] A. Anantapadmanabhan, A. Bellur, and H. A. Murthy, "Modal analysis and transcription of strokes of the mridangam using non-negative matrix factorization," in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2013*.
- [9] P. Chordia, A. Sastry, T. Mallikarjuna, and A. Albin, "Multiple viewpoints modeling of tabla sequences." in *Proc. 11th International Society for Music Information Retrieval (ISMIR), 2010*, p. 11th.
- [10] S. Gupta, A. Srinivasamurthy, M. Kumar, H. A. Murthy, and X. Serra, "Discovery of syllabic percussion patterns in tabla solo recordings," in *Proc. 16th International Society for Music Information Retrieval (ISMIR); 2015 Oct 26-30; Málaga, Spain.[Málaga]*. p. 385-391.
- [11] A. Vidwans, K. K. Ganguli, and P. Rao, "Classification of indian classical vocal styles from melodic contours," in *Serra X, Rao P, Murthy H, Bozkurt B, editors. Proceedings of the 2nd CompMusic Workshop; 2012 Jul 12-13; Istanbul, Turkey. Barcelona: Universitat Pompeu Fabra; 2012*. p. 139-146, 2012.
- [12] A. Vidwans, P. Verma, and P. Rao, "Classifying cultural music using melodic features," in *2020 International Conference on Signal Processing and Communications (SPCOM)*. IEEE, 2020, pp. 1–5.
- [13] P. Agarwal, H. Karnick, and B. Raj, "A comparative study of indian and western music forms." in *Proc. 14th International Society for Music Information Retrieval (ISMIR), 2013*, pp. 29–34.
- [14] C. P. Tang, K. L. Chui, Y. K. Yu, Z. Zeng, and K. H. Wong, "Music genre classification using a hierarchical long short term memory (lstm) model," in *Third International Workshop on Pattern Recognition*, vol. 10828. International Society for Optics and Photonics, 2018, p. 108281B.
- [15] G. Gessle and S. Åkesson, "A comparative analysis of cnn and lstm for music genre classification," 2019.
- [16] K. Gogineni, J. Kuriakose, and H. A. Murthy, "Mridangam artist identification from taniavartanam audio," in *Twenty Fourth National Conference on Communications (NCC) 2018*. IEEE, pp. 1–6.
- [17] R. S. Gottlieb, *Solo tabla drumming of North India: Its repertoire, styles, and performance practices*. Motilal Banarsidass Publishers, 1993.
- [18] S. K. Saxena, *The Art of Tablā Rhythm: Essentials, Tradition, and Creativity*. Sangeet Natak Akademi, 2006, no. 8.
- [19] A. Anantapadmanabhan, J. Bello, R. Krishnan, and H. Murthy, "Tonic-independent stroke transcription of the mridangam," in *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.
- [20] M. Rohit and P. Rao, "Acoustic-prosodic features of tabla bol recitation and correspondence with the tabla imitation." in *Interspeech*, 2018, pp. 1229–1233.
- [21] B. Kumaraswamy and P. Poonacha, "Deep convolutional neural network for musical genre classification via new self adaptive sea lion optimization," *Applied Soft Computing*, p. 107446, 2021.
- [22] D. Ghosal and M. H. Kolekar, "Music genre recognition using deep neural networks and transfer learning." in *Interspeech*, 2018, pp. 2087–2091.
- [23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.
- [24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.
- [25] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

- [26] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [27] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, “End-to-end asr-free keyword search from speech,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1351–1359, 2017.
- [28] J. Driedger, M. Müller, and S. Ewert, “Improving time-scale modification of music signals using harmonic-percussive separation,” *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 105–109, 2014.
- [29] S. Yong, S. Choi, and J. Nam, “Pytsmod: A python implementation of time-scale modification algorithm,” in *Extended Abstracts for the Late-Breaking Demo Session of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [30] R. Gowriprasad and K. S. R. Murty, “Onset detection of tabla strokes using lp analysis,” in *International Conference on Signal Processing and Communications (SPCOM)*. IEEE, 2020, pp. 1–5.
- [31] S. Dixon, “Simple spectrum-based onset detection,” *MIREX 2006*, p. 62, 2006.

NAVIGATING NOISE: MODELING PERCEPTUAL CORRELATES OF NOISE-RELATED SEMANTIC TIMBRE CATEGORIES WITH AUDIO FEATURES

Lindsey Reymore

Emmanuelle
Beauvais-Lacasse

Bennett Smith

Stephen McAdams

Schulich School of Music, McGill University, Canada

lindsey.reymore@mail.mcgill.ca, emmanuelle.beauvais-lacasse@mail.mcgill.ca, bennett.smith@mcgill.ca, stephen.mcadams@mcgill.ca

ABSTRACT

Audio features such as inharmonicity, noisiness, and spectral roll-off have been identified as correlates of “noisy” sounds; however, such features are likely involved in the experience of multiple semantic timbre categories of varied meaning and valence. This paper examines the relationships among audio features and the semantic timbre categories *raspy/grainy/rough*, *harsh/noisy*, and *airy/breathy*.

Participants ($n = 153$) rated a random subset of 52 stimuli from a set of 156 ~2-second orchestral instrument sounds from varied instrument families, registers, and playing techniques. Stimuli were rated on the three semantic categories of interest and on perceived playing effort and emotional valence. With an updated version of the Timbre Toolbox (R-2021 A), we extracted 44 summary audio features from the stimuli using spectral and harmonic representations. These features were used as input for various models built to predict mean semantic ratings (*raspy/grainy/rough*, *harsh/noisy*, *airy/breathy*) for each sound.

Random Forest models predicting semantic ratings from audio features outperformed Partial Least-Squares Regression models, consistent with previous results suggesting non-linear methods are advantageous in timbre semantic predictions using audio features. In comparing Relative Variable Importance measures from the models among the three semantic categories, results demonstrate that although these related semantic categories are associated in part with overlapping features, they can be differentiated through individual patterns of feature relationships.

1. INTRODUCTION

Several audio features have been identified as correlates of “noisy” sounds, including inharmonicity, spectral flatness, spectral centroid, and spectral roll-off. However, not all types of noise are semantically equal: when timbre categories are nuanced, “noisy” features may be correlates of

multiple semantic categories with varied meanings and even varied valence. Through interviews and rating tasks, Reymore and Huron [1] built a 20-dimensional model of musical instrument timbre qualia. Intriguingly, the final model included three timbre dimensions plausibly related to noise components—*shrill/harsh/noisy*, *raspy/grainy* and *airy/breathy*—while a further two dimensions appeared to potentially refer to harmonicity and/or a lack of “noisy” features—*pure/clear* and *focused/compact*. Speculating on correlates of these semantic categories, Reymore and Huron [1] noted that while noise has been often associated with negative valence and high physical exertion as in Wallmark, Iacoboni, Deblieck, and Kendall [2], noise components in breathy timbres, typically measured in speech research with harmonic-to-noise ratio (HNR), may convey a sense of proximity or intimacy that carries positive valence. Thus, a feature such as HNR may be important for multiple semantic categories. Although semantic categories can share acoustic correlates, varying relationship strengths with audio features may create distinctive, perceptible patterns for listeners that are associated with varying semantic content.

The current study examined three semantic categories derived from Reymore and Huron’s model: *raspy/grainy/rough*, *harsh/noisy*, and *airy/breathy*. The aim was to determine how these semantic categories may be distinguished based on their relationships with audio features. We used linear and non-linear approaches to model semantic ratings using audio features, with the goal of uncovering distinctive acoustic signatures for each semantic category. Predictors included spectral and harmonic features from a recently updated version of the Timbre Toolbox (R-2021A) [3]. Among these features, several have been associated with noise in previous literature and so were of particular interest for the interpretation of our results (see Section 2.2).

We first describe the methods used in the rating study and in audio feature extraction. These features are then used in models to predict semantic ratings. McAdams et al. [4] used Timbre Toolbox audio features to model affective qualities of timbre using both linear and nonlinear modeling approaches and found that the nonlinear approach was more successful. Accordingly, we compare linear and nonlinear methods for analysis to assess whether this observation holds in a similar dataset. We consider our findings with regard to comparative relative importance



© L. Reymore, E. Beauvais-Lacasse, B. Smith, and S. McAdams. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** L. Reymore, E. Beauvais-Lacasse, B. Smith, and S. McAdams, “Navigating noise: Modeling perceptual correlates of noise-related semantic timbre categories with audio features”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

values of features for each of the semantic categories. Our results illustrate in detail relationships between low-level features extracted using MIR techniques and high-level semantic features whose validity and reliability have been established through perceptual studies [1, 5-6].

2. METHODS

2.1 Semantic Ratings

2.1.1 Participants

Participants ($n = 153$; $F = 95$, $M = 57$, other = 1), were recruited using the internet platform Prolific. Participants were on average 32 years of age ($SD = 11.2$, range = 18–68); 41 of the 153 self-identified as musicians using the single-question measure from the Ollen Musical Sophistication Index [7]. As identified through Prolific’s screening process, all participants were native English speakers. Participants provided informed consent and were compensated for their participation.

2.1.2 Materials

Stimuli consisted of 156 approximately 2-second sound clips of single notes (pitch class C) played by various orchestral instruments, normalized and matched for loudness by the researchers. Stimuli were taken from three sound banks: Vienna Symphonic Library (VSL) [8], McGill University Master Samples (MUMS) [9] and conTimbre [10]. The stimulus set included 42 instruments playing in 5 registers (C2–C6) using both traditional and unconventional playing techniques. The unconventional playing techniques that were selected generated additional noise components (such as bowing a violin at the bridge). In selecting stimuli, we aimed to sample as widely as possible from the semantic space of interest—that is, to sample sounds representing high, moderate, and low ratings on all given categories of interest. To help achieve this goal, the final stimulus selection process was guided by the results of a pilot study ($n = 10$) of 46 sounds.

Principal Component Analysis of the pilot results, in which sounds were rated on individual rather than grouped terms (e.g., separate ratings were made for “airy” and “breathy”) confirmed the appropriateness of grouped terms. Specifically, a three-component PCA model with promax rotation demonstrated strong loadings which aligned with the groupings of terms used in the main study (*raspy/grainy/rough, harsh/noisy, airy/breathy*).

2.1.3 Procedure

To avoid an overly long experiment, the stimulus set was partitioned into three subsets for each group of three participants, each of whom rated one-third of the stimulus set (52 sounds). This resulted in 51 complete sets of rating data on the entire stimulus set.

Participants rated how applicable each semantic category was to a given stimulus using a continuous sliding scale from 1 (*does not describe at all*) to 7 (*describes extremely well*), where the midpoint was labeled *describes moderately well*. Participants also rated valence (*negative to positive*) and perceived playing effort (*little to no exertion to high exertion*).

Ratings were made in separate blocks for each scale; participants thus rated their subset of 52 stimuli a total of five times. At the beginning of each trial (except the first in each block), the stimulus was automatically played, and participants could play the stimulus again as many times as desired. The presentation order of the scales and the presentation order of stimuli within each block were randomized. The experiment took approximately 30 minutes to complete.

2.2 Audio Feature Extraction

To investigate relationships between semantic timbre categories and acoustic features, we used an updated version of the Timbre Toolbox [3]. The Timbre Toolbox calculates spectral, temporal, and spectrotemporal acoustic features from an audio signal in Matlab [11]. First, input representations of the signal are computed. Then, both scalar and time-series features are extracted from different input representations. Lastly, the Timbre Toolbox calculates interquartile range (IQR) and median values of time-series features. These values represent the central tendency and variability of the audio features, respectively [12].

For this study, we used the STFT (Short-Time Fourier Transform) and HARM (Harmonic) input representations. The STFT is a spectrotemporal representation obtained using a sliding-window analysis over the audio signal. Then, the amplitude spectrum of the STFT is used as one of the representations to derive the audio features. HARM (sinusoidal harmonic model) is a harmonic representation that uses frame analysis to estimate slowly varying amplitude and frequency of individual harmonics [12].

Each stimulus was analyzed in the Timbre Toolbox to determine the median and IQR of audio features. Several features are derived from both the STFT and HARM representations. Results reported here for these overlapping features were taken from the STFT representation. In total, we used medians and IQRs of 22 features provided by the Timbre Toolbox [3], listed in Table 1.

Several of the features provided by these two representations in the Timbre Toolbox have been previously associated with noise, including inharmonicity, noise energy, noisiness, spectral flatness, HNR, and spectral centroid.

Audio Features from Timbre Toolbox	
Inharmonicity	Spectral Spread
Noisiness	Spectral Centroid
Noise Energy	Spectral Variation
Harmonic Energy	Spectral Roll-Off
Pitch	Spectral Decrease
Harmonic-to-Noise Ratio	Spectral Skewness
Tristimulus 1	Spectral Flux
Tristimulus 2	Spectral Kurtosis
Tristimulus 3	Spectral Flatness
Harmonic Odd-to-Even Ratio	Spectral Crest
Harmonic Spectral Deviation	Spectral Slope

Table 1. List of Timbre Toolbox audio features; median and IQR values were extracted for each feature.

Inharmonicity is the degree to which the frequencies of overtones depart from multiples of the fundamental frequency; inharmonicity within the auditory signal manifests as noise [2], [13]. Noise energy is calculated as the energy of the signal not explained by stable partials [3]. (Note that this definition differs from that of the feature in previous versions of Timbre Toolbox, which calculates noise energy based only on stable harmonic partials [12]). Noisiness refers to the ratio of noise energy to total energy; high noisiness values indicate a signal that is mainly nonharmonic. Spectral flatness, which has also been associated semantically with “noisy” timbres [2], [14], roughly discriminates noise from harmonic content because sinusoidal components produce a peak in the spectrum, whereas white noise produces a flat spectrum [12].

Phonetic processes, such as breathy and creaky voice, may also offer insight into the acoustic correlates that underlie semantic timbre categories. Keating et al. [15] found that different acoustic features or combinations of these features, including HNR, characterized different varieties of creaky voice. HNR is an assessment of the ratio between periodic and non-periodic components comprising a segment of an acoustic signal [15].

Spectral centroid is the center of mass of the power spectrum of an acoustic signal and is related to the perception of brightness [16]. Because Wallmark [14] suggests that increased brightness is associated with the perception of physical exertion, and because we anticipate that our semantic categories of interest will be related to perceived exertion, spectral centroid may be a relevant correlate for one or more categories.

3. ANALYSIS

3.1 Semantic Ratings

All analyses reported in this paper were carried out in R, version 4.0.5 [17]. Cronbach’s alpha was calculated among complete sets of ratings using the *alpha* function in the *psych* package [18], where each set of ratings was completed by three participants (see 2.1.3). All alpha values were greater than .9, indicating excellent internal consistency. Mean semantic ratings were distributed over a large portion of the 1–7 rating scale for each category, suggesting that our stimulus set was successful in representing the semantic space of interest. Ranges among mean ratings and Cronbach’s alpha values are reported in Table 2.

With Holm corrections implemented by the *corr.test* function in the *psych* package [18],¹ we observed significant Pearson correlations between *harsh/noisy* and *rough/raspy/grainy*, $r(154) = .53$ and between *harsh/noisy* and *airy/breathy*, $r = -.54$. The correlation between *rough/raspy/grainy* and *airy/breathy* was not significant.

Semantic category	Min	Max	Cronbach’s α
<i>Raspy/grainy/rough</i>	1.63	6.72	.97
<i>Harsh/noisy</i>	2.12	6.45	.95
<i>Airy/breathy</i>	1.50	5.65	.93

Table 2. Range of mean ratings among stimuli and Cronbach’s alpha for each semantic category.

¹ This method is used for all correlations reported in this paper.

3.2 Models

Following McAdams et al. [4], we performed both linear and nonlinear modeling. Scaled and centered values for the audio features from the Timbre Toolbox were used to predict mean semantic ratings; separate models were generated for each of the three semantic categories. The linear method of analysis used was partial least-squares regression (PLSR), a supervised learning algorithm that takes a dimension-reduction approach including a Principal Component Analysis process. Unlike principal component regression, however, PLSR takes both the predictor and outcome variables into account when building the linear model. This kind of statistical approach can handle data that exhibit multicollinearity and thus was appropriate for our dataset. Random forest regression was used as the nonlinear method of analysis. A random forest (RF) is a supervised machine learning algorithm that builds multiple decision trees by randomly selecting observations and specific variables and then averaging the results [19]. Both types of models were built with the *caret* package [20].

R^2 was computed on the complete dataset using ten-fold cross-validation repeated three times. To obtain Q^2 , we applied a further five-fold cross-validation to each model. The observations were divided into five subsets; the model was trained on four out of the five subsets and then predicted the last remaining subset. The subsets were rotated to ensure that the training and prediction steps were applied to every combination of the subsets. Within each of the train-test subsets, models were trained using a 10-fold cross-validation repeated three times. This process also produced the RMSE values that are reported below. Table 3 contains values for R^2 , Q^2 , and RMSE for each model.

Semantic category	Model Type	R^2	Q^2	RMSE
<i>Raspy/grainy/rough</i>	RF	.82	.78	.47
	PLSR	.64	.56	.70
<i>Harsh/noisy</i>	RF	.56	.54	.69
	PLSR	.43	.28	.93
<i>Airy/breathy</i>	RF	.45	.43	.78
	PLSR	.36	.29	.89

Table 3. Average R^2 and RMSE values from PLSR and RF models for all three semantic categories.

3.3 Relative Variable Importance

We calculated Relative Variable Importance (RVI) using the *varImp* function from the *caret* package [20]. RVI values are reported in Tables 4 and 5. RVI for the PLSR is based on the weighted sums of the absolute regression coefficients. The weights are a function of the reduction of the sums of squares across the number of PLS components and are computed separately for each outcome [20]. For the RF, the mean squared error is recorded on the out-of-bag portion of the data and after permuting each predictor variable. Differences between these values are averaged across all trees and normalized by the standard deviation of the differences [21]

<i>Airy/breathy</i>		<i>Harsh/noisy</i>		<i>Raspy/grainy/rough</i>	
Feature	Value	Feature	Value	Feature	Value
1. Harm Spec Dev IQR	100.00	Spectral Decrease Med	100.00	HNR Med	100.00
2. Harm Spec Dev Med	72.27	Spectral Centroid Med	64.79	Noisiness Med	92.05
3. Spectral Roll-Off Med	68.76	Pitch Med	54.92	Spectral Variation IQR	74.63
4. Spectral Spread Med	64.50	Spectral Spread Med	50.88	Harmonic Energy Med	73.24
5. Spectral Centroid Med	62.17	Spectral Roll-Off Med	50.03	Inharmonicity Med	73.17
6. Spectral Flux IQR	58.61	Spectral Variation IQR	46.63	Pitch Med	72.26
7. Spectral Slope IQR	58.24	Harm Spec Dev IQR	40.62	Spectral Slope Med	70.63
8. Tristimulus 1 Med	56.17	Harm Spec Dev Med	39.97	Spectral Crest Med	66.84
9. Spectral Flux Med	52.86	HNR Med	31.52	Tristimulus 3 Med	63.57
10. Spectral Skewness Med	48.44	Spectral Decrease IQR	31.36	Spectral Variation Med	59.95

Table 4. Top 10 important variables and their respective relative variable importance values for each semantic category using partial least-squares regression.

<i>Airy/breathy</i>		<i>Harsh/noisy</i>		<i>Raspy/grainy/rough</i>	
Feature	Value	Feature	Value	Feature	Value
1. Odd:Even Ratio Med	100.00	Spectral Decrease Med	100.00	HNR Med	100.00
2. Odd:Even Ratio IQR	72.34	Spectral Spread Med	87.68	Inharmonicity IQR	62.11
3. Harm Spec Dev IQR	71.73	Spectral Roll-Off Med	72.04	Spectral Variation Med	54.34
4. Spectral Roll-Off Med	49.60	Spectral Centroid Med	71.21	Noisiness Med	40.84
5. Spectral Flux IQR	40.98	Spectral Spread IQR	62.49	Spectral Variation IQR	39.43
6. Spectral Spread Med	30.97	Spectral Variation IQR	37.58	Tristimulus 3 Med	21.00
7. Spectral Centroid Med	29.13	Spectral Flatness IQR	31.09	Inharmonicity Med	18.98
8. Spectral Variation IQR	27.13	Spectral Variation Med	26.13	Pitch Med	18.81
9. Spectral Skewness IQR	19.02	Spectral Flatness Med	24.44	Harmonic Energy Med	5.11
10. Tristimulus 1 IQR	16.67	Noisiness IQR	24.24	Tristimulus 1 Med	3.13

Table 5. Top 10 important variables and their respective relative variable importance values for each semantic category using random forest regression.

4. DISCUSSION

4.1 Relative feature importance profiles for semantic categories

Partial least-squares and random forest models were built to predict mean semantic ratings from extracted audio features. These models were most successful in predicting ratings of *raspy/grainy/rough* (Q^2 : RF .78, PLSR .56); models predicting *harsh/noisy* (Q^2 : RF .54, PLSR .28) and *airy/breathy* (Q^2 : RF .43, PLSR .29) were also moderately successful. McAdams et al. [4] found that a nonlinear approach produced better models than a linear approach when modeling affective qualities using Timbre Toolbox audio features. We also observed an advantage for the nonlinear method, as random forest models consistently yielded higher Q^2 values and lower RMSE values than the linear PLSR models. Because random forest regression offered the more successful models, the current discussion of results focuses on the random forest models unless otherwise noted.

HNR median was the most important variable in predicting ratings of *raspy/grainy/rough*. Spectral decrease median was the most important feature for predicting *harsh/noisy*, and harmonic odd-to-even ratio median was the most important feature for *airy/breathy*. Spectral variation IQR was in the top ten important features predictive of ratings for all three semantic categories.

Patterns of variable importance were distinct for each semantic category. Particularly among the RF models, features ranking especially high in relative importance were often unique to one of the three semantic categories, though some important features were overlapping between categories. This suggests that specific combinations of features may be important for the perception of varying semantic information.

One method of comparing feature importance among the three semantic categories is to choose a minimum importance value in order to define what constitutes a “relevant” feature. Relevant features—i.e., the features exceeding that threshold for each category—can then be compared across models. For example, spectral variation IQR is the only feature with an RVI value over 25 for all three semantic categories, suggesting that it is at least moderately relevant for models of all three categories.

In this manner, we can identify which features are uniquely “relevant” to each semantic category, where “relevant” is defined by the researcher as referring to features with RVI greater than a given value. A threshold of 25 was set for the purpose of this analysis, based on the distribution of RVI values and tractability for discussion. Definitions of “relevance” in similar interpretations can be defined with respect to the goals of the interpretation.

With this definition in mind, uniquely relevant features for *raspy/grainy/rough* include the HNR median, inharmonicity IQR, and noisiness median. Of these features,

Pearson correlations (r) demonstrate that median HNR and median noisiness were strongly negatively correlated in the dataset, $r(154) = -.96, p < .001$.

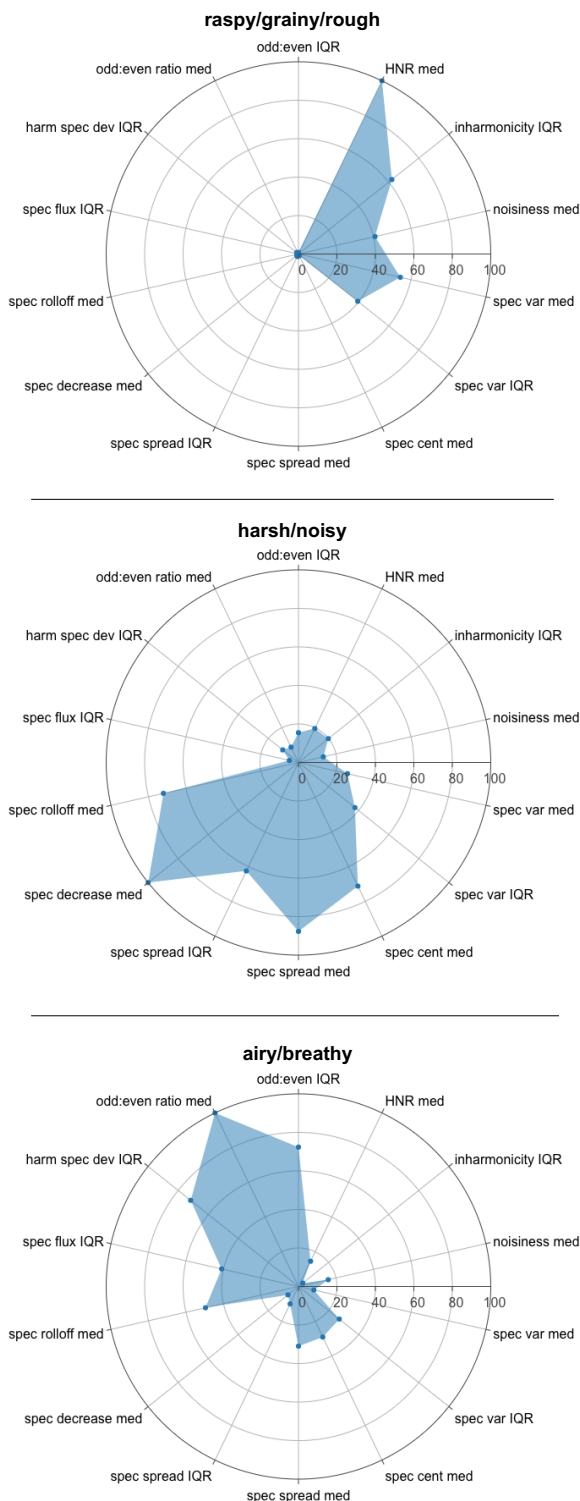


Figure 1. Radar plots of Relative Variable Importance measures for random forest models of each semantic category.

For *harsh/noisy*, uniquely relevant features include the spectral decrease median, spectral spread IQR, and spectral flatness IQR. Unique features for *airy/breathy* include the harmonic odd-to-even ratio (median and IQR, which are intercorrelated at $r = .99$), harmonic spectral deviation IQR, and spectral flux IQR.

The spectral variation median was relevant for both *raspy/grainy/rough* and *harsh/noisy*. *Harsh/noisy* and *airy/breathy* also shared relevant features—spectral roll-off median, spectral spread median, and spectral centroid median; these three features were strongly correlated among our stimuli set (roll-off/spread, $r = .97$; roll-off/centroid, $r = .95$, centroid/spread, $r = .91$).

RVI values for the 14 most important features across semantic categories from the random forest models are illustrated in the radar plots of Figure 1. The radius represents RVI; features are listed in the same order around the circles for all three plots in order to facilitate visual comparisons of semantic categories.

4.2 Noise-related features

In Section 2.2, we reviewed several features which previous literature suggests may be relevant for our semantic categories of interest, including inharmonicity, noisiness, noise energy, spectral flatness, spectral centroid, and HNR. We will now consider these specific features in relation to our semantic rating results.

Inharmonicity IQR was of particular relative importance in both linear and nonlinear models predicting *raspy/grainy/rough*, but neither the IQR nor the median were ranked highly in importance for either of the other semantic categories. Spearman correlations (ρ) suggest a robust monotonic relationship between mean ratings of *raspy/grainy/rough* and inharmonicity IQR, $\rho(154) = .78$; correlation with the median is $\rho = .59$. *Harsh/noisy* values demonstrate a moderate correlation with inharmonicity IQR, $\rho = .40$, but not with the median.

The noisiness median also received high RVI values for both linear and nonlinear models predicting *raspy/grainy/rough*, but this feature (and the corresponding IQR) received relatively low RVI values for the other semantic categories. However, both the noisiness median and noisiness IQR correlated positively with ratings of all three semantic categories. Spearman correlations were strongest for *raspy/grainy/rough*: median, $\rho = .75$; IQR, $\rho = .55$. *Harsh/noisy* demonstrated moderate correlations, median, $\rho = .39$; IQR, $\rho = .38$, and *airy/breathy* was weakly correlated but not significant, median, $\rho = .15, p = .06$; IQR, $\rho = .14, p = .08$.

Noise energy somewhat unexpectedly was not given particular importance in any of the models and demonstrated relatively weak correlations with semantic ratings.

HNR was the most important contributor to models of *raspy/grainy/rough*. Neither median nor IQR were significantly correlated with *airy/breathy*, but both were correlated with *raspy/grainy/rough*, median, $\rho = -.77$; IQR, $\rho = .40$, and *harsh/noisy*, median, $\rho = -.42$; IQR, $\rho = .35$, where higher ratings in these categories were associated with a lower HNR median but a higher HNR IQR. Given this feature's use in speech research in relation to breathy voice, the lack of significant correlation with *airy/breathy* was

surprising. Because higher HNR is also associated with higher ratings for *raspy/grainy/rough* and *harsh/noisy*, one explanation for this may be that our dataset contained many stimuli with high HNR that were very *raspy/grainy/rough* and/or very *harsh/noisy* but not *airy/breathy*. While the HNR-*airy/breathy* correlation was not significant, it was in the anticipated direction, $\rho = -.11$, $p = .17$.

Spectral flatness figured in *harsh/noisy* models but was not highly important for any of the three categories. Spearman correlations suggest positive monotonic relationships between spectral flatness (both median and IQR) with *rough/raspy/grainy*, median, $\rho = .34$; IQR, $\rho = .39$, and *harsh/noisy*, median, $\rho = .49$; IQR, $\rho = .50$. These relationships with *airy/breathy* were both weaker and in the opposite direction, median, $\rho = -.26$; IQR, $\rho = -.22$.

Spectral centroid, the correlate for semantic brightness, figured as relatively important in models for *harsh/noisy* and *airy/breathy*, but not for *raspy/grainy/rough*. The median correlated positively with ratings of *harsh/noisy*, $\rho = .58$, and negatively with *airy/breathy*, $\rho = -.45$. The IQR correlated positively with both *harsh/noisy*, $\rho = .43$, and *raspy/grainy/rough*, $\rho = .36$.

In summary, among our features of interest, we found that inharmonicity IQR, noisiness median, and HNR median seemed to be most specifically associated with *raspy/grainy/rough*, although some moderate relationships among these features can also be identified with *harsh/noisy*. Spectral flatness was weakly to moderately correlated to all three categories but did not figure prominently in models. Spectral centroid was primarily associated with *harsh/noisy* and *airy/breathy*. Both median and IQR for noisiness were correlated positively with all three categories, whereas for roll-off, flatness, and centroid, correlations for *raspy/grainy/rough* and *harsh/noisy* were in the opposite direction than those for *airy/breathy*.

4.3 Variance in valence and perceived exertion associated with semantic categories

Rating results demonstrated that the three semantic categories varied in perceived valence and playing exertion. While ratings of *raspy/grainy/rough* and *harsh/noisy* were negatively correlated with valence, $r(154) = -.90$ and $r = -.61$, respectively, ratings of *airy/breathy* were positively correlated with valence, $r = .31$. *Raspy/grainy/rough* and *harsh/noisy* were also moderately correlated with increased exertion, $r = .50$ and $r = .46$, respectively; however, ratings of *airy/breathy* did not correlate significantly with perceived playing exertion. Thus, we can consider *rough/raspy/grainy* to be associated strongly with negative valence and moderately with perceived exertion. *Harsh/noisy* is moderately associated with negative valence and perceived exertion, and *airy/breathy* is moderately associated with positive valence but not associated with exertion.

These descriptive statistics demonstrate how two semantic categories with relatively similar perceptual relationships to emotional valence and exertion may be differentiated by patterns of relationships with audio features; for example, the most important predictors of *raspy/grainy/rough* include HNR, inharmonicity, and

noisiness, while the most important predictors of *harsh/noisy* include spectral decrease, spread, roll-off, and centroid. We can also see that categories with differing relationships to emotional valence and perceived exertion may both have relevant relationships with a given feature. Such overlapping relationships may be in either opposite directions—for example, *harsh/noisy* is positively associated with spectral roll-off median and spectral flatness median—or in the same direction—for example, noisiness median and HNR median.

5. CONCLUSION

This research examined associations between spectral and harmonic audio features and the timbre semantic categories *raspy/grainy/rough*, *harsh/noisy*, and *airy/breathy*. We collected semantic ratings from 153 participants for 156 orchestral instrument sounds varying in register, instrument family, and playing technique. Ratings confirmed that the three semantic categories were distinct, and that categories differed in their relationships with exertion and valence.

We built partial least-squares and random forest models predicting mean semantic ratings for each category. Across the three categories, nonlinear random forest regression models outperformed linear partial least-squares regression models. The spectral and harmonic features used in this paper were most successful for predicting *rough/raspy/grainy*, followed by *harsh/noisy*. Models were least successful in predicting ratings for *airy/breathy*.

In comparing Relative Variable Importance measures from the models among the three semantic categories, results demonstrate that although these semantic categories are associated in part with overlapping features, they can be differentiated through individual patterns of feature relationships. Among plausibly noise-related features, we observed that inharmonicity IQR, noisiness, and HNR were in general related strongly to *raspy/grainy/rough* and moderately to *harsh/noisy*. Spectral roll-off, flatness, and centroid demonstrated moderate relations to *harsh/noisy* and *airy/breathy*, but in opposite directions. Finally, the directions of associations with HNR and noisiness were the same across all three semantic categories but varied in strength.

These results contribute to efforts to bridge understandings of timbre in MIR and music cognition by clarifying the relationships between low-level audio features and nuanced semantic categories generated from perceptual studies. The methods presented here may be used to build feature profiles of other semantic categories beyond those related to noise. Furthermore, our findings may be useful in timbre synthesis, in that they can help guide the creation of sounds with specific semantic content. Such applications to synthesis may be especially relevant to audio branding and electroacoustic composition.

6. REFERENCES

- [1] L. Reymore and D. Huron, "Using auditory imagery tasks to map the cognitive linguistic dimensions of musical instrument timbre qualia," *Psychomusicology: Music, Mind, and Brain*, vol. 30, no. 3, pp. 124–144, June 2020.
- [2] Z. Wallmark, M. Iacaboni, C. Deblieck, and R. Kendall, "Embodied listening and timbre: Perceptual, acoustical, and neural correlates," *Music Perception*, vol. 35, no. 3, pp. 332–363, 2018.
- [3] S. Kazazis, et al. Timbre Toolbox R-2021A. [Manuscript in preparation.]
- [4] S. McAdams, C. Douglas, and N. N. Vempala, "Perception and modeling of affective qualities of musical instrument sounds across pitch registers," *Frontiers in Psychology*, vol. 8, no. 1, June 2017.
- [5] L. Reymore, "Characterizing prototypical musical instrument timbres with Timbre Trait Profiles," In *Musicae Scientiae*, vol. 27, no. 1, April 2021.
- [6] L. Reymore, "Variations in timbre qualia with register and dynamics in the oboe and French horn," In *Empirical Musicology Review*, In Press.
- [7] J. Ollen, "A criterion-related validity test of selected indicators of musical sophistication using expert ratings," Ph.D. dissertation, Music, OSU, Columbus, OH, 2006.
- [8] Vienna Symphonic Library GmbH, "Vienna Symphonic Library." 2011. Available: <http://vsl.co.at>
- [9] F. Opolko and J. Wapnick, "McGill University Master Samples (3CDs)." 1987. McGill University.
- [10] T. Hummel, "conTimbre Database Project." 2012. Available: <http://www.contimbre.com>
- [11] MATLAB 2021a, The MathWorks, Inc., Natick, Massachusetts, United States.
- [12] G. Peeters, B. L. Giordano, P. Susini, N. Misdariis, and S. McAdams, "The Timbre Toolbox: Extracting audio descriptors from musical signals," *The Journal of the Acoustical Society of America*, vol. 130, no. 5, pp. 2902–2916, November 2011.
- [13] M. Caetano, C. Saitis, and K. Siedenberg, "Audio content descriptors of timbre," in *Timbre: Acoustic, Perception, and Cognition*, 2019, pp. 297–333.
- [14] Z.T. Wallmark, "Appraising timbre: embodiment and affect at the threshold of music and noise," Ph.D. dissertation, Musicology, UCLA, Los Angeles, CA, 2014.
- [15] P. Keating, M. Garellek, and J. Kreiman, "Acoustic properties of different kinds of creaky voice," in *ICHPHs*, Glasgow, UK, 2015, pp. 2–7.
- [16] E. Schubert and J. Wolfe, "Does timbral brightness scale with frequency and spectral centroid?," *Acta Acustica United with Acustica*, vol. 92, no. 5, pp. 820–825, September/October 2006.
- [17] R: A language and environment for statistical computing. (2021). [Online]. Available: <https://www.R-project.org/>
- [18] W. Revelle. psych: Procedures for Personality and Psychological Research. (2020). [Online]. Available: <https://CRAN.R-project.org/package=psych> Version = 2.1.3
- [19] G. Biau and E. Scornett, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.
- [20] M. Kuhn. The Caret Package. (2012) [Online]. Available: <http://cranrproject.org/web/packages/caret/caret.pdf>
- [21] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. *R News*, vol. 2, no. 3, pp. 18–22.

QUANTITATIVE USER PERCEPTIONS OF MUSIC RECOMMENDATION LIST DIVERSITY

Kyle Robinson Dan Brown

David R. Cheriton School of Computer Science, University of Waterloo, Canada

kyle.robinson@uwaterloo.ca, dan.brown@uwaterloo.ca

ABSTRACT

Diversity is known to play an important role in recommender systems. However, its relationship to users and their satisfaction is not well understood, especially in the music domain. We present a user study: 92 participants were asked to evaluate personalized recommendation lists at varying levels of diversity. Recommendations were generated by two different collaborative filtering methods, and diversified in three different ways, one of which is a simple and novel method based on genre filtering. All diversified lists were recognised by users to be more diverse, and this diversification increased overall recommendation list satisfaction. Our simple filtering approach was also successful at tailoring diversity to some users. Within the collaborative filtering framework, however, we were not able to generate enough diversity to match all user preferences. Our results highlight the need to diversify in music recommendation lists, even when it comes at the cost of "accuracy".

1. INTRODUCTION

Music recommender systems play an ever-increasing role in individual listening habits as music consumption moves to online platforms and services such as Spotify, and Apple Music. Along with this growth has been an equivalent growth in research on how to better tailor music recommendations to match individual users' preferences and habits. Much of this research, especially in academia, depends on offline evaluations and metrics calculated on existing known listening histories as a proxy for real user satisfaction and list evaluation.

Along with metrics measuring the overall predictive ability (accuracy metrics) are offline evaluation metrics that measure additional qualities such as *Diversity*. These metrics are less standardised than accuracy metrics, and numerous definitions of each have been used in previous research [1, 2]. There is little public research on the effect of, and preference for, recommendation list diversity of actual music listeners.

We asked 92 online participants to evaluate three differently diversified personal recommendation lists from each of two different collaborative filtering recommendation algorithms. Participants were also asked questions about their preference for novel music and diversity as they relate to concepts discovered in the first study. We identified that accuracy and individual song ratings differed from overall *list satisfaction*, and our implementation of *inner diversity* filtering resulted in higher levels of list satisfaction despite no significant decrease in perceived diversity. We also found that participants were less satisfied with the recommendations from a neural network model's recommendations despite its superior performance in offline testing accuracy. Finally, we found that none of our diversification methods resulted in too diverse recommendations, suggesting that we were not able to match all users diversity preferences: some users wanted more diversity in their recommendations than we could provide.

2. BACKGROUND AND RELATED WORK

With novelty, coverage, and serendipity, diversity has long been identified as an important metric in providing satisfying automated recommendations to users [1]. Diversity has its origins in information retrieval, where it was used as a solution to ambiguous searches [3]. In recommender systems, diversity prevents over-personalization of recommendations to users in order to increase user satisfaction [1, 2]. Recommender system diversity has often been described as the opposite of *similarity* [4,5]. One definition of diversity in *music recommender systems* is intra-list diversity (ILD): the average pairwise dissimilarity of items by a similarity metric [5,6]. There are many alternatives: modifications of ILD [7,8] and novel approaches that do not rely on pairwise dissimilarity [9–11].

Vargas *et al.* use the distributions of genres in a user listening histories and recommendations to satisfy three *properties* of diversity: genre coverage, redundancy, and size-awareness [10]. Oliveira *et al.* similarly seek to *Pareto-optimize* a set of self-selected *aspects* of diversity: contemporaneity, locality, gender, and genre [12]. None of these methods are evaluated with any users, though they outperform other methods on the defined metrics.

Anderson *et al.* found that use of personalized recommendations leads to a reduction in overall consumption diversity, diversity was related to higher user retention, and



users’ consumption diversity was increased by a migration away from personalized recommendations [13]. Holtz *et al.* found similar results with podcast recommendations [14]. Finally, Hansen *et al.* examine different methods of shifting consumption on one large music platform towards more diversity [15]. Although these works provide vital information on the current diversity of users on music platforms, they do so using commercial metrics such as retention and consumption. We provide a more foundational view of diversity for user satisfaction and perception.

3. METHODOLOGY

We trained our own recommendation models to control for all aspects of recommendation and diversification.

3.1 Recommendation Overview

3.1.1 Data

We extended the publicly available LastFM data set created for our previous work by retroactively topping up each user’s listening history [16]. For each LastFM username we collected up to 10,000 new song Listening Events (LEs) starting from July 2020 and working back to their latest LE in the existing data set. Users who did not have any new LEs during this period were removed from the data set. Tracks in this data set are identified using unique artist and track name tuples. The total un-processed and updated data set consists of 520,134,112 unique LEs, and 15,804,356 unique artist-track tuples recorded by 50,440 users over a period of roughly 2 years during 2019 and 2020.

To remove noise, we eliminated tracks which were listened to 10 or less times. This filtering resulted in a drastic reduction in unique artist-track tuples to 2,817,819 (82.2% decrease), while only modestly reducing the number of LEs to 488,528,514 (6% decrease) and the number of users to 50,437 (< 0.01% decrease).

In the filtered data set, the median number of LEs per user is 9,857, the 25th percentile is 4,663, and the 75th percentile is 14,277. The user-track-interaction matrix contains 176,151,310 non-zero entries (play counts) across 2,817,819 unique tracks, resulting in a 50,437×2,817,819-sparse matrix. Entries in this matrix correspond to the number of unique times a user (row) played the track (column). An anonymized version of this updated data is available upon request.

Data was split into training, validation, and test sets using weak generalization, where user-item interactions are sampled at random from the entire dataset to form the subsets. This differs from the strong generalization used by Liang *et al.* which samples entire users resulting in each user occurring in only one data subset [17]. We used weak generalization because matrix factorization can not efficiently deal with a large number of unseen users. Data was split into train, validation, and test subsets by successively splitting by a ratio of 85/15.

3.1.2 Algorithms

We chose two collaborative filtering recommendation algorithms designed for implicit feedback data sets: Alternating Least Squares matrix factorization (ALS) [18], and Variational Autoencoders for Collaborative filtering (MultVAE) [17]. The results of both algorithms are presented as a form of replication for diversity, and we plan to contrast the overall performance of both models in another work. We provide a brief overview of how these algorithms work in practice, and refer readers to the original papers for detailed descriptions and mathematical processes.

ALS uses classical matrix factorization, and has been used frequently in recommendation research and production [18–20]. The algorithm generates recommendations by factorizing a large sparse matrix of user and item playcounts to compute a low-rank matrix approximation. The factorizations give a vector for each user and item. The product of any user vector and item vector represents relevance. Vectors for unseen users can be calculated using their listening history and the latent item factorizations. This method is known as the *fold-in* method [21].

In addition to generating recommendations, the column factorizations give latent features representing each song.

MultVAE is a modern neural network approach based on a Variational Autoencoder architecture. It is the only neural network approach identified by Dacrema *et al.* to outperform basic top-*n* recommendation algorithms using various measurements of accuracy on commonly used benchmark data sets [22]. MultVAE passes a dense input vector with length equal to the total number of recommendable items (*x*) through an encoder (g_ϕ) to a lower-dimensional latent representation (*z*), and then through a decoder (f_θ) which has an inverse architecture to the encoder. The general architecture of MultVAE is:

$$x \rightarrow g_\phi \rightarrow z \rightarrow f_\theta \rightarrow x'$$

The authors suggest that g_ϕ and f_θ consist of 0 or 1 densely connected perceptron layers with a dimensionality of 600, and the dimensionality of *z* to be 200. The dimensionality of *x* and *x'* is equal to the number of items in the database. The vector *x'* gives expected play counts which we can sort in decreasing order and select top-*n* items from.

3.1.3 Hyperparameter Optimization and Training

For the general performance analysis, hyperparameter optimization, and baseline comparison we adopt binary Normalized Discounted Cumulative Gain (NDCG) [4]. Discounted Cumulative Gain (DCG) is based on recall and defined as: $DCG = \sum_{i=1}^k \frac{rel_i}{\log(i+1)}$ where *rel* is a binary value representing whether the recommendation at rank *i* appears in the unseen portion of the users listening history. The denominator then *discounts* the relevance based on how far from rank 1 it appears. NDCG for one user is: $NDCG = \frac{DCG}{DCG'}$ where DCG’ is the ideal DCG: *rel_i* is always equal to 1. Total NDCG@*k* is the average value across all users for some defined list length *k*.

We optimized ALS hyperparameters using randomized search over 60 iterations. The best performance on valida-

Model	Validation NDCG@100	Test NDCG@100
ALS	0.217	0.325
MultVAE	0.223	0.349

Table 1: Final results of both recommendation algorithms. Note that the test results reflect models trained using the combined training and validation data sets.

tion data was achieved using 224 factors, $\lambda = 1$, $\alpha = 1$, after 98 iterations.

Our implementation of MultVAE was based on the original author’s Tensorflow 1 implementation, and a PyTorch implementation by James Le¹. Due to the large number of unique tracks in our data set, full cross-validation of MultVAE architectures was not computationally feasible. We instead trained a number of different models and architectures concurrently based on the original authors results. The best performance on validation data was achieved using 0 hidden encoder/decoder layers, annealing cap of 1, 10000 annealing steps, learning rate of 0.001, and batch size of 500 over 250 epochs. We implemented early stopping based on NDCG@100, but it was not triggered. The final dimensionality of the model was:

$$[2, 817, 819] \longrightarrow [200] \longrightarrow [2, 817, 819]$$

Both models were retrained on the combined training and validation data, and evaluated on the unseen test data. The final evaluation results can be seen in Table 1.

ALS, generates a new latent user vector using their listening history and the existing latent item vectors. We multiply this new user vector with all item vectors to generate item relevance. For MultVAE we feed the user’s listening history through the trained network and obtain a new vector containing each item’s relevance. The relevance values from each list are then sorted in decreasing order to form top- n lists.

3.2 Item Features

We calculate diversity with latent item features generated from ALS matrix factorization. To lessen the effect of popularity on latent features, each track’s feature vector was ℓ_2 -normalized to unit-length. Item distances were computed using simple Euclidean distance.

3.3 Music Recommendation Lists

We used three different techniques to generate top-10 music recommendation lists for both recommendation algorithms, giving 6 different top-10 recommendation lists per user. Recommendation lists generated using ALS are prefixed with *als*, and recommendation lists generated using MultVAE are prefixed with *vae*.

¹The original authors’ code can be found at <https://github.com/dawenl/vae> cf. Permission to use Le’s code was obtained through email correspondence; his implementation can be found at <https://github.com/khanhnamle1994/MetaRec>

3.3.1 Control (*als, vae*)

Our control recommendation lists consist of the raw ranked output from each recommendation algorithm after removing tracks which appeared in the user’s listening history. This gives the metrics reported in Table 1.

3.3.2 Maximally Diverse (*als_max_div, vae_max_div*)

We generated these recommendation lists using the greedy ILD diversification method described by Ziegler *et al.* using $\beta = 1$ [6]. This greedy diversification algorithm starts with the maximally diverse track from some larger recommendation list; we start with the top-1000 recommendations from each model. The algorithm incrementally adds the track maximally distant from the already selected tracks until the list is of the desired length. This method ensures that the tracks are not only maximally distant from all other tracks, but that the final recommendation list traverses multiple extremes in the item feature space. We do not consider the relevance ranking within the top-1000 recommendations when generating diverse recommendation lists; this corresponds to setting $\beta = 1$ in the original diversification process.

3.3.3 Filtered Diverse (*als_filt_div, vae_filt_div*)

We also use filtered diverse lists, where the top-1000 recommendations are filtered based on the user’s existing listening history. This aims to better align recommendations with user preference for *inner diversity* identified in existing research [16]. We considered two methods for filtering recommendations: feature clustering, and genre filtering.

For feature clustering, we tried to remove tracks too distant from existing LEs. We clustered user LE history into n groups, and filtered recommendations which fall outside the clustering. This approach proved unsuccessful.

For genre filtering we remove recommendations in genres which do not appear in the user’s existing listening history. We used Spotify artist genre tags, and defined a track’s genres as the genres of that track’s artist retrieved from Spotify.

For each user, we identified all genres which appear in the user’s listening history and their frequencies. Next, we find the most diverse track among the top-1000 recommendations (the first track in the Maximally Diverse list) and its genres. The user’s genre list is searched for this track’s genres, and we save the lowest frequency found (or 0 if none) to be the user’s genre threshold. We remove from the top-1000 recommendation list any tracks with a single genre either not in the user’s hash table, or with a frequency below the found threshold. We run greedy diversification on the filtered list.

3.4 User Study

Our interactive user study consisted of a pre-interaction survey on personal music consumption, discovery, and preference, followed by 6 personalized top-10 music recommendation lists as described in Section 3.3. The recommendation lists included a 5-point Likert evaluation for each track, and questions on the recommendation list as

a whole using the same 5-point Likert scale. The music recommendations were displayed as 30 second song previews using Spotify Play Button widgets.² The study was hosted as an online web-app which collected participant LastFM data and generated recommendations while participants completed the surveys.

We completed a pilot study with participants recruited at our institution in order to test the system before completing the primary study. The pilot study included a post-interview on their experience with the system. No significant concerns were discovered during the pilot study. Primary study participants were recruited through Amazon Mechanical Turk, whose terms of service prohibit asking workers (participants) to register for a service, or log into an existing service. We therefore required workers to have a LastFM account in order to participate, and specified such in the HIT description, the HIT layout, and as a question on the consent form. After obtaining informed consent, we had workers provide their LastFM username which we used to obtain their public listening history. We also required that the LastFM account had at least 50 LEs recorded in the last 6 months. To verify ownership without requiring a login, workers were given 3 attempts to name one artist they had listened to in the previous 6 months. Pilot participants were compensated \$10CAD, and primary participants were compensated \$4USD.

Participants were shown recommendation lists using a balanced Latin square design to control for differences in recommendation list order.

4. RESULTS

4.1 Data and Demographics

We recruited 9 pilot participants, and 97 primary participants. Only primary participant data was used for analysis.

Five participants were removed for completing lists too quickly, resulting in a final participant count of 92. The proceeding results include only these 92 participants. Completion times for *vae* and *als* were observably lower than for diversified lists.

The median participant age was 29, the youngest was 19 and the oldest 62; 50 identified as male (54%), 40 identified as female (43%), and 2 identified as non-binary (2%).

Only tracks in our base training data set can be used to generate recommendations and be recommended. The median count of LEs per participant was 857 before removing tracks not in the base data set, and 627 after. This is compared to a median value of 3110 for users in the base data set.

4.2 Pre-Interaction Survey

In addition to demography, the pre-interaction survey asked questions focused on music consumption, discovery, and music recommendation preferences.

Participants agreed that their diversity preference depended on who makes the recommendations, the quality

of the recommendations, what they are doing, and their mood. Almost 50% of participants disagreed or strongly disagreed that their location was important to how they felt at a given time about music diversity.

We also asked yes/no questions about diversity preference—most participants selected "yes" for all with the notable exception of the question: "Do you want music recommendations outside of genres you like?", for which 32% indicated they were unsure, and 15% responded "no". This question also serves as a parallel to the ideas of *inner* and *outer* diversity preference.

4.3 Recommendation List Evaluation

We assign labels to the recommendation list evaluation questions based on the order in which they were presented to participants. These questions, their labels, and their responses can be seen in Figure 1.

4.4 List Comparisons

We performed a Friedman test on the distributions of responses between each list for all LQ and found that at least one list type's distribution differed significantly for each LQ ($p < 0.001$ for all). A post-hoc Nemenyi test is performed to identify which list's distributions differ from each other. The results of the post-hoc tests are visualised in Figure 1 as black bars connecting significantly different distributions. Note that statistical significance is found more readily among LQ0 because there are 10 samples per list, and we used Dunn's tests instead of Nemenyi test³.

In the responses to rating questions (LQ0) and satisfaction questions (LQ3, LQ4), the control *als* recommendations were consistently rated more positively than the *vae* recommendations (LQ0: $p \leq 0.001$, LQ4: $p = 0.005$). The *als_max_div* recommendations were also consistently rated higher than *vae_max_div* (LQ0: $p \leq 0.001$, LQ3: $p = 0.007$, LQ4: $p \leq 0.001$). Regarding the filtered lists, *als_filt_div* and *vae_filt_div* were rated similarly or better than their un-diversified counterparts in list satisfaction (LQ3, LQ4) despite receiving less positive individual track ratings (LQ0). Filtered lists also performed similar to or better than their maximally diverse counterparts in all cases.

We also examined the distributions of responses to LQ0 and LQ3 by list type using a Kruskal Wallis test, and found significant differences among the control lists ($p \leq 0.001$), which highlights a clear distinction between track ratings and overall list satisfaction, especially for control lists.

The diversity results (LQ1, LQ2, LQ5, LQ6, LQ7) show that all diversified lists were recognised to be significantly more diverse, and to portray a wider range of genres than than their non-diversified controls ($LQ1, LQ5, LQ7 : p \leq 0.001$). No significant differences in perceived diversity or genre range were found between filtered diverse and maximally diverse lists. Participants did not find any list

² <https://developer.spotify.com/documentation/widgets/generate/play-button/>

³ We perform Kruskal-Wallis and Dunn's (with Bonferroni adjustment) tests for LQ0 instead of Friedman and Nemenyi due to the unbalanced data. While this test is typically used for independent samples, we are unaware of a better non-parametric alternative.

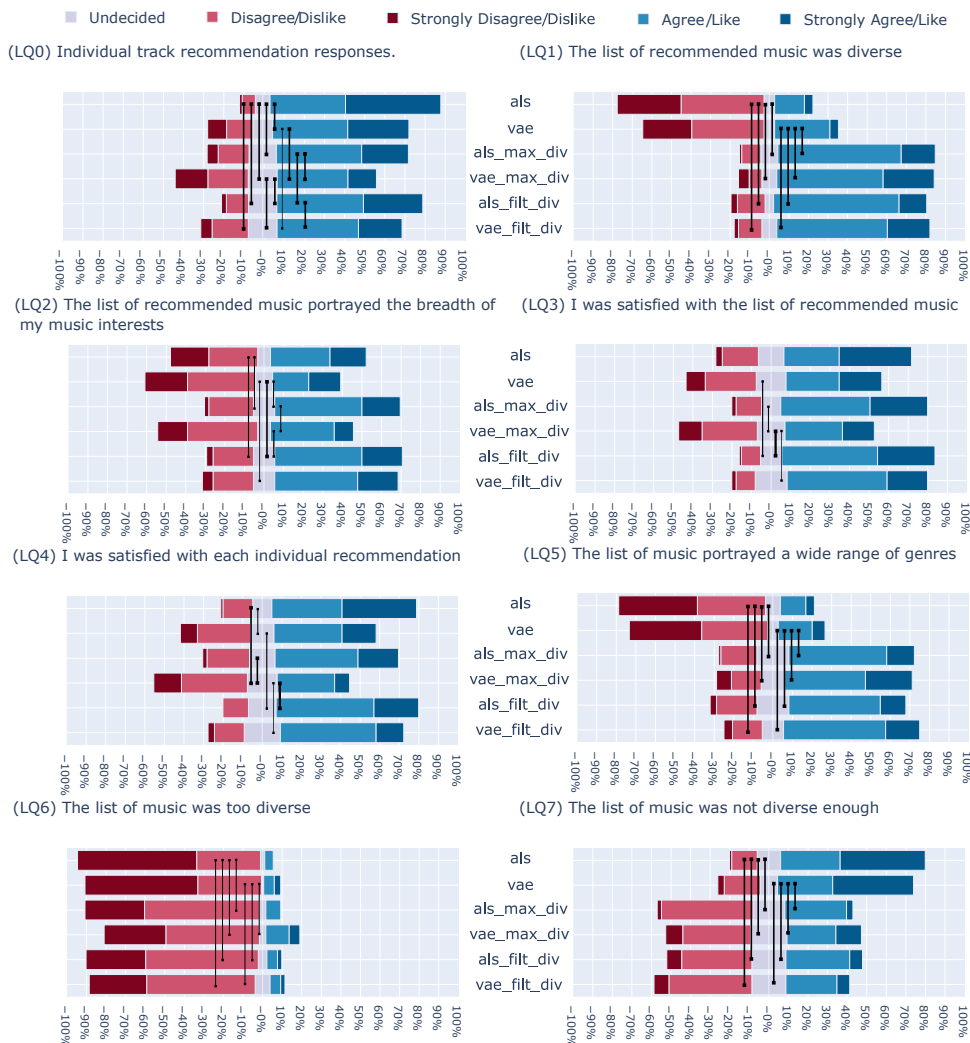


Figure 1: Responses to Likert questions on recommendation lists. Black bars connect significantly different distributions (thick: $p \leq 0.001$, thin: $p \leq 0.05$). LQ0 ratings used Like/Dislike, while all others used Agree/Disagree.

to be overly diverse, though they did feel more strongly that the control lists were not overly diverse as compared to most diversified lists ($LQ6 : p \leq 0.05$). The filtered *als_filt_div* and *vae_filt_div* lists most consistently portrayed the breadth of participants’ music interests (LQ2).

An additional Kruskal-Wallis test was performed on the distributions of responses to LQ1 and LQ5 for all list types with no statistically significant results, supporting the idea that one way users perceive diversity is genre range.

4.5 Summary of Statistically Significant Results

We found statistically significant differences among recommendation algorithms and list generation approaches. In general, recommendations from the ALS model were more satisfying than those from the VAE model, and filtered lists performed similar to or better than control, and maximally diversified lists from the same model. List satisfaction and individual track ratings also differed significantly.

In analysing diversity responses we found that all diverse recommendation lists were recognised as such, and filtered lists were found to be just as diverse as maximally

diversified ones. Filtered lists also most consistently conveyed the breadth of participants interests. Additionally, participant responses on genre range mirrored their evaluations of diversity, and no list types were found to be too diverse.

In the next section, we explore what our results suggest about how to build good music recommender systems.

5. DISCUSSION

5.1 Satisfaction

In recommender systems research, the quality of a recommendation list is often inferred from some accuracy measures computed on known data sets [4]. Our results show that accuracy does **not** tell the whole story. MultVAE outperformed ALS on the base data set using NDCG@100 (Table 1), but participant responses on individual recommendations showed markedly higher satisfaction from the ALS model with an ostensibly lower test accuracy. This gap in satisfaction only grows larger for the maximally diverse recommendations generated from top-1000 lists. We plan to explore this dichotomy through additional and

more exhaustive offline evaluation in another manuscript.

5.2 Diversity

Previous research has identified that mood and context are important factors in determining how much diversity a user wants [16], and this is further supported by user responses on diversity preference which showed that users identify mood and context (among other factors) as important. The difference in responses between ‘what I am doing’ vs. ‘where I am’ emphasise the difference between context and location. A user may be working out if they are detected at a gym, but the location alone is not as significant as the action.

Previous work on optimising diversity levels in recommendation lists has also depended heavily on accuracy measurements [1, 2, 4]. Our results suggest that the difference between control and diversified lists is not well portrayed in individual recommendation ratings. Although maximally diverse lists did result in lower individual track ratings (LQ0), there was no detectable impact on overall list satisfaction (LQ3). Despite strong statistical evidence that both filtered and maximally diversified lists were significantly more diverse. It is especially important to keep in mind that the maximally diverse lists were created using a beta value of 1 from all top-1000 participant recommendations. Either the additional diversity of the lists made up for a decrease in the quality of each recommendation, or the top-1000 recommendations are all of a relatively high quality.

5.3 Genre and Filtering

The nearly identical responses to questions about list diversity and the range of genres further solidify the relationship between the two concepts [10, 16]. When users are asked to evaluate the diversity of a music recommendation list, genre is clearly one of the primary factors they consider.

Overall, the filtered recommendation lists performed as well or better than the maximally diversified lists for satisfaction while also portraying similar levels of diversity. When maximally diverse recommendations were good (as for *als_max_div*) the filtering had no statistically significant impact on list satisfaction or diversity. Alternatively, when maximally diverse recommendations were poor (as for *vae_max_div*) the filtering had a sizeable positive impact on satisfaction without impacting perceived diversity.

The filtered and maximally diverse lists can be viewed as simple implementations of a system for inner and outer diversity.

5.4 Diversity and Personalization

The ILD method we chose is arguably the simplest such diversity metric. Despite its simplicity, our results add to the existing evidence that increasing ILD is perceived by users as increasing diversity, this time in the domain of music [7, 20]. In fact, the significant negative impact of this diversification method was only observed in the Mult-VAE recommendations, and was removed through genre

filtering.

We extend this one step further by noting that the filtered recommendations were generated with $\beta = 1$. Filtering can result in positive satisfaction even with maximal ILD, suggesting that any and all values of β will present viable recommendation lists for each user. This may simplify the task of selecting an optimal level of diversification based on mood and context.

5.5 Pushing Diversity Further

We were unable to generate recommendation lists which reached outside the bounds of our participant’s diversity preferences. Even maximally diverse recommendations were not seen as too diverse. It is very hard to generate overly diverse recommendations using either model. In an ideal collaborative filtering system, diversity preference would be implicitly considered. Also, some users prefer *outer diversity*: recommendations which differ from their existing listening preferences.

Since hyper-parameter optimization of recommendation models make use of accuracy measurements such as NDCG@ k (Section 3.1.3) which incentivize only recommendations in training users’ hidden listening history; most existing recommender systems, because they so strongly focus on accuracy, are unlikely to make risky recommendations.

In order to generate **truly** diverse music recommendations that match user preference, we first need to understand the extents of their preferences for diversity. The idea of recommending surprising items is typically associated with the related beyond-accuracy metric of serendipity [23]. It is easy to equate user preference for *outer diversity* to a preference for serendipity, but this does not explain why even maximally diverse recommendations were not too diverse. Perhaps by extending beyond the top-1000 most relevant recommendations, we can find more diverse recommendations.

If collaborative filtering algorithms do not generate adequate levels of diversity, then are they really working towards generating better music recommendations for users?

5.6 Summary

Our results highlight the large disconnect between offline and online accuracy and diversity evaluations of music recommender systems. Through a sizeable within-subjects study, we evaluated two collaborative filtering algorithms and found that the offline accuracy—and even the user provided track ratings—were not good indicators of overall list satisfaction.

Diversity continues to be an important topic of discussion in recommender systems. Our genre filter-based diversification approach enabled satisfying and diverse recommendations within users’ existing preferences despite using a simple diversity definition. We found success in modeling diversity based on user ideas of the term, and then asking them to evaluate it. In doing so, we brought to light the limited diversity contained within collaborative filtering recommendation algorithms.

6. REFERENCES

- [1] M. Kaminskas and D. Bridge, “Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems,” *ACM Transactions on Interactive Intelligent Systems (TiIS)*, vol. 7, no. 1, pp. 1–42, 2016.
- [2] M. Kunaver and T. Požrl, “Diversity in recommender systems – A survey,” *Knowledge-Based Systems*, vol. 123, pp. 154–162, 2017.
- [3] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon, “Novelty and diversity in information retrieval evaluation,” in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 659–666.
- [4] A. Gunawardana and G. Shani, “Evaluating recommender systems,” in *Recommender Systems Handbook, Second Edition*. Springer, 2015, pp. 265–308.
- [5] K. Bradley and B. Smyth, “Improving recommendation diversity,” in *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, 2001, pp. 85–94.
- [6] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, “Improving recommendation lists through topic diversification,” in *Proceedings of the World Wide Web Conference, WWW*, 2005, p. 22.
- [7] S. Castagnos, A. Brun, and A. Boyer, “When Diversity Is Needed... But Not Expected!” in *International Conference on Advances in Information Mining and Management*, 2013, pp. 44–50.
- [8] A. L’Huillier, S. Castagnos, and A. Boyer, “Understanding usages by modeling diversity over time,” in *CEUR Workshop Proceedings*, vol. 1181, 2014, pp. 81–86.
- [9] S. Vargas, “New Approaches to Diversity and Novelty in Recommender Systems,” in *FDIA’11: Proceedings of the Fourth BCS-IRSG conference on Future Directions in Information Access*, 2011, pp. 8–13.
- [10] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells, “Coverage, redundancy and size-awareness in genre diversity for recommender systems,” in *RecSys 2014 - Proceedings of the 8th ACM Conference on Recommender Systems*, 2014, pp. 209–216.
- [11] D. M. Fleder and K. Hosanagar, “Recommender systems and their impact on sales diversity,” in *EC’07 - Proceedings of the Eighth Annual Conference on Electronic Commerce*, 2007, pp. 192–199.
- [12] R. S. Oliveira, C. Nóbrega, L. B. Marinho, and N. Andrade, “A multiobjective music recommendation approach for aspect-based diversification,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 414–420.
- [13] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, “Algorithmic Effects on the Diversity of Consumption on Spotify,” in *Proceedings of the World Wide Web Conference, WWW*, apr 2020, pp. 2155–2165.
- [14] D. Holtz, B. Carterette, P. Chandar, Z. Nazari, H. Cramer, and S. Aral, “The Engagement-Diversity Connection: Evidence from a Field Experiment on Spotify,” in *EC 2020 - Proceedings of the 21st ACM Conference on Economics and Computation*, jul 2020, pp. 75–76.
- [15] C. Hansen, R. Mehrotra, C. Hansen, B. Brost, L. Maystre, and M. Lalmas, “Shifting Consumption towards Diverse Content on Music Streaming Platforms,” in *WSDM 2021 - Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, aug 2021, pp. 238–246.
- [16] K. Robinson, D. G. Brown, and M. Schedl, “User insights on diversity in music recommendation lists,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 446–453.
- [17] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, “Variational autoencoders for collaborative filtering,” in *Proceedings of the World Wide Web Conference, WWW*, 2018, pp. 689–698.
- [18] Y. Hu, C. Volinsky, and Y. Koren, “Collaborative filtering for implicit feedback datasets,” *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 263–272, 2008.
- [19] B. Frederickson, “Fast Python Collaborative Filtering for Implicit Datasets.” <https://github.com/benfred/implicit>, 2019.
- [20] B. Ferwerda, M. Graus, A. Vall, M. Tkalčič, and M. Schedl, “The Influence of Users’ Personality Traits on Satisfaction and Attractiveness of Diversified Recommendation Lists,” in *Proceedings of the 4th Workshop on Emotions and Personality in Personalized Services (EMPIRE 2016)*, 2016, pp. 43–47.
- [21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Incremental singular value decomposition algorithms for highly scalable recommender systems,” in *Fifth International Conference on Computer and Information Science*, vol. 1, no. 012002, 2002, pp. 27–8.
- [22] M. F. Dacrema, P. Cremonesi, and D. Jannach, “Are we really making much progress? A worrying analysis of recent neural recommendation approaches,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 101–109.
- [23] P. Castells, N. J. Hurley, and S. Vargas, “Novelty and diversity in recommender systems,” in *Recommender Systems Handbook, Second Edition*. Springer, 2015, pp. 881–918.

A FORMAL MODEL OF EXTENDED TONAL HARMONY

Martin Rohrmeier

Digital and Cognitive Musicology Lab, EPFL

`martin.rohrmeier@epfl.ch`

Fabian C. Moss

Digital and Cognitive Musicology Lab, EPFL

`fabian.moss@epfl.ch`

ABSTRACT

Extended tonality is a central system that characterizes the music from the 19th up to the 21st century, including styles like popular music, film music or Jazz. Developing from classical major-minor tonality, the harmonic language of extended tonality forms its own set of rules and regularities, which are a result of the freer combinatoriality of chords within phrases, non-standard chord forms, the emancipation of dissonance, and the loosening of the concept of key. These phenomena posit a challenge for formal, mathematical theory building. The theoretical model proposed in this paper proceeds from Neo-Riemannian and Tonfeld theory, a systematic but informal music-theoretical framework for extended tonality. Our model brings together three fundamental components: the underlying algebraic structure of the Tonnetz, the three basic analytical categories from Tonfeld theory (octatonic and hexatonic collections as well as stacks of fifths), and harmonic syntax in terms of formal language theory. The proposed model is specified to a level of detail that lends itself for implementation and empirical investigation.

1. INTRODUCTION

Harmony is a central latent structure governing Western music since centuries until today [1]. While a considerable amount of research has focused on theoretical, mathematical, and computational exploration of harmony in common-practice major-minor tonality, comparably less attention has been devoted to the challenges that come with the paradigm shift of extended tonality, as it is in place since the 19th century up until the present day in various styles like Jazz, popular or film music.

Extended tonality exhibits harmonic sequences that defy the logic of common-practice (major-minor) tonality. In this paper, we address the problem of accounting for such phenomena with a grammar approach that bridges formal language theory and (mathematical) music theory.

1.1 The Challenge of Extended Tonality

Briefly construed, harmonic progressions in pieces in diatonic common-practice tonality involve chords that are

mostly stacks of thirds derived from the seven diatonic degrees of major or minor scales. Larger harmonic structures emerge through modulations between different keys that are usually close to one another on the line of fifths [1–4], thus forming a system that governs the global hierarchy of pieces [5–9].

In contrast, pieces employing extended tonality may rely on a variety of different scales (e.g., pentatonic, hexatonic, and octatonic scales; see Section 2), freely use harmonies that are not necessarily construed by stacking diatonic thirds, and modulate to or immediately combine chords from relatively distant keys [10–13]. For example, late-Romantic pieces often distinguish themselves from earlier diatonic ones by featuring frequent enharmonic exchanges of pitches, resulting in uncommon chord combinations [14–16], and by the frequent usage of symmetrical scales that impede a listener’s orientation towards a unique tonic, possibly resulting in multiple parallel tonal centers [17, 18].

However, extended tonality is not restricted to late 19th-century pieces, but reaches into many more recent styles, in particular in Jazz with its highly chromatic harmonies [19, 20], and film music, such as scores by Korngold or Williams [21–24]. It also plays a role in Rock and Pop [25–27], and minimalist music, such as by Glass, Frahm or Richter. Extended tonality thus describes not a historical time span but rather captures characteristic features of a harmonic language that extends common-practice major-minor tonality with a variety of phenomena reaching from the late 18th century until the present day [28].

1.2 Related Work

1.2.1 The Tonnetz and Neo-Riemannian Theory

A major analytical approach to extended tonality is neo-Riemannian theory (NRT), which models harmonic progressions between pairs of triads or keys through parsimonious voice-leading transformations [16, 29, 30]. For instance, the *relative* transformation R converts C major into A minor, the *parallel* transformation P converts C major into C minor, and the *leading-tone exchange* transforms C major into E minor. All transformations are involutions, i.e. they are self-inverse. Repeated application of (combinations of) NRT transformations leads to patterns on the *Tonnetz* (Section 2.1) that visualize a particular analysis.

Figure 1 shows an excerpt of the *Tonnetz*. Note that the alternation of P and R transformations creates a pattern



on the Tonnetz that covers all pitches from an octatonic scale (shown in blue), the alternation of P and L transformations creates a path on the Tonnetz that contains all members of a hexatonic scale (shown in orange), and combinations of R and L transformations generate a sequence of triads that modulates through all diatonic scales (not shown; the diatonic is encompassed by a horizontal line of six fifths or two adjacent horizontal lines connecting 6 triangles). The green rectangle delineates a stack of fifths (see below). Due to its focus on triadic transformations, and in particular those that form hexatonic or octatonic cycles, NRT analyses are commonly restricted such that the assumed algebraic spaces imposes some inflexibility with regard to the chord form. Further, stacks of fifths (see below) are commonly not addressed in NRT. Also, though some analyses work with reductions and abstractions from the score, there is no formalized theory of harmonic hierarchy in NRT. In further work, mathematical spaces have been extended or generalized [31–33], used in computational models of harmony or tonality [34–38], or explored empirically [39–42].

1.2.2 Tonfeld Theory

Another recent theory addressing the challenges of extended tonality is Tonfeld theory (TFT) [43–49]. Unlike NRT, it does not fundamentally rely on triads or keys. Instead, it departs directly from three so-called *Tonfelder* (‘tone fields’) that correspond to hexatonic, octatonic, and fifths-related (e.g., pentatonic, diatonic) tone collections (see Section 2.2 for details) that are assumed to govern segments of pieces at the granularity of the pitch level. It mostly focuses on late 19th-century compositions but some analyses for 18th and 20th century pieces exist as well [28,50,51]. Moreover, TFT subscribes to a fundamentally hierarchical conception of compositions by analyzing a piece’s tonal coherence through the presence and interactions of *Tonfelder* on several structural levels of abstraction/reduction. This allows in principle for the expression of nested structures and non-local dependencies.

1.2.3 Harmonic Syntax

Syntactic formalisms derive musical (e.g. harmonic or melodic) sequences through generative models that result in trees or similar hierarchical dependency structures [6, 52–57]. They frequently adopt frameworks from formal language theory and adapt them to the particular needs for the case of music. In recent years, several formal and computational approaches have been developed for Western classical music [7, 58, 59], Blues [60, 61], Jazz [20, 62–65]. We build on such previous approaches and expand their scope to extended tonality.

2. THE MODEL

2.1 The Tonnetz

One foundation of the present model is the *Tonnetz*. It goes back to Leonard Euler’s definition of intervals in just intonation [66], leading to an abstract pitch class space [67].

Accordingly, every just interval can be expressed by a frequency ratio:

$$f_1/f_2 = 2^x \cdot 3^y \cdot 5^z, \quad x, y, z \in \mathbb{Z}. \quad (1)$$

Since the factor 2 defines the octave, the two other integer factors y and z span a coordinate system of pitch classes (modulo the octave) that defines an infinite plane of fifths and major thirds. Taking into account that a fifth is composed of a major and a minor third, the plane corresponds to a triangular graph such that there are three main axes of major, minor thirds and fifths, in which each triangle defines a major or minor triad (see Figure 1).

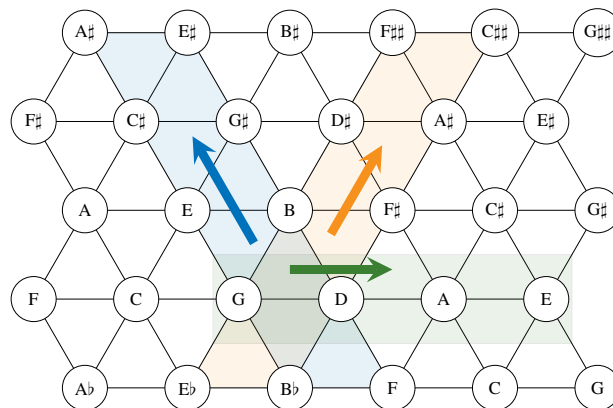


Figure 1: The Tonnetz and the construction of the three types of *Tonfeld* structures. The pitch class set shaded in blue defines one octatonic, the set shaded in orange defines one hexatonic, the set in green defines a stack of fifths.

The Tonnetz defines an *infinite* space of spelled pitch classes none of which are identical, i.e. different nodes in the graph with the same label are indeed distinct. If one identifies nodes with the same labels, the infinite line of fifths wraps itself around the so-called *Spiral Array* [37]. If, further, enharmonic equivalence is assumed (e.g. $D\# \equiv E\flat$), the space becomes a torus [39, 68] and the resulting pitch-class space is isomorphic to \mathbb{Z}_{12} .

2.2 The Tonfelder

Tonfeld theory comprises three fundamental concepts that are expressed in terms of pitch collections: octatonic, hexatonic, and stacks of fifths. This section introduces these building blocks that the theory operates on. All concepts are formulated in the toroidal \mathbb{Z}_{12} pitch space, yet they could be easily generalized to the spelled pitch space and the infinite Tonnetz. At first, a stack of intervals SI is defined by a starting pitch p , and an interval i with k iterations. If the iterations reach the starting pitch, SI defines a cyclic group (simply written as $SI_{p,i}$).¹

$$SI_{p,i,k} := \{p + im \mid m \leq k; k, m \in \mathbb{N}_0\} \quad (2)$$

Interval cycles are musically meaningful units to ground a tonal theory [69, 70]. Here, the three *Tonfelder* are constructed from the three directions in the Tonnetz. The *octatonic* is defined by shifting a fifth along the axis of minor

¹ An analogous definition in the infinite Tonnetz space would not result in a cyclic group.

thirds, the *hexatonic* by shifting a fifth along the axis of major thirds, and the *stack of fifths* by collecting consecutive fifths along the axis of fifths (see Figure 1).

$$Oct_p := \{i, i + 7 \mid i \in SI_{p,3}\} \equiv SI_{p,3} \uplus SI_{p+7,3} \quad (3)$$

$$Hex_p := \{i, i + 7 \mid i \in SI_{p,4}\} \equiv SI_{p,4} \uplus SI_{p+7,4} \quad (4)$$

$$Fif_{p,k} := SI_{p,7,k} \quad (5)$$

Notably, all three Tonfelder are based on the foundational tonal interval of the fifth. This construction results in the three different octatonic (half-tone–whole-tone) scales and the four hexatonic (minor-third–half-tone) scales ($Oct_i = Oct_{i+3}$; $Hex_i = Hex_{i+4}$):

$$Oct_0 = \{C, D\flat, E\flat, E\sharp, G\flat, G\sharp, A, B\flat\} \quad (6)$$

$$Oct_1 = \{D\flat, D, E\sharp, F, G, A\flat, B\flat, C\flat\} \quad (7)$$

$$Oct_2 = \{C, D, E\flat, F, G\flat, A\flat, A\sharp, B\} \quad (8)$$

$$Hex_0 = \{C, E\flat, E, G, A\flat, B\} \quad (9)$$

$$Hex_1 = \{C\sharp, E, F, G\sharp, A, C\} \quad (10)$$

$$Hex_2 = \{D, F, F\sharp, A, B\flat, C\sharp\} \quad (11)$$

$$Hex_3 = \{E\flat, F\sharp, G, B\flat, B\sharp, D\} \quad (12)$$

Since stacks of fifths do not form a mode of limited transposition [71], there are 12 different types of stacks of fifths until the whole chromatic is reached.

$$Fif_{C,2} = \{C, G, D\} \quad (13)$$

$$Fif_{C,3} = \{C, G, D, A\} \quad (14)$$

$$\dots Fif_{C,5} = \{C, G, D, A, E, B\} \quad (15)$$

The set of all Tonfelder \mathbb{T} is defined as $\mathbb{T} := \{Oct_0, Oct_1, Oct_2\} \cup \{Hex_0, \dots, Hex_3\} \cup \{Fif_{p,k} \mid p \in 0, \dots, 11, k \in 2, \dots, 10\}$. One can apply *filters* to a Tonfeld t to arrive at basic musical units, such as triads or tetrads. For instance, all triads in a Tonfeld are filtered out by

$$f : t \mapsto \{\{a, b, c\} \mid iv(\{a, b, c\}) = (0, 0, 1, 1, 1, 0)\} \quad (16)$$

where iv is the interval vector of a given pitch-class set, which counts all possible interval classes [72]. One can easily define filters for other chord types or, in fact, arbitrary pitch-class sets [72, 73].

In terms of common chord types, the octatonic scale yields four major, minor, dominant seventh, minor seventh and half-diminished chords each, all related by minor thirds, and two fully diminished tetrads a fifth/a semitone apart. The hexatonic scale in turn yields three major, minor, major seventh, minor-major-seventh chords a major third apart, and two augmented triads a fifth/a semitone apart. Stacks of fifths yield chords that are often classified as *sus*-chords or quartal voicings in Jazz harmony terminology,² and also cover complex add 6, 9th or 11th chords as they appear in Jazz [74]. Notably, also complex chords

² The stack of fifths chords (e.g. $C - G - D$) are technically not suspension chords since they do commonly not imply a resolution of the dissonant fourth into a third as in standard common-practice. [3]

such as the “Tristan chord”, the “Petrouchka chord”, Scriabin’s “mystical chord” as well as many upper structure chords in Jazz (e.g. $G-F-B-D-E-G\sharp-B$) [75], are captured within the octatonic set. Non-diatonic minor chords with major sevenths as they occur in Jazz are captured by the hexatonic set. The set of all chords derived by suitable filters (for the particular surface to be modeled) from a Tonfeld $t \in \mathbb{T}$ is denoted by C_t . This definition encompasses occurrences of non-standard chord forms, such as the ones above.

Tonfelder further bear generalizing expressive power with respect to central tonal relations. The set of chords with a dominant function may, for instance, involve $V, V^7, vii^0, \flat VII, \flat II^7$ (tritone substitution), or III^7 . Similar, subdomantic/pre-dominant chords may, e.g., involve $IV, ii, II, iv^6, \flat VI$. Both of these sets of equivalences are all encompassed by the set of chords from the two octatonics neighboring the reference tonic chord [44, 70]. Therefore, the octatonic Tonfeld can be understood as a generalization over the concept of tonal harmonic function (tonic, dominant, predominant) as well as intra-functional prolongation/substitution (within the same octatonic).

Conversely, Neo-Riemannian theories (as well as TFT) have identified chords from the hexatonic to establish contrastive relationships, such as the hexatonic pole (e.g., C major – $A\flat$ minor), [14, 76]. Such relations between harmonies are the basis of the hierarchical dependencies that are modelled by the grammar outlined in the next subsection.

2.3 The Grammar

The proposed harmonic grammar formalism is based on abstract context-free grammars (ACFGs) [63] and extends previous models of harmonic syntax [7, 20, 58]. It consists of four components: $\mathcal{G} = (K, \Sigma, P, s)$, non-terminal categories K , terminal symbols Σ , production rules P , and a start symbol $s \in K$. The set of all terminals Σ encompasses all (potentially non-standard) chords derived from the Tonfelder: $\Sigma := \{c \mid c \in C_t \forall t \in \mathbb{T}\}$.

There are three kinds of non-terminal category symbols: $K = \{s\} \cup \{\Theta_t \mid t \in \mathbb{T}\} \cup \{c_t \mid c \in C_t, t \in \mathbb{T}\}$. s denotes the abstract start symbol. Except for the start symbol, non-terminal category symbols have a feature t , which indicates the assigned Tonfeld of the category. Abstract Tonfeld categories $\Theta_t (\in \{\Theta\} \times \mathbb{T})$ define an unspecific Tonfeld $t \in \mathbb{T}$ that has not yet been instantiated in terms of a concrete chord category. Chord categories $c_t (\in \Sigma \times \mathbb{T})$ are defined in terms of any chord symbol c derived from its assigned Tonfeld t .

In ACFGs, the production rules P are defined as functions mapping the left-hand side to the right-hand side. Here, P involves three kinds of rules: general rules (*start, instantiation, Tonfeld cast, termination*), rules characterizing hierarchical functional relations (*prolongation, substitution, preparation, plagal dependency, contrast*), and rules with set operations for manipulating stacks of fifths (*fifth shift, fifth expansion & contraction, fifth split*). Notably, it is not necessary to formally assume substitution

because substitutable equivalences are already expressed at the level of the octatonic Tonfeld. The following paragraphs define each rule type:

2.3.1 General Rules

Start. A piece is modeled as a sequence of different Tonfeld categories Θ .

$$s \longrightarrow \Theta_{t_1}^{(1)} \dots \Theta_{t_n}^{(n)} \quad (17)$$

Note that in contrast to previous (diatonic) syntax theories [7], there is no requirement of an overarching single tonic node of a derivation tree (although this ‘downward compatability’ can be achieved by a single overarching octatonic chord category). This comes from the different logic of Tonfeld structures [28,44]. Thus, the top level may consist in a sequence of different Tonfelder. For abridged derivations or analyses of partial sequences, the trees can also directly be headed by a single Tonfeld or chord category, omitting the start symbol (see the examples below).

Tonfeld instantiation. An abstract Tonfeld symbol Θ_t of the Tonfeld t may be instantiated with one or more member chords from its set.

$$\Theta_t \longrightarrow Y_t^{(1)} \dots Y_t^{(n)} \quad (18)$$

Tonfeld casting. Generalizing modulations, a chord category can change its underlying Tonfeld and recursively yield different generations. Importantly, this operation can only be performed over chord categories, since the abstract Tonfeld categories are ambiguous with regard to their chord instantiation, and therefore their cast to a different Tonfeld is not well-defined. An abstract Tonfeld category can only be cast into another through *instantiation* in terms of a pivot chord category. Therefore, Tonfeld casting necessarily involves the set intersection of two Tonfelder (which in turn enforces the chord type category for X).

$$X_m \longrightarrow X_k, \quad X \in C_m \cap C_k \quad (19)$$

Terminal rules. The grammar needs to ensure that the sequence generation terminates. The production can terminate when there are no more abstract categories in the sequence. Since chord categories are already absolute chords matching surface chord forms, the only final step is to cast the chord category into a terminal chord without a Tonfeld feature: $X_t \longrightarrow X$. For stacks of fifths, the resulting chord forms may be non-standard, e.g. non-triadic (see Figure 6).

2.3.2 Rules for functional relations

Prolongation. Going beyond previous models [7,20], prolongation need not only combine identical categories but may combine elements of the same Tonfeld. Following the generalization by Steedman, prolongation can be understood as an instance of syntactic coordination [62,77]. Prolongation can be established with two different types, abstract Tonfeld categories and chord categories, and may combine two or more categories.

$$\Theta_t \rightarrow \Theta_t \dots \Theta_t \quad (20)$$

$$X_t \rightarrow Y_t^{(1)} \dots Y_t^{(n)}, \quad X_t = \Theta_t \vee \exists i : Y^{(i)} = X \quad (21)$$

(Substitution.) The octatonic moreover generalizes over possible substitutions of certain sets of chords. It is useful to formulate this as a rule even though, in most cases, the direct derivation of dominant or subdominant substitutions may be achieved directly though the preparation- and plagal-dependency rules (and therefore, the rule may not be necessary in computational implementations).

$$X_{Oct_a} \longrightarrow Y_{Oct_a}, X \neq Y \quad (22)$$

Preparation. The octatonic generalizes over the class of chords that may prepare other chords [20]. Since there are only three different octatonics (3) there are only two possible preparations (motions between them): preparation and plagal dependency (see below). A preparation is derived as the preceding left child of the prepared chord. The types of X and Y can be an abstract octatonic Tonfeld Θ_{Oct_a} or a chord category c_t .

$$X_{Oct_a} \longrightarrow Y_{Oct_{a+1}} X_{Oct_a} \quad (23)$$

Plagal dependency. The octatonic also generalizes over the set of subdominants. A plagal dependency/relaxation into a chord is modeled as its left child. Although the plagal relaxation has a similar form as the preparation rule, its semantics is different. This implies that the semantic of the applied dependency type (preparation or plagal) cannot be inferred from its shape in the tree (i.e. a left child is not necessarily a dominant – pace GTTM [6]). Similarly to the preparation rule, the types of X and Y can either be both abstract octatonic Tonfelder or chord categories.

$$X_{Oct_a} \longrightarrow Y_{Oct_{a-1}} X_{Oct_a} \quad (24)$$

Contrast. The hexatonic and the stack of fifths allow for the instantiation of the relation of a contrast to a given element. For the hexatonic, contrast is established *within* the same hexatonic. The most frequent form of the hexatonic contrast is the *hexatonic pole* [78]. For the stack of fifths, contrast is established between two quasi-complementary stacks of fifths (*fifth flipover*).

$$X_{Hex_i} \longrightarrow Y_{Hex_i} X_{Hex_i}, X \neq Y \quad (25)$$

$$X_{Hex_i} \longrightarrow X_{Hex_i} Y_{Hex_i}, X \neq Y \quad (26)$$

$$\Theta_{Fif_{a,b}} \longrightarrow \Theta_{Fif_{c,d}} \Theta_{Fif_{a,b}}, \quad \text{if } inv(Fif_{a,b}, Fif_{c,d}) \quad (27)$$

$$\Theta_{Fif_{a,b}} \longrightarrow \Theta_{Fif_{a,b}} \Theta_{Fif_{c,d}}, \quad \text{if } inv(Fif_{a,b}, Fif_{c,d}) \quad (28)$$

The Boolean *inverse* relation *inv* between two stacks of fifths $Fif_{p,k}$ and $Fif_{q,l}$ is fulfilled if they are sufficiently distinct for some distance function d and threshold δ (29). Since the stacks are sets, a suitable candidate is the Jaccard distance (30).

$$inv(Fif_{p,k}, Fif_{q,l}) := d(Fif_{p,k}, Fif_{q,l}) \leq \delta \quad (29)$$

$$d(A, B) := \frac{|A \cap B|}{|A \cup B|} \quad (30)$$

2.3.3 Rules for Stacks of Fifths

Fifth shift. A stack of fifths can be shifted one step in either direction of the circle of fifths. The rule can instantiate

an instance before or after the parent category.

$$\Theta_{Fif_{p,k}} \rightarrow \Theta_{Fif_{p,k}} \Theta_{Fif_{p\pm 7,k}} \quad (31)$$

$$\Theta_{Fif_{p,k}} \rightarrow \Theta_{Fif_{p\pm 7,k}} \Theta_{Fif_{p,k}} \quad (32)$$

Fifth expansion and contraction. A stack of fifths can be extended or reduced by a number of fifths. This results in a stack of fifths with a different number m of fifths, respecting the defining criterion of a stack of fifths of $2 < m < 11$. Fifth expansion and contraction may propagate to the left or the right within the sequence.

$$\Theta_{Fif_{p,k}} \rightarrow \Theta_{Fif_{p,k}} \Theta_{Fif_{p,m}} \quad , m \neq k \quad (33)$$

$$\Theta_{Fif_{p,k}} \rightarrow \Theta_{Fif_{p,m}} \Theta_{Fif_{p,k}} \quad , m \neq k \quad (34)$$

Fifth split. A stack of fifths can be split into two different, potentially overlapping, stacks of fifths.

$$\Theta_{Fif_{p,k}} \rightarrow \Theta_{Fif_{q,l}} \Theta_{Fif_{r,m}} \quad , Fif_{q,l} \cup Fif_{r,m} = Fif_{p,k} \quad (35)$$

3. EXAMPLES

Figure 2 illustrates how the use of the octatonic generalizes over remote variants of authentic preparation progressions or cadences, without requiring modulation, borrowing, tritone substitutions, or chromatic operations.

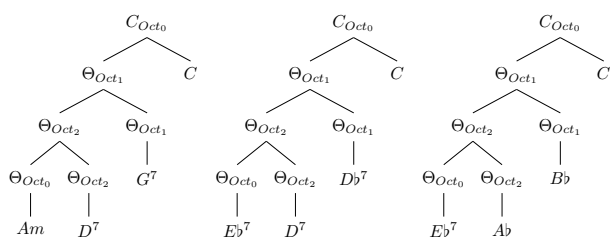


Figure 2: Generalizing over 3 preparatory progressions.

One example that well illustrates the octatonic set is given by two phrases from the second movement of Antonin Dvořák’s ninth Symphony (Figure 3). It illustrates (a) that the chords used in sequence defy a purely diatonic (e.g. $D\flat$ major) analysis, thus requiring a different analytical framework, and (b) that the two excerpts sound very similar even though they use different and remote chords at the surface. Our analysis illuminates that the chords stem from two octatonic Tonfelder which are identical for both examples. The very similar impression of both excerpts is modeled by the identical deep structure of the tree derivations. Further octatonic examples include Schubert’s *Ganymed* D544 and Scriabin’s *Prelude* op. 74/2.

One paradigmatic example for hexatonic Tonfelder is John Coltrane’s piece “Giant Steps” (Figure 5). Similarly to the previous example, the chord sequence here also defies diatonic derivations because of the overarching major-third relations of the harmonic centers B major, G major and $E\flat$ major. Notably, the piece abandons the sense of an overarching key, oscillating between the harmonic centers establishing an overarching *abstract* hexatonic Tonfeld instead. The tree analysis demonstrates that the chord sequence is simple to derive once hexatonic relations are assumed at the top level. The local $ii-V-I$ progressions are

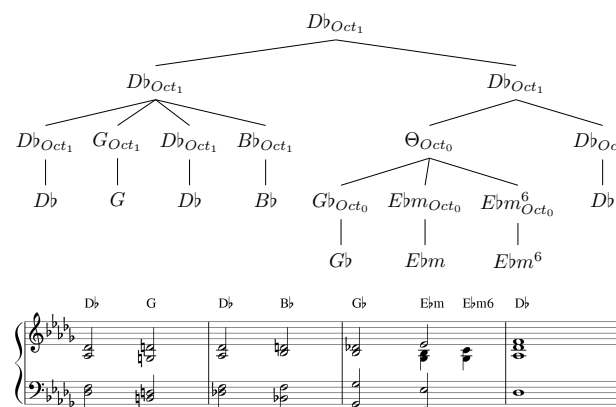
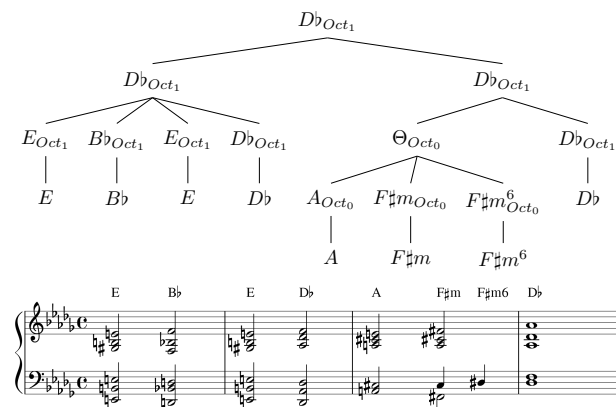


Figure 3: Dvořák, Symphony IX, op. 95–II, mm. 22-25, mm. 120-123

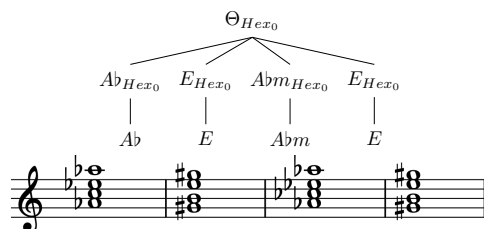


Figure 4: Chord progression for Aragorn and Arwen’s love scene from *Lord of the Rings* (Howard Shore).

well-modeled with octatonic preparatory relations. Thus, this example also illustrates a case of hierarchical intertwining of two different types of Tonfelder.

Another example for a hexatonic progression is taken from Howard Shore’s score to the film *Lord of the Rings* for the scene in which the characters Aragorn and Arwen, a couple that embodies contrast (human/mortal vs. elf/immortal), engage in intimate conversation. Underlying this scene is a loop of the chord progression $Ab - E - Abm - E$. Similar to the octatonic example above, these chords can not be subsumed under a single key and an analysis where each chord change entails a key modulation seems implausible. Rather, these chords are all taken from the hexatonic Tonfeld Hex_0 , as shown by the analysis in Figure 4. Not all triads possible in this Tonfeld do occur but the sequence expresses all pitch classes from Hex_0 , except G . Further hexatonic examples can be found

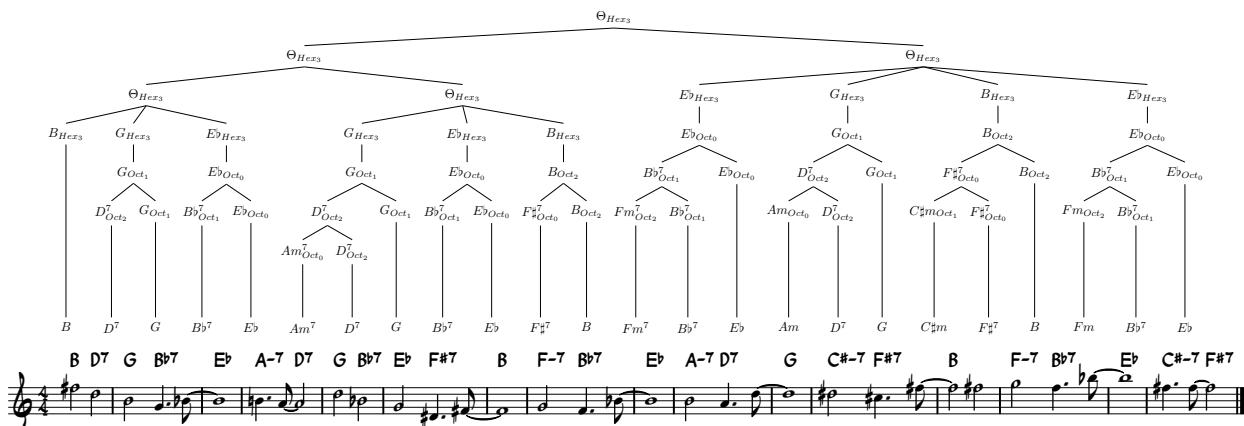


Figure 5: John Coltrane’s “Giant Steps”. The three harmonic centers $B, G, E\flat$ span a complete hexatonic Hex_3 .

in the prelude of Wagner’s *Parsifal* and Bruckner’s *Ecce Sacerdos Magnus* WAB 13. As shown in [25], octatonic and hexatonic structures occur frequently in Popular music, e.g. *Shake the Disease* by Depeche Mode, *Easy Meat* by Frank Zappa, *Creep* by Radiohead, or *Lay, Lady, Lay* by Bob Dylan.

“Maiden Voyage” by Herbie Hancock provides a good illustration for the use of stacks of fifths (Figure 6). First, none of the *sus* chords in the leadsheet are proper *suspension* chords because they do not imply a resolution to an omitted harmonic interval. In fact, they are implicit notations for quartal voicings, which are in fact stacks of fifths. Both chord pairs in both sections constitute a *split* of an overarching stack of fifths. The relation between both parts is that the overarching *prolonged* stack of fifths is contrasted by *fifth flipover*. Other examples for this Tonfeld are Bartók’s *Boating* from Mikrokosmos V, Tailleferre’s *Pastorale*, Ligeti’s piano etude no. 8, Kraftwerk’s *Trans Europa Express*, Zimmer’s film music to *Interstellar*.

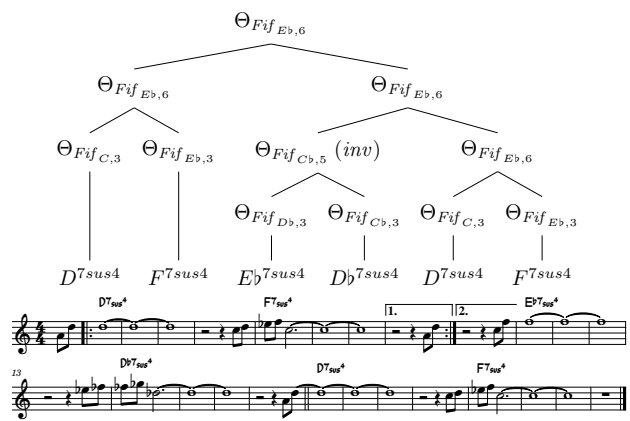


Figure 6: Analyzing Herbie Hancock’s “Maiden Voyage” illustrating operations on stacks of fifths.

4. CONCLUSION

We present a model of extended tonality bridging music theoretical accounts and formal grammars. It captures chord forms and chord sequences that are challenging for

other (diatonic) approaches. While some details, e.g. cases not yet covered, may be subject to debate, we argue that the core innovation and main benefit lies in providing a well-formalized hierarchical model of extended tonal harmony. As an aside, our model offers an explanation why some previous computational models find octatonic and hexatonic structures as efficient structures in their latent space [40, 42].

Our model is based on three Tonfelder as construed from the Tonnetz and the foundational interval of the perfect fifth, both of which we consider fundamental for extended tonality. We argue that the collections that can be constructed from the Tonnetz have a special status in establishing (extended) tonality compared with the manifold other scales. For instance, the whole-tone scale cannot be directly represented on the Tonnetz, and we argue that it can thus not form a deep structure, despite appearing on the surface. Our model constitutes an extension of formal grammars for diatonic music, meaning that it can also generate purely diatonic sequences, in analogy to extended tonal compositions containing also purely tonal sections.

The aim of the theory is to not only capture musical surface events but to model the kinds of underlying dependencies with theoretically meaningful concepts and assumptions, i.e. *strong generativity* [79]. Similarly to previous syntactic theories of music, the latent analytic derivations link *structure* and *interpretation* in terms of dependencies and chord functions (e.g. preparation, contrast) [20, 80, 81]. Because of the expressive richness of the model multiple concurrent analyses are possible for a given sequence. This makes it possible to express diverging hearings and nuances of a passage that different listeners may experience. The theory will not suffice as a forward generative model for computational composition on its own. It would require additional (inferable) style-specific parameters, since extended harmony works differently in Dvořák, Ravel, the Beatles, Coltrane, or Richter, and may as well benefit from a joint model of rhythm [20, 82]. The theory is sufficiently well-specified such that it is testable, debatable, and lends itself for empirical investigation in a probabilistic formulation in future work.

5. ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (GA No 760081), and from the Swiss National Science Foundation (GA No 182811). We thank Claude Latour for supporting this research through the Latour Chair in Digital Musicology. We thank Christoph Finkensiep, Gabriele Cecchetti and Markus Neuwirth for many valuable discussions.

6. REFERENCES

- [1] C. Dahlhaus, J. Anderson, C. Wilson, R. Cohn, and B. Hyer, "Harmony," in *The New Grove Dictionary of Music and Musicians*, 2nd ed., S. Sadie and J. Tyrrell, Eds. London: Macmillan Publishers, 2001, pp. 858–877.
- [2] B. Hyer, "Tonality," in *The New Grove Dictionary of Music and Musicians*, 2nd ed., S. Sadie and J. Tyrrell, Eds. London: Macmillan Publishers, 2001, pp. 583–594.
- [3] E. Aldwell, C. Schachter, and A. Cadwallader, *Harmony and Voice Leading*, 4th ed. Cengage Learning, 2010.
- [4] D. Temperley, "The Line of Fifths," *Music Analysis*, vol. 19, no. 3, pp. 289–319, Oct. 2000.
- [5] H. Schenker, *Der Freie Satz*. Wien: Universal Edition, 1935.
- [6] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music*. Cambridge, MA: MIT Press, 1983.
- [7] M. Rohrmeier, "Towards a generative syntax of tonal harmony," *Journal of Mathematics and Music*, vol. 5, no. 1, pp. 35–53, Mar. 2011.
- [8] R. Lieck and M. Rohrmeier, "Modelling Hierarchical Key Structure with Pitch Scapes," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR 2020)*, Montreal, Canada, 2020, pp. 811–818.
- [9] C. Viaccoz, D. Harasim, F. C. Moss, and M. Rohrmeier, "Wavescapes: A Visual Hierarchical Analysis of Tonality using the Discrete Fourier Transformation," *Musicae Scientiae*, accepted.
- [10] A. Schoenberg, *Structural Functions of Harmony*, L. Stein, Ed. Faber and Faber, 1969.
- [11] F. Lerdahl, *Tonal Pitch Space*. Oxford: Oxford University Press, 2001.
- [12] D. Harrison, *Harmonic Function in Chromatic Music: A Renewed Dualist Theory and an Account of Its Precedents*. Chicago and London: University of Chicago Press, 1994.
- [13] A. T. Katz, *Challenge to Musical Tradition - A New Concept of Tonality*. New York: Alfred Knopf, 1945.
- [14] R. Cohn, *Audacious Euphony: Chromatic Harmony and the Triad's Second Nature*. Oxford: Oxford University Press, 2012.
- [15] D. Harrison, "Nonconformist Notions of Nineteenth-Century Enharmonicism," *Music Analysis*, vol. 21, no. 2, pp. 115–160, 2002.
- [16] S. Rings, *Tonality and Transformation*. New York: Oxford University Press, 2011.
- [17] J. Horton, "Form and Orbital Tonality in the Finale of Bruckner's Seventh Symphony," *Music Analysis*, vol. 37, no. 3, pp. 271–309, Oct. 2018.
- [18] F. C. Moss, "Tonality and functional equivalence: A multi-level model for the cognition of triadic progressions in 19th century music," in *International Conference of Students of Systematic Musicology - Proceedings*, vol. 1, London, 2014, pp. 1–8.
- [19] D. Terefenko, *Jazz Theory: From Basic to Advanced Study*. New York: Routledge, 2014.
- [20] M. Rohrmeier, "The Syntax of Jazz Harmony: Diatonic Tonality, Phrase Structure, and Form," *Music Theory and Analysis (MTA)*, vol. 7, no. 1, pp. 1–63, Apr. 2020.
- [21] F. M. Lehman, "Schubert's SLIDEs: Tonal (Non-) Integration of a Paradoxical Transformation," *Music Theory & Analysis*, vol. 1, no. I & II, pp. 61–100, 2014.
- [22] E. Heine, "Chromatic Mediants and Narrative Context in Film," *Music Analysis*, vol. 37, no. 1, pp. 103–132, 2018.
- [23] F. Lehman, *Hollywood Harmony: Musical Wonder and the Sound of Cinema*. Oxford: Oxford University Press, 2018.
- [24] S. Murphy, "Scoring Loss in Some Recent Popular Film and Television," *Music Theory Spectrum*, vol. 36, no. 2, pp. 295–314, 2014.
- [25] G. Capuzzo, "Neo-Riemannian Theory and the Analysis of Pop-Rock Music," *Music Theory Spectrum*, vol. 26, no. 2, pp. 177–200, 2004.
- [26] D. Temperley, *The Musical Language of Rock*. Oxford: Oxford University Press, Feb. 2018.
- [27] N. Biamonte, "Triadic Modal and Pentatonic Patterns in Rock Music," *Music Theory Spectrum*, vol. 32, no. 2, pp. 95–110, Oct. 2010.
- [28] B. Haas, "Zur Sonatenform II mit analytischen Bemerkungen zum ersten Satz von Mozarts Sonate KV 570," in *Funktionale Analyse: Music - Malerei - Antike Literatur*, B. Haas and B. Haas, Eds. Hildesheim, Zürich, New York: Olms, 2010, pp. 261–291.

- [29] R. Cohn, "Neo-riemannian operations, parsimonious triads, and their "Tonnetz" representations," *Journal of Music Theory*, vol. 41, no. 1, pp. 1–66, 1997.
- [30] —, "Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective," *Journal of Music Theory*, vol. 42, no. 2, pp. 167–180, 1998.
- [31] D. Tymoczko, *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. Oxford: Oxford University Press, 2011.
- [32] D. Harasim, S. E. Schmidt, and M. Rohrmeier, "Bridging scale theory and geometrical approaches to harmony: The voice-leading duality between complementary chords," *Journal of Mathematics and Music*, vol. 10, no. 3, pp. 193–209, 2016.
- [33] D. Harasim, T. Noll, and M. Rohrmeier, "Distant Neighbors and Interscalar Contiguities," in *Mathematics and Computation in Music*, ser. Lecture Notes in Computer Science, M. Montiel, F. Gomez-Martin, and O. A. Agustín-Aquino, Eds. Cham: Springer International Publishing, 2019, pp. 172–184.
- [34] J. Bragg, E. Chew, and S. Shieber, "Neo-riemannian cycle detection with weighted finite-state transducers," in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 399–404.
- [35] R. Lieck, F. C. Moss, and M. Rohrmeier, "The Tonal Diffusion Model," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 153–164, 2020.
- [36] A. J. Milne and S. Holland, "Empirically testing Tonnetz, voice-leading, and spectral models of perceived triadic distance," *Journal of Mathematics and Music*, vol. 9737, no. April, pp. 1–27, 2016.
- [37] E. Chew, "Towards a mathematical model of tonality," Doctoral Dissertation, Massachusetts Institute of Technology, 2000.
- [38] M. Navarro-Cáceres, M. Caetano, G. Bernardes, M. Sánchez-Barba, and J. Merchán Sánchez-Jara, "A Computational Model of Tonal Tension Profile of Chord Progressions in the Tonal Interval Space," *Entropy*, vol. 22, no. 11, p. 1291, Nov. 2020.
- [39] C. L. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys," *Psychological Review*, vol. 89, no. 4, pp. 334–368, 1982.
- [40] F. Korzeniowski and G. Widmer, "Improved Chord Recognition by Combining Duration and Harmonic Language Models," in *Proceedings of the 19th ISMIR Conference*, Paris, France, 2018, pp. 10–17.
- [41] P. Janata, J. Birk, J. V. Horn, M. Leman, J. D. Van Horn, M. Leman, B. Tillmann, and J. J. Bharucha, "The cortical topography of tonal structures underlying Western music," *Science*, vol. 298, no. 5601, pp. 2167–2170, Dec. 2002.
- [42] G. Bernardes, D. Cocharro, M. Caetano, C. Guedes, and M. E. Davies, "A multi-level tonal interval space for modelling pitch relatedness and musical consonance," *Journal of New Music Research*, vol. 8215, no. July, pp. 1–14, 2016.
- [43] A. Simon, "Béla Bartók: ›Secondes mineures – septièmes majeures‹ (Mikrokosmos, VI/144)," *Schweizerische Musikzeitung - Revue musicale suisse*, vol. 123, pp. 82–86, 1983.
- [44] B. Haas, *Die Neue Tonalität von Schubert Bis Webern: Hören Und Analysieren Nach Albert Simon*. Wilhelmshaven: Florian Noetzel, 2004.
- [45] M. Polth, "The Individual Tone and Musical Context in Albert Simon's Tonfeldtheorie," *Music Theory Online*, vol. 24, no. 4, 2018.
- [46] —, "›Tonalität der Tonfelder. Anmerkungen zu Bernhard Haas, Die neue Tonalität von Schubert bis Webern. Hören und Analysieren nach Albert Simon, Wilhelmshaven: Noetzel 2004‹," *Zeitschrift der Gesellschaft für Musiktheorie*, vol. 3, no. 1, pp. 167–178, 2006.
- [47] J. Schild, "'... zum Raum wird hier die Zeit.' Tonfelder in Wagners Parsifal," in *Funktionale Analyse: Musik - Malerei - Antike Literatur*, B. Haas and B. Haas, Eds. Hildesheim, Zürich, New York: Georg Olms Verlag, 2010, pp. 313–373.
- [48] D. Schiltknecht, "›Konstrukt‹ und ›Funktion‹ – Eine Herleitung der Simonschen Tonfelder," *2Zeitschrift der Gesellschaft für Musiktheorie*, vol. 8, no. 2, pp. 351–363, 2011.
- [49] F. C. Moss, "'Theorie der Tonfelder' nach Simon und 'Neo-Riemannian Theory': Systematik, historische Bezüge und analytische Praxis im Vergleich," MA Thesis, Hochschule für Musik und Tanz Köln, Nov. 2012.
- [50] M. Polth, "Zur Artikulation von Tonfeldern bei Brahms, Debussy und Stockhausen," *Zeitschrift der Gesellschaft für Musiktheorie [Journal of the German-speaking Society of Music Theory]*, vol. 8, no. 2, pp. 225–265, 2011.
- [51] —, "„nicht tonal und nicht atonal“ Zur Bedeutung der Quinten in Ligetis Etüde Nr. 8 „Fém“,," *Studia Musicologica*, vol. 57, no. 1-2, pp. 121–138, Jun. 2016.
- [52] M. Rohrmeier and M. Pearce, "Musical Syntax I: Theoretical Perspectives," in *Springer Handbook of Systematic Musicology*, R. Bader, Ed. Berlin: Springer, 2018, pp. 473–486.

- [53] J. Katz and D. Pesetsky, “The Identity Thesis for Language and Music,” *Draft published online: ling-Buzz/000959*, no. January, 2011.
- [54] C. Finkensiep, R. Widdess, and M. A. Rohrmeier, “Modelling the Syntax of North Indian Melodies with a Generalized Graph Grammar,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, no. CONF. ISMIR, Nov. 2019, p. 462.
- [55] K. Hirata, S. Tojo, and M. Hamanaka, “Implementing “A Generative Theory of Tonal Music”,” *Journal of New Music Research*, vol. 35, no. 4, pp. 249–277, 2006.
- [56] S. Abdallah, N. Gold, and A. Marsden, “Analysing symbolic music with probabilistic grammars,” in *Computational Music Analysis*, D. Meredith, Ed. Springer, 2016, pp. 157–189.
- [57] A. Marsden and G. A. Wiggins, “Schenkerian reduction as search,” *Fourth Conference on Interdisciplinary Musicology (CIM08)*, no. July, 2008.
- [58] M. Rohrmeier and M. Neuwirth, “Towards a Syntax of the Classical Cadence,” in *What Is a Cadence?*, M. Neuwirth and P. Bergé, Eds. Leuven: Leuven University Press, 2015, pp. 285–336.
- [59] D. Quick and P. Hudak, “Grammar-based automated music composition in Haskell,” *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design - FARM '13*, pp. 59–70, 2013. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2505341.2505345>
- [60] M. Steedman, “The blues and the abstract truth: Music and mental models,” *Mental models in cognitive science: essays in honour of Phil Johnson-Laird*, p. 305, 1996.
- [61] J. Katz, “Harmonic Syntax of the Twelve-Bar Blues Form,” *Music Perception: An Interdisciplinary Journal*, vol. 35, no. 2, pp. 165–192, Dec. 2017.
- [62] M. Granroth-Wilding and M. Steedman, “A robust parser-interpreter for jazz chord sequences,” *Journal of New Music Research*, 2014.
- [63] D. Harasim, M. Rohrmeier, and T. J. O’Donnell, “A Generalized Parsing Framework for Generative Models of Harmonic Syntax,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, Paris, France, Sep. 2018, pp. 152–159.
- [64] B. De Haas, M. Rohrmeier, R. Veltkamp, and F. Wiering, “Modeling harmonic similarity using a generative grammar of tonal harmony,” in *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, K. Hirata and G. Tzanetakis, Eds., 2009.
- [65] Y. Ogura, H. Ohmura, Y. Uehara, S. Tojo, and K. Katsurada, “Expectation-based parsing for jazz chord sequences,” in *The Proceedings of 17th SMC Sound and Music Computing Conference*, 2020.
- [66] L. Euler, “De harmoniae veris principiis per speculum musicum repraesentatis,” in *Novi Commentarii Academiae Scientiarum Petropolitanae*, St. Petersburg, 1774, vol. 18, pp. 330–353.
- [67] C. Longuet-Higgins, “The Three Dimensions of Harmony,” in *Mental Processes: Studies in Cognitive Science*. Cambridge, MA: MIT Press, 1987, pp. 59–63.
- [68] J. Douthett and P. Steinbach, “Parsimonious graphs: A study in parsimony, contextual transformations and modes of limited transposition,” *Journal of Music Theory*, vol. 42, no. 2, pp. 241–263, 1998.
- [69] M. Woolhouse, “Wagner in the Round: Using Interval Cycles to Model Chromatic Harmony,” in *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the European Society for the Cognitive Sciences of Music*, E. Cambouropoulos, C. Tsougras, P. Mavromatis, and K. Pasiadis, Eds., Thessaloniki, 2012, pp. 1142–1145.
- [70] E. Lendvai, *Béla Bartók: An Analysis of His Music*. London: Kahn & Averill, 1971.
- [71] O. Messiaen, *Technique de Mon Langage Musical*. Paris: Alphonse Leduc, 1944, vol. 1.
- [72] J. N. Straus, *Introduction to Post-Tonal Theory*, 3rd ed. New York: Pearson Prentice Hall, 2005.
- [73] A. Forte, *The Structure of Atonal Music*. New Haven and London: Yale University Press, 1977.
- [74] M. Levine, *The Jazz Theory Book*. Sebastopol, CA: O’Reilly Media, 2011.
- [75] ———, *The Jazz Piano Book*. Sher Music Co. / Advance Music, 1990.
- [76] R. Cohn, “Hexatonic poles and the uncanny in Parsifal,” *Opera Quarterly*, vol. 22, no. 2, pp. 230–248, Apr. 2006.
- [77] M. Steedman, *The Syntactic Process*. Cambridge, MA: MIT press, 2000.
- [78] R. Cohn, “On Hexatonic Poles,” *Music Analysis*, vol. 0, pp. 1–5, 2016.
- [79] N. Chomsky, “Formal properties of grammar,” in *Handbook of Mathematical Psychology*, R. Luce, R. Bush, and E. Galanter, Eds. New York: Wiley, 1963, pp. 323–418.
- [80] M. Rohrmeier, “Towards a formalization of musical rhythm,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020.

- [81] S. Tojo, “Modal logic for tonal music,” in *Perception, Representations, Image, Sound, Music: 14th International Symposium, CMMR 2019, Marseille, France, October 14–18, 2019, Revised Selected Papers 14*. Springer International Publishing, 2021, pp. 113–128.
- [82] D. Harasim, T. J. O’Donnell, and M. A. Rohrmeier, “Harmonic Syntax in Time: Rhythm Improves Grammatical Models of Harmony,” in *Proceedings of the 20th ISMIR Conference*, no. CONF. ISMIR, 2019, p. 335.

CRASH: RAW AUDIO SCORE-BASED GENERATIVE MODELING FOR CONTROLLABLE HIGH-RESOLUTION DRUM SOUND SYNTHESIS

Simon Rouard

Sony CSL - CentraleSupélec

simon.rouard@student-cs.fr

Gaëtan Hadjeres

Sony CSL

gaetan.hadjeres@sony.com

ABSTRACT

In this paper, we propose a novel score-base generative model for unconditional raw audio synthesis. Our proposal builds upon the latest developments on diffusion process modeling with stochastic differential equations, which already demonstrated promising results on image generation. We motivate novel heuristics for the choice of the diffusion processes better suited for audio generation, and consider the use of a conditional U-Net to approximate the score function. While previous approaches on diffusion models on audio were mainly designed as speech vocoders in medium resolution, our method termed CRASH (Controllable Raw Audio Synthesis with High-resolution) allows us to generate short percussive sounds in 44.1kHz in a controllable way. Through extensive experiments, we showcase on a drum sound generation task the numerous sampling schemes offered by our method (unconditional generation, deterministic generation, inpainting, interpolation, variations, class-conditional sampling) and propose the *class-mixing* sampling, a novel way to generate “hybrid” sounds. Our proposed method offers flexible generation capabilities with lighter and easier-to-train models than GAN-based methods.

1. INTRODUCTION AND RELATED WORK

After multiple works in the spectral domain [1, 2], deep generative models in the waveform domain have recently shown the ability to produce high fidelity results with different methods: autoregressive [3, 4], flow-based [5], energy-based [6] or based on Generative Adversarial Networks [7].

In the task of generating drum sounds in the waveform domain, GAN-based approaches have been explored in [7], [8] and [9]. Interactive sound design is often a major motivation behind these works: in [10] the authors use Variational Autoencoders (VAE) in order to generate spectrograms of drums apply a principal component analysis on the latent space of the VAE in order to explore the drum timbre space. One of the disadvantages of this model is

that the reconstruction of the sounds by the VAE tends to be blurry. In [11], the authors use a VQ-VAE2 [12] in order to perform inpainting on instrument sound spectrograms.

Score-based generative models [13–16] propose a different approach to generative modeling, which consists in estimating the gradient of noise-corrupted data log-densities (score function): by iteratively denoising a sampled noise, these approaches obtained promising results, but mainly on image data.

To this day, only two score-based generative models in the waveform domain have been published [17, 18] and they are mostly focused on the task of neural vocoding with conditioning on a mel-spectrogram. In [17], the authors achieved the task of generating audio with an unconditioned model trained on the speech command dataset [19]. The inference scheme of [17] does not provide a flexible sampling scheme because it is trained on a fixed discrete noise schedule whereas [18] is trained on a continuous scalar indicating the noise level.

In the image domain, [16] generalizes the works of [14, 15, 20] by framing the noise corruption procedure as stochastic differential equation.

Score-based generative models offer the following advantages over GAN-based approaches:

- Training time is reduced and training is more stable since there is only one network to train.
- Class-conditioning generation can be achieved by training a classifier a posteriori, which lets us train a model only one time.
- Data can be mapped to a latent space without the need to train an additional encoder compared to GANs [21], which makes the interpolation between two given input data readily available with only one model.

These properties alleviate us to search for directions in the latent space as in [22] or to directly hardcode conditional features in the architecture as in [23]. This easily controllable latent space permits sound design applications. One downside of score-based models compared to GANs is their higher inference times to generate new samples.

In this work, we extend the approach of [16] and propose CRASH (Controllable Raw Audio Synthesis with High-resolution), a score-based generative model adapted to the waveform domain. On a drum sound dataset, the numerous capabilities offered by this architecture allows for



musically-relevant sound design applications. Our contributions are the following:

- A score-based model for unconditional generation that can achieve high fidelity 44.1 kHz drum sounds directly in the waveform domain,
- The use of a noise-conditioned U-Net to estimate the score function,
- A novel *class-mixing* sampling scheme to generate "hybrid" sounds.
- Experimental and practical insights about the choice of the stochastic differential equation used to corrupt the data.

2. BACKGROUND

2.1 Score Based Modelling through Stochastic Differential Equations

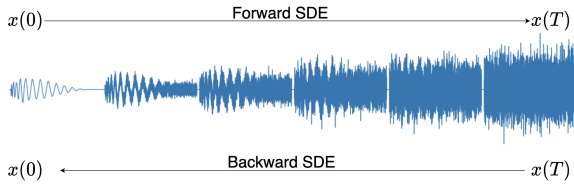


Figure 1. Illustration of the noising and denoising processes of a kick sound with a VP schedule

2.1.1 Forward Process

Let p_{data} be a data distribution. Diffusion models consist in progressively adding noise to the data distribution to transform it into a known distribution from which we can sample from as shown in Fig. 1. In [16], the authors formalize this noising process as the following **forward** Stochastic Differential Equation (SDE):

$$d\mathbf{x} = f(t)\mathbf{x}dt + g(t)d\mathbf{w} \quad (1)$$

where $f(t)$ is a continuous negative function from $[0, T] \rightarrow \mathbb{R}^-$, $g(t)$ a continuous positive function from $[0, T] \rightarrow \mathbb{R}^+$, and \mathbf{w} is a standard Wiener process. Such approach can be understood as a continuous-time generalization of Denoising Diffusion Probabilistic Models (DDPMs) [14, 20] and denoising Score Matching with Langevin Dynamics (SMLD) [15]. For $\mathbf{x}(0) \sim p_{\text{data}}$, the transition kernel of Eq. 1 is given by a normal distribution:

$$p_t(\mathbf{x}(t) | \mathbf{x}(0)) = \mathcal{N}(\mathbf{x}(t); m(t)\mathbf{x}(0), \sigma^2(t)\mathbf{I}), \quad (2)$$

where $m(t)$ and $\sigma(t)$ follow the system:

$$\begin{cases} \frac{dm}{dt} = f(t)m(t) \\ \frac{d\sigma^2(t)}{dt} = 2f(t)\sigma^2(t) + g^2(t) \end{cases} \quad (3)$$

with the following initial conditions $m(0) = 1$ and $\sigma^2(0) = 0$. The solutions for $m(t)$ and $\sigma(t)$ are :

$$\begin{cases} m(t) = e^{\int_0^t f(s)ds} \\ \sigma^2(t) = e^{\int_0^t 2f(s)ds} \int_0^t g^2(u)e^{\int_0^u -2f(s)ds} du. \end{cases} \quad (4)$$

In [16], the authors define three types of SDEs which are presented in Tab. 1. For the Variance Preserving (VP)

	$f(t)$	$g(t)$
VP	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}$
VE	0	$\sqrt{\frac{d[\sigma^2(t)]}{dt}}$
sub-VP	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)(1 - e^{-2\int_0^t \beta(s)ds})}$

Table 1. Functions used in the VP, VE and sub-VP SDEs

and sub-Variance Preserving (sub-VP) schedules, $m(T) \approx 0$ and $\sigma(T) \approx 1$ which means that the original data distribution is transformed into a distribution close to a standard normal distribution i.e. $p_T \approx \mathcal{N}(\mathbf{0}, \mathbf{I})$. For the Variance Exploding (VE), $\sigma^2(T) \gg m \approx 1$ which means that the original data is not discernable at $t = T$ and that $p_T \approx \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$.

2.1.2 Generation with the Reverse Process

In order to sample from the data distribution, we can sample $\mathbf{x}_T \sim p_T$ and apply the associated reverse time SDE [24] given by:

$$d\mathbf{x} = [f(t)\mathbf{x} - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\tilde{\mathbf{w}} \quad (5)$$

where $\tilde{\mathbf{w}}$ is a standard Wiener process running backwards from T to 0 and dt is an infinitesimal negative timestep.

It means that by knowing $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, we can use a discretization of Eq. 5 to sample $\mathbf{x}(0)$ from $p_0 = p_{\text{data}}$.

In practice, the score function $s(\mathbf{x}(t), \sigma(t)) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is intractable and it is approximated by a neural network $s_{\theta}(\mathbf{x}(t), \sigma(t))$ parameterized by θ . In order to train the network, [13] shows that for any t , minimizing

$$\mathbb{E}_{p_t(\mathbf{x})} \|s_{\theta}(\mathbf{x}, \sigma(t)) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\|_2^2 \quad (6)$$

is equivalent to minimizing

$$\mathbb{E} \|s_{\theta}(\mathbf{x}, \sigma(t)) - \nabla_{\mathbf{x}} \log p_t(\mathbf{x}(t) | \mathbf{x}(0))\|_2^2 \quad (7)$$

where the expectation is over $\mathbf{x}(0) \sim p_{\text{data}}$, $\mathbf{x}(t) \sim p_t(\mathbf{x}(t) | \mathbf{x}(0))$, and the latter distribution is given by Eq. 2.

Now, in order to train the network for all $t \in [0, T]$ we consider the following mixture of Eq. 7 losses over all noise levels:

$$L(\theta) = \mathbb{E} \lambda(t) \|s_{\theta}(\mathbf{x}(t), \sigma(t)) - \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t) | \mathbf{x}(0))\|_2^2 \quad (8)$$

where we sample $t \sim [0, T]$, $\mathbf{x}(0) \sim p_{\text{data}}$, $\mathbf{x}(t) \sim p_t(\mathbf{x}(t) | \mathbf{x}(0))$ and where $\lambda(t)$ is a weighting function.

In [14–16], $\lambda(t)$ is empirically set such that $\lambda(t)^{-1} \propto \mathbb{E} \left\| \nabla_{\mathbf{x}(t)} \log p_t(\mathbf{x}(t) | \mathbf{x}(0)) \right\|_2^2 \propto \sigma^2(t)^{-1}$ while in [25] the authors show that the maximum likelihood estimator is obtained with $\lambda(t) = g^2(t)$ in $L(\theta)$.

The training procedure is described in Alg. 1, where we reparameterize our neural network as $\epsilon_\theta(\mathbf{x}(t), \sigma(t)) := -\sigma(t)s_\theta(\mathbf{x}(t), \sigma(t))$ in order to estimate ϵ .

Algorithm 1 Training procedure

while Training **do**

 Sample $t \sim \mathcal{U}([0, T])$, $\mathbf{x}(0) \sim p_{\text{data}}$, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

 Compute $\mathbf{x}(t) = m(t)\mathbf{x}(0) + \sigma(t)\epsilon$

 Gradient descent on $\nabla_\theta \left\| \frac{\sqrt{\lambda(t)}}{\sigma(t)} [\epsilon_\theta(\mathbf{x}(t), \sigma(t)) - \epsilon] \right\|_2^2$

end while

Once the network is trained, a N-step discretization of the **backward** SDE is done in order to unconditionally generate samples. This process is described in Alg. 2, it is non-deterministic since we obtain various sounds by starting from the same sample $\mathbf{x}(T)$.

Algorithm 2 Sampling via SDE

Choose N , sample $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$

for $i = N - 1, \dots, 0$ **do**

$t_i = T \frac{i}{N}$, $f_i = f(t_i)$, $g_i = g(t_i)$, $\sigma_i = \sigma(t_i)$

$\mathbf{x}_i = (1 - \frac{f_{i+1}}{N})\mathbf{x}_{i+1} - \frac{g_{i+1}^2}{N\sigma_{i+1}}\epsilon_\theta(\mathbf{x}_{i+1}, \sigma_{i+1})$

if $i > 0$ **then**

 Sample $\mathbf{z}_{i+1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_i = \mathbf{x}_i + \frac{g_{i+1}}{\sqrt{N}}\mathbf{z}_{i+1}$

end if

end for

2.2 Deterministic Sampling via Score based Ordinary Differential Equation

As mentioned in [16], for any SDE, there exists a corresponding deterministic process which satisfies an ordinary differential equation (ODE):

$$d\mathbf{x} = [f(t)\mathbf{x} - \frac{1}{2}g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt \quad (9)$$

This defines a flow ϕ^t such that the marginal distributions $\phi_*^t(p_{\text{data}})$ are identical to the ones obtained by applying the SDE of Eq. 1. This mapping is interesting because it provides a latent representation for each $\mathbf{x} \sim p_{\text{data}}$.

The procedure of sampling via the N-step discretization of the ODE is described in Alg. 3.

Algorithm 3 Sampling via ODE

Choose N , sample $\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma^2(T)\mathbf{I})$

for $i = N - 1, \dots, 0$ **do**

$t_i = T \frac{i}{N}$, $f_i = f(t_i)$, $g_i = g(t_i)$, $\sigma_i = \sigma(t_i)$

$\mathbf{x}_i = (1 - \frac{f_{i+1}}{N})\mathbf{x}_{i+1} - \frac{g_{i+1}^2}{2N\sigma_{i+1}}\epsilon_\theta(\mathbf{x}_{i+1}, \sigma_{i+1})$

end for

2.3 Inpainting

Let's imagine that we don't like the attack of a kick (or any other part of a sound), the method of inpainting permits us to regenerate the desired part. In order to do that, we apply a reverse-time SDE or ODE discretization to an isotropic Gaussian and fix the part that we want to keep (with the associated noise corruption) after each denoising timestep. As presented in section 6, we obtain very diverse and coherent results.

2.4 Interpolations

The flexibility of SDEs and ODEs allows to compute interpolations between sounds. In fact, there exists an infinity of latent spaces indexed by $t \in [0, T]$. We present here two types of interpolations: ODE interpolation in the latent space of isotropic Gaussians and SDE interpolation in any t-indexed latent space.

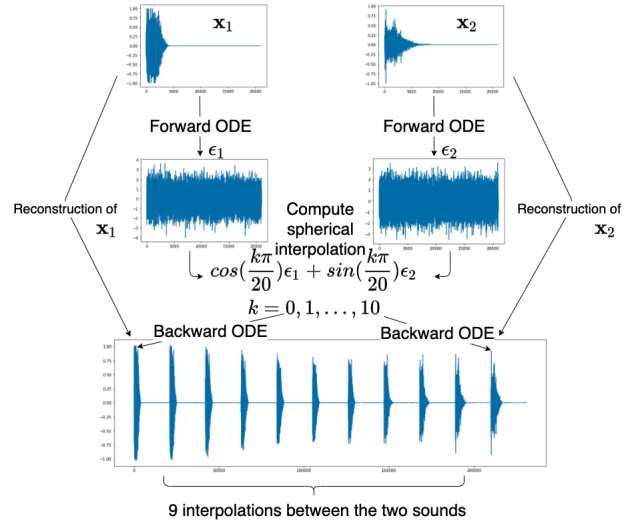


Figure 2. Interpolation of two sounds via Forward and Backward ODE

2.4.1 ODE interpolation in the latent space of isotropic Gaussians

Let ϵ_1 and ϵ_2 be two samples from a standard normal distribution of \mathbb{R}^L where L is our space dimension and $0 \leq \lambda \leq 1$. We consider the spherical interpolation $\epsilon_\lambda = \lambda\epsilon_1 + \sqrt{1 - \lambda^2}\epsilon_2$ and then apply the ODE sampling to it. We choose a spherical interpolation in order to preserve a variance close to 1 for ϵ_λ .

Moreover, if we want to interpolate two sounds \mathbf{x}_1 and \mathbf{x}_2 , we can apply the Forward ODE in order to obtain the corresponding latent codes ϵ_1 and ϵ_2 , apply the desired spherical interpolation and then apply an ODE sampling.

2.4.2 ODE interpolation in a t-indexed latent space

In [17], the authors perform a linear interpolation between two sounds at a corresponding intermediate t-indexed latent space before applying Denoising Diffusion Probabilistic Model (DDPM is the discrete equivalent of a VP SDE). We adapt the method to the continuous framework with

SDE and ODE. Here again, the interpolation can be done between two t -indexed latent codes or between sounds corrupted using the transition kernel of Eq. 2.

2.5 Class-Conditional sampling with a classifier

For any class y , we can train a noise-conditioned classifier on corrupted data $\mathbf{x}(t)$. As a consequence, the output of the classifier gives us $p_t(y | \mathbf{x}(t))$ for each class y . We can use automatic-differentiation to differentiate this quantity and by the Bayes Formula, since $p(y)$ is constant for each class y , we have the following formula:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x} | y) = \nabla_{\mathbf{x}} \log p_t(\mathbf{x}) + \nabla_{\mathbf{x}} \log p_t(y | \mathbf{x}) \quad (10)$$

As a consequence, we can generate samples of one class by solving this reverse time SDE:

$$d\mathbf{x} = [f(t)\mathbf{x} - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x} | y)]dt + g(t)d\tilde{\mathbf{w}} \quad (11)$$

This approach is flexible since it only requires to train a noise-conditioned classifier: there is no need to design and train a class-conditional score-based model as done in [17].

3. A DISCUSSION ABOUT CHOOSING THE RIGHT SDE: A GENERALIZATION OF THE SUB-VP SDE

In this section $T = 1$.

3.1 About the relation between $m(t)$ and $\sigma(t)$

The VP SDE is the continuous version of the Denoising Diffusion Probabilistic Model (DDPM) used in [14] [17] [18]. One of the main features of this model is that the mean coefficient $m(t)$ of the perturbation kernel is linked to the standard deviation $\sigma(t)$ (or noise-level) by the following equation $m(t) = \sqrt{1 - \sigma^2(t)}$. Because the decrease of m is relatively small on a large range of values for σ , this means that a (very-noisy) drum sound audio must begin to appear after only a few steps of denoising during the sampling algorithm. (For instance if $\sigma = 0.8$, $m = 0.6$). We believe that this fast decay of the signal-to-noise ratio can be detrimental when sampling with Alg. 2 and 3.

Moreover, without mentioning this fact, in [16] the authors introduce the sub-VP SDE which is characterized by the following formula $m(t) = \sqrt{1 - \sigma(t)}$. The authors obtained state of the art results on the image generation task which corroborates our intuition that m might be too large for values of σ near 1 tends to be right.

In this work, we explore the four relations between m and σ plotted in Fig. 3. The blue one corresponds to the VP SDE, the yellow is the sub-VP SDE and the green and red ones corresponds to a generalization of the sub-VP schedule. In Tab. 2 we write the functions $f(t)$ and $g(t)$ for each of these 4 relation. For the rest of the paper we take the convention $f(t) := -\frac{1}{2}\beta(t)$ in order to compare the VP and sub-VP schedules with ours.

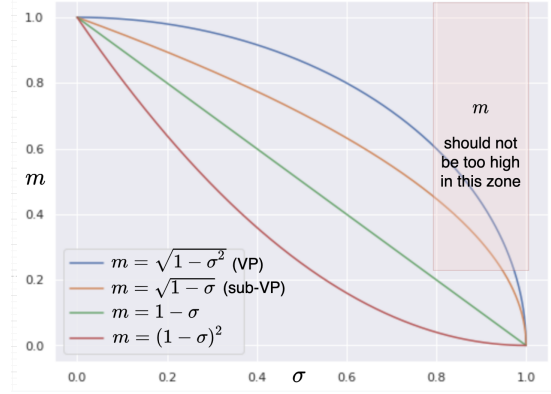


Figure 3. Different relations between m and σ

m - σ relation	$f(t)$	$g(t)$
$m = \sqrt{1 - \sigma^2}$ (VP)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}$
$m = \sqrt{1 - \sigma}$ (sub-VP)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}(1 - e^{-2\int_0^t \beta(s)ds})$
$m = 1 - \sigma$ (sub-VP 1-1)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}(1 - e^{-\frac{1}{2}\int_0^t \beta(s)ds})$
$m = (1 - \sigma)^2$ (sub-VP 1-2)	$-\frac{1}{2}\beta(t)$	$\sqrt{\beta(t)}(1 - \frac{3}{2}e^{-\frac{1}{2}\int_0^t \beta(s)ds} + \frac{1}{2}e^{-\int_0^t \beta(s)ds})$

Table 2. Functions used in the VP, sub-VP and generalized sub-VP SDEs.

3.2 Choosing the right functions for the SDE

Choosing a particular relation between m and σ , imposes a relation between g and β . The remaining free parameter is the function β , needed to fully define the SDE. In [16], the authors use a linear schedule for $\beta(t)$ because it is the continuous generalization of DDPMs. As presented in Fig. 4, this choice leads to a $\sigma(t)$ function that rapidly grows to its maximum. In [26], the authors mention this fast growing σ function as a potential shortcoming and propose a smoother function (the green one in Fig. 4).

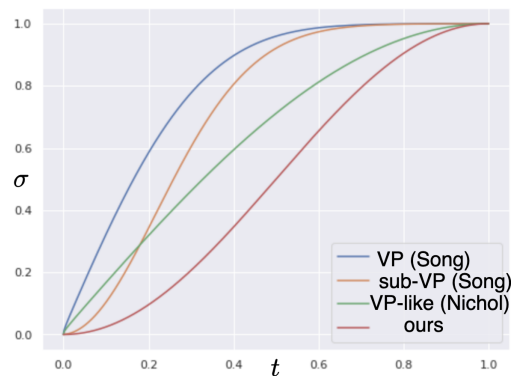


Figure 4. Different choices for the $\sigma(t)$ function

Our approach differs from [16] in that the definition of our SDE is motivated by choosing a relatively smooth increasing function $\sigma(t)$ such as $\sigma(0) = 0$ and $\sigma(1) = 1 - \epsilon$ (where ϵ is a small constant), together with a m - σ relation, from which all other quantities can be computed as shown in Tab. 2. If the two approaches are equivalent, we believe that these quantities are more interpretable. In the regime of a small number of discretization steps, a slow increasing

function may induce less approximation errors. For our experiments we propose $\sigma(t) = \frac{1}{2}[1 - \cos((1-s)\pi t)]$ with $s = 0.006$ which is the red plot in Fig. 4. We also sample t in the interval $[\eta, 1]$ during the training where η is chosen such that $\sigma(\eta) = 10^{-4}$ because 10^{-4} is imperceptible.

4. CLASS-MIXING SAMPLING WITH A CLASSIFIER

Drum classes are not perfectly distinct. For instance, the dataset contains drum sounds that are percussive enough to be seen as kicks but also sufficiently brilliant to be seen as snares and some kicks are combined with a hi-hat sound. We observe that our classifier (at the noise-level $\sigma = 0$) sometimes outputs a mixed classes such as $[0.3, 0.3, 0.4]$ and that it aligns well with our feeling when hearing the sound.

We introduce the Class-Conditional sampling to a mixture of classes: For a given noisy sound $\mathbf{x}(t)$, the vector $\nabla_{\mathbf{x}(t)} \log p_t(y_i | \mathbf{x}(t))$ points out to the direction of the class y_i in the noisy t -indexed latent space. Now, assuming that we have N classes $(y_i)_{i=1, \dots, N}$, let $(\lambda_i)_{i=1, \dots, N}$ be positive real numbers such as $\sum_{i=1}^N \lambda_i = 1$, we define a mixture of classes that we note $\{(y_i, \lambda_i)\}$ and the associated vector:

$$\nabla_{\mathbf{x}} \log p_t(\{(y_i, \lambda_i)\} | \mathbf{x}) := \sum_{i=1}^N \lambda_i \nabla_{\mathbf{x}} \log p_t(y_i | \mathbf{x}) \quad (12)$$

In practice, we put this term in equation 10 in replacement of the last term and use equation 11 to sample class-mixed audios. It gives us interesting results with a great palette of sounds.

5. ARCHITECTURE

5.1 Conditioned U-Net

Our model architecture is a conditioned U-Net [27], originally proposed for source separation. It takes two inputs: the noise level $\sigma(t)$ and the noisy audio $\mathbf{x}(t)$. The noise-level is encoded by Random Fourier Features [28] followed by a Multi-Layer Perceptron. The noisy audio goes into FiLM-conditioned [29] Downsampling Blocks. Then, the signal goes into Upsampling Blocks that receive skip connections from the DBlocks of same levels. The output of the network is the estimated noise $\epsilon_{\text{estimated}}$.

This bears similarities with the architecture from [17] which has a similar succession of blocks with dilated convolutions but no downsampling or upsampling layers, which makes it slow in terms of computation. The architecture from [18] has a U-Net-like shape [30], but heavily depends on the spectrogram conditioning and relies on a different noise-conditioning scheme. The σ -conditioned U-Net architecture seems to retain advantages from both approaches and is particularly suited for unconditional generation (see Fig. 5).

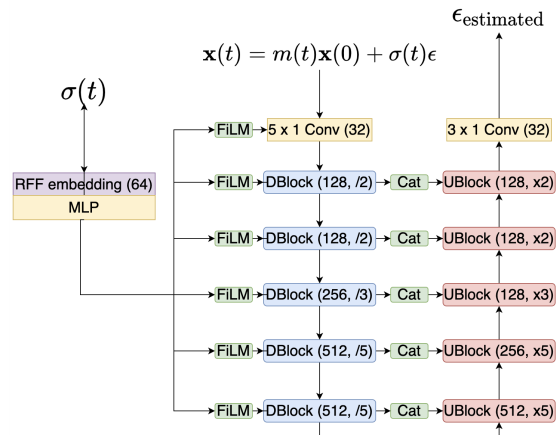


Figure 5. Architecture of the Conditioned U-Net

5.2 Noise conditioned classifier

Our noise-conditioned classifier closely mimics the architecture of our Conditioned U-Net presented in Sect. 5.1. The classifier is composed of a succession of FiLM-conditioned DBlocks followed by a projection layer and a softmax.

6. EXPERIMENTS AND RESULTS

Code is available at: <https://github.com/simonrouard/CRASH>

6.1 Dataset

For this work, we use an internal non-publicly available dataset of drum sounds which has also been used in [8]. It is composed of approximately 300.000 one-shot kick, snare and cymbal sounds in equal proportions. The samples have a sample rate of 44.1kHz and are recorded in mono. We restricted and padded the audio to 21.000 time-steps because most sounds last less than 0.5 second. We used 90% of the dataset in order to train our model.

6.2 Models and process

We evaluate the influence of $\sigma(t)$ and four m - σ schedules. The training of the network is done with a learning rate of 2.10^{-4} and the Adam optimizer. In parallel, smoothed weights with exponential moving average (EMA) with a rate of 0.999 are computed and saved at each step. For each model, the network is trained for about 120 epochs and the weights are saved each 8 epochs. We generated drum sounds with the regular weights and with the EMA weights and we observed the same phenomenon as in [31]: for the regular weights the quality of the sounds is not necessarily increasing with the training time whereas the EMA weights provide better and more homogeneous Fréchet Audio Distance [32] (FAD) during training¹.

¹ We use the original implementation [32] available at https://github.com/google-research/google-research/tree/master/frechet_audio_distance

After generating 2700 sounds for each checkpoint of each model, we choose the best checkpoints and generate 27000 drum sounds for each. It takes 12 hours to generate 27000 drum sounds on a Nvidia RTX-3090 GPU with an ODE or SDE schedule of 400 steps and batches of 180 sounds per generation (maximum memory capacity).

6.3 Quantitative Results

We report the FAD (lower is better) between the 27000 generated drum sounds and the test set for each unconditional generation with SDE and ODE (with a discretization of 400 steps) in the table 6.3. The cos schedule refers to the function $\sigma(t) = \frac{1}{2}[1 - \cos((1 - s)\pi t)]$ (the red one in Fig. 4) and the exp schedule corresponds to the function $\sigma(t) = \sqrt{1 - e^{-0.1t - 9.95t^2}}$ used in [16] (the blue one in Fig. 4).

Schedule	SDE 400 steps	ODE 400 steps
VP exp schedule (as in [16])	4.30	4.03
VP cos schedule	2.32	1.79
sub-VP cos schedule	2.46	2.07
sub-VP 1-1 cos schedule	2.62	1.73
sub-VP 1-2 cos schedule	2.56	1.75

Table 3. FAD comparison (lower is better)

Choosing a smoother σ function indeed improves the FAD of the generated sounds for a fixed number of discretization steps.

By using the classifier that we trained, we observe that all models generate kicks, snares and cymbals in equal proportions but the generated samples are less diverse than in the original dataset. For a fixed number of discretization steps, we think that the cos schedule performs better because it is smoother than the exp schedule.

6.4 Interactive sound design

Audio samples for all experiments described in this section can be heard on the accompanying website: <https://crash-diffusion.github.io/crash/>.

6.4.1 Interpolations

The relative lack of diversity of the unconditional generation is not dramatic since the model can still perform interactive sound design by modifying existing samples from the dataset. In order to do that, we apply the forward ODE to an existing sound and obtain its corresponding noise in the latent space of isotropic Gaussians. As presented in Fig. 7, we can perform spherical combinations on the latent codes and apply the backward ODE to obtain interpolations. Moreover the reconstructed sounds (at the left and right of the schema) are accurate.

6.4.2 Obtaining Variations of a Sound by Noising it and Denoising it via SDE

Let's take a sound $\mathbf{x}(0)$. We can noise it at a desired noise level $\sigma(t)$ via $\mathbf{x}(t) = m(t)\mathbf{x}(0) + \sigma(t)\epsilon$ and then denoise

it with a SDE discretization from t to 0. We obtain then variations of the original sound.

6.4.3 Inpainting

We can also perform inpainting on a sound in order to regenerate any desired part. We show this method on Fig. 6 where we regenerate 6 endings of a snare sound.

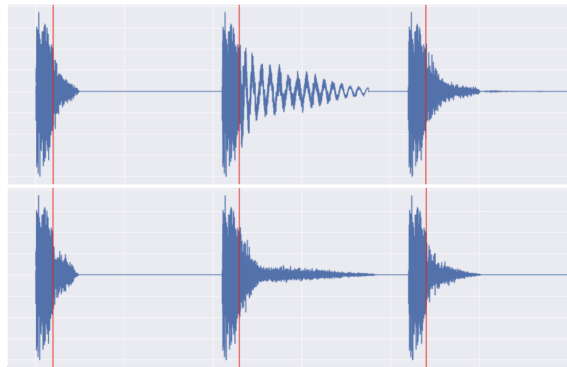


Figure 6. Six Inpaintings on the end of a snare sound

This provides an innovative way to generate a variety of plausible sounds starting with the same attack.

6.4.4 Class-Conditioning and Class-Mixing with a Classifier

We trained a noise-conditioned classifier on the 3 classes (kick, snare, cymbal) and used it to generate class-conditioned and class-mixing generation. Once again, by using the latent representation of a sound we can regenerate it (via ODE) with control on its "kickiness, snariness or cymbaliness".

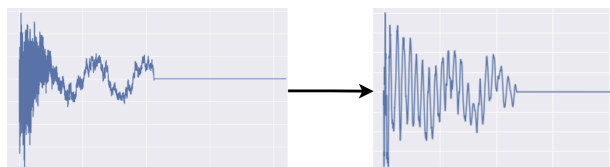


Figure 7. Transformation of a cymbal into a kick via class-conditioning ODE

7. CONCLUSION

We presented CRASH, a score-based generative model for the generation of raw audio based on the latest developments in modeling diffusion processes via SDEs. We proposed novel SDEs, well-suited to drum sound generation with high-resolution, together with an efficient architecture for estimating the score function. We showcased how the many controllable sampling schemes offered new perspectives for interactive sound design. In particular, our proposed *class-mixing* strategy allows the controllable creation of convincing "hybrid" sounds that would be hard to obtain with conventional means. We hope that these new methods will contribute to enrich the workflow of music producers.

8. REFERENCES

- [1] S. Vasquez and M. Lewis, “Melnet: A generative model for audio in the frequency domain,” 2019.
- [2] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, “Gansynth: Adversarial neural audio synthesis,” 2019.
- [3] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “Wavenet: A generative model for raw audio,” 2016.
- [4] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, “Samplernn: An unconditional end-to-end neural audio generation model,” 2017.
- [5] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” 2018.
- [6] A. A. Gritsenko, T. Salimans, R. van den Berg, J. Snoek, and N. Kalchbrenner, “A spectral energy distance for parallel speech synthesis,” 2020.
- [7] C. Donahue, J. McAuley, and M. Puckette, “Adversarial audio synthesis,” 2019.
- [8] J. Nistal, S. Lattner, and G. Richard, “Drumgan: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks,” 2020.
- [9] J. Drysdale, M. Tomczak, and J. Hockman, “Adversarial synthesis of drum sounds,” 2020.
- [10] C. Aouameur, P. Esling, and G. Hadjeres, “Neural drum machine: An interactive system for real-time synthesis of drum sounds,” in *International Conference on Computational Creativity*, 2019.
- [11] T. Bazin, G. Hadjeres, P. Esling, and M. Malt, “Spectrogram inpainting for interactive generation of instrument sounds,” *arXiv preprint arXiv:2104.07519*, 2021.
- [12] A. Razavi, A. v. d. Oord, and O. Vinyals, “Generating diverse high-fidelity images with vq-vae-2,” *NeurIPS*, 2019.
- [13] P. Vincent, “A connection between score matching and denoising autoencoders,” pp. 1661–1674, 2011.
- [14] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” 2020.
- [15] Y. Song and S. Ermon, “Generative modeling by estimating gradients of the data distribution,” in *Advances in Neural Information Processing Systems*, 2019, pp. 11 895–11 907.
- [16] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, “Score-based generative modeling through stochastic differential equations,” 2021.
- [17] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, “Diffwave: A versatile diffusion model for audio synthesis,” 2021.
- [18] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, “Wavegrad: Estimating gradients for waveform generation,” 2020.
- [19] P. Warden, “Speech commands: A dataset for limited-vocabulary speech recognition,” 2018.
- [20] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep unsupervised learning using nonequilibrium thermodynamics,” 2015.
- [21] O. Tov, Y. Alaluf, Y. Nitzan, O. Patashnik, and D. Cohen-Or, “Designing an encoder for stylegan image manipulation,” *CoRR*, vol. abs/2102.02766, 2021. [Online]. Available: <https://arxiv.org/abs/2102.02766>
- [22] E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris, “Ganspace: Discovering interpretable gan controls,” 2020.
- [23] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” 2014.
- [24] B. D. O. Anderson, “Reverse-time diffusion equation models,” pp. 313–326, 1982.
- [25] C. Durkan and Y. Song, “On maximum likelihood training of score-based generative models,” 2021.
- [26] A. Nichol and P. Dhariwal, “Improved denoising diffusion probabilistic models,” 2021.
- [27] G. Meseguer-Brocal and G. Peeters, “Conditioned-u-net: Introducing a control mechanism in the u-net for multiple source separations,” *ArXiv*, vol. abs/1907.01277, 2019.
- [28] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, “Fourier features let networks learn high frequency functions in low dimensional domains,” 2020.
- [29] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, “Film: Visual reasoning with a general conditioning layer,” 2017.
- [30] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” 2015.
- [31] Y. Song and S. Ermon, “Improved techniques for training score-based generative models,” 2020.
- [32] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, “Fréchet audio distance: A metric for evaluating music enhancement algorithms,” 2019.

CURRICULUM LEARNING FOR IMBALANCED CLASSIFICATION IN LARGE VOCABULARY AUTOMATIC CHORD RECOGNITION

Luke Rowe
University of Victoria
lukerowe@uvic.ca

George Tzanetakis
University of Victoria
gtzan@cs.uvic.ca

ABSTRACT

A problem inherent to the task of large vocabulary automatic chord recognition (ACR) is that the distribution over the chord qualities typically exhibits power-law characteristics. This intrinsic imbalance makes it difficult for ACR systems to learn the rare chord qualities in a large chord vocabulary. While recent ACR systems have exploited the hierarchical relationships that exist between chord qualities, few have attempted to exploit these relationships explicitly to improve the classification of rare chord qualities.

In this paper, we propose a convolutional Transformer model for the task of ACR trained on a dataset of 1217 tracks over a large chord vocabulary consisting of 170 chord types. In order to address the class imbalance of the chord quality distribution, we incorporate the hierarchical relationships between chord qualities into a curriculum learning training scheme that gradually learns the rare and complex chord qualities in the dataset. We show that the proposed convolutional Transformer model achieves state-of-the-art performance on traditional ACR evaluation metrics. Furthermore, we show that the proposed curriculum learning training scheme outperforms existing methods in improving the classification of rare chord qualities.

1. INTRODUCTION

The task of automatic chord recognition (ACR) has been an active area of research in the field of music information retrieval (MIR) for over 20 years [1]. This task automates the process of chord sequence annotation, which can be time consuming when done manually. ACR systems have been shown to be useful in other MIR applications, as chord annotations can be used as descriptive low-level features to assist other MIR tasks, such as key detection, harmonic analysis, and even style analysis [2]. Typically, an ACR system takes as input the audio signal corresponding to a musical recording. Then, the system outputs a time-aligned sequence of chord labels describing the underlying harmonic structure of the musical recording.

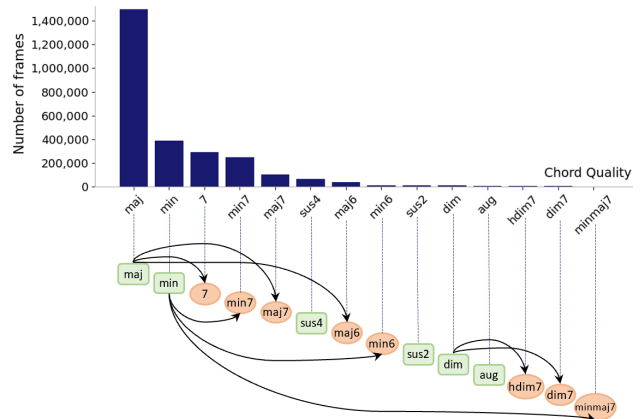


Figure 1: Top: Power-law distribution of chord qualities in BRIM (see Section 3 for details). Bottom: Hierarchy of chord qualities.

Most early ACR systems operated over a small chord vocabulary consisting of only major and minor chords and lacked the complexity needed for more complex chords. Recently, the focus has shifted to large vocabulary ACR, which includes a wider variety of chord qualities, such as augmented, diminished, sixth, seventh, and suspended chords. A critical issue with large vocabulary ACR is that the distribution over the chord qualities – and hence over the chord classes – exhibits a power-law distribution (see Figure 1). This imbalance is not specific to any particular ACR dataset but is intrinsic to large vocabulary ACR. Specifically, chord progressions seen across almost all genres of music overwhelmingly favor the major and minor chord qualities, which makes it difficult for ACR systems to learn the rare chord qualities.

Despite the complexities that arise in large vocabulary ACR, important structural relationships exist between the chord qualities in a large chord vocabulary [3]. As outlined in Figure 1, the chord qualities can be arranged into a hierarchical structure consisting of *base* chord qualities, or triads, (in rectangles) and *extended* chord qualities, or tetrads, (in ovals). Given an extended chord quality q_E and a base chord quality q_B , we say q_E *extends* q_B if the set of intervals that defines q_E is a superset of the set of intervals that defines q_B . Each extended chord quality extends a corresponding base chord quality, with the *extends* relationship indicated by an arrow in Figure 1. We hypothesize that an ACR model can better learn an extended chord quality when it has sufficiently learned its corresponding

base chord quality. The underlying intuition is that the base chord quality can be viewed as a harmonic base for the extended chord quality, and thus each extended chord quality can be interpreted as a more complex variant of its corresponding base chord quality. For example, the extended chord quality 7 can be viewed as a variant of base chord quality $major$, where an additional $b7$ interval is included.

Each extended chord quality is *rarer* in frequency than its corresponding base chord quality as can be seen in Figure 1. Based on this observation, we introduce a curriculum learning (CL) reweighting scheme that gradually converges from the initial distribution to a balanced chord quality distribution. This way, the curriculum allows the model to learn the base chord qualities prior to learning its corresponding extended chord qualities. The proposed scheme is integrated with a convolutional Transformer model which is trained over a chord vocabulary of 170 chord types. We show that the proposed model achieves state-of-the-art performance on traditional ACR evaluation metrics. Moreover, we show that the proposed CL scheme outperforms existing methods in improving the classification of rare chord qualities.

2. RELATED WORK

2.1 Automatic Chord Recognition

Most ACR systems have two stages: feature extraction and chord sequence decoding. Arguably the most notable recent advancement in ACR is the replacement of traditional machine learning with deep learning for both stages of the ACR pipeline. For example, recent ACR systems have utilized deep neural networks [4–6], convolutional neural networks [7–11], and deep belief networks [12, 13] to produce robust feature representations that outperform earlier conventional methods. Moreover, recurrent neural networks [4, 9–14] and conditional random fields (CRFs) [8, 10, 11] have largely replaced hidden Markov models to capture the temporal dependencies in the chord sequence decoding process. Inspired by the recent success of Transformer-based models in the field of natural language processing [15–18], recent approaches have applied end-to-end Transformer-based models to the task of ACR [19] and to the related tasks of symbolic chord recognition and functional harmony recognition [20, 21].

Recent focus has shifted to the large vocabulary variant of the ACR task [3, 9, 11, 13, 14]. Since the chord quality distribution over a large chord vocabulary is extremely skewed, these systems must explicitly overcome the *imbalanced class-learning problem*, whereby model learning is biased towards the frequently-labelled classes, resulting in poor classification performance of the sparsely-labelled classes [22]. To address this problem within large vocabulary ACR, recent approaches have incorporated auxiliary training targets by decomposing chords into structured components [9, 11]. However, these structured training methods still provide limited exposure to the rare chord qualities, and thus model learning is still heavily biased towards the frequently-labelled chord qualities. Deng and

Kwok addressed this problem by implementing an “even-chance” training scheme, which ensures that each chord type has an even chance of being chosen at the beginning of each training sample [14]. Jiang *et al.* combined their structured chord representation with a reweighting scheme to reduce the model learning bias induced by the imbalanced distribution of each structured component [11].

2.2 Curriculum Learning

CL was first proposed by Bengio *et al.* in [23], where they demonstrated that for certain tasks with an established difficulty metric, introducing training data from easy to hard difficulty in a deep neural network can lead to faster convergence and guide training towards better local minima. To the best of our knowledge, the only existing application of CL to the task of ACR is in [24]. In [24], McVicar *et al.* designed a curriculum to train an ACR system using ground-truth annotations with noisy alignments. CL has recently been utilized to address the imbalanced class-learning problem. In [25], Wang *et al.* proposed a CL training scheme for the task of human attribute recognition, which gradually converges from the training distribution to a balanced distribution to improve the classification performance of sparsely-labelled classes.

3. DATA PREPARATION

The ground-truth chord labels are mapped to a chord vocabulary V of 170 chords [9]. V includes chords that span all 12 pitch classes and the following 14 chord qualities Q : maj , min , dim , aug , $min6$, $maj6$, $min7$, $minmaj7$, $maj7$, 7 , $dim7$, $hdim7$, $sus2$, and $sus4$. Additionally, V contains two extra labels: N (no chord) and X (unknown chord). Our model also integrates the structured chord representation proposed in [9], whereby each chord label is decomposed into its root, bass, and pitch structured components.

For training and evaluation, we use the dataset collected by Humphrey and Bello [9, 11, 26], which comprises of 1217 tracks from the **Billboard**, **RWC Pop**, **Isophonics**, and **MARL** collections. We refer to this collected dataset as **BRIM**. We augment the training data using MUDA [27] by pitch-shifting each audio track across -6 to $+5$ semitones. To properly compare the performance on traditional ACR metrics with previous methods, we use the same 5-fold cross-validation split that is used in [9, 11, 26].

We employ a separate 5-fold cross-validation split for the imbalanced class-learning ACR experiments. Since the chord-type distribution in BRIM is extremely imbalanced, it is imperative for proper evaluation that this distribution is maintained across each fold of our 5-fold split. However, since the 5-fold split occurs at the *track* level but the distribution is measured at the *frame* level, we found stratifying over the chord types to be intractable. Therefore, we propose an approximate 5-fold stratification algorithm that instead ensures that the distribution over the *chord qualities* in BRIM is approximately maintained across each fold. The proposed algorithm (Algorithm 1) takes as input a set of *chord quality profiles* $P = \{P_t\}$, where $P_t[q]$ is the pro-

Algorithm 1: N -Fold Chord Quality Stratification

Input: Number of folds: N ; set of chord qualities: Q ; set of tracks T ; chord quality profiles: $P = \{P_t\}$; rarest chord quality: q_r .

for $i = 0$ **to** $N - 1$ **do**
 initialize empty list $folds[i]$
 initialize fold profile F_i , where for each $q \in Q$,
 $F_i[q] = 0$

$T_{sorted} = \text{SortDescending}(T, \text{by}=P[q_r])$

for $i = 0$ **to** $N - 1$ **do**
 append $T_{sorted}[i]$ to $folds[i]$
 $F_i[q] = F_i[q] + P_{T_{sorted}[i]}[q]$ for each $q \in Q$
 remove $T_{sorted}[i]$ from T

while $T \neq \emptyset$ **do**
 $q' = \underset{q \in Q}{\operatorname{argmax}} \operatorname{Var}(F_0[q], \dots, F_{N-1}[q])$
 $t_{\min} = \underset{t \in T}{\operatorname{argmin}} P_t[q']$
 $i_{\max} = \underset{i \in \{0, \dots, N-1\}}{\operatorname{argmax}} F_i[q']$
 append t_{\min} to $folds[i_{\max}]$
 $F_{i_{\max}}[q] = F_{i_{\max}}[q] + P_{t_{\min}}[q]$ for each $q \in Q$
 remove t_{\min} from T
 for each remaining fold $i \neq i_{\max}$ **do**
 $t' = \underset{t \in T}{\operatorname{argmin}} P_t[q'] + F_i[q'] - F_{i_{\max}}[q']$
 append t' to $folds[i]$
 $F_i[q] = F_i[q] + P_{t'}[q]$ for each $q \in Q$
 remove t' from T

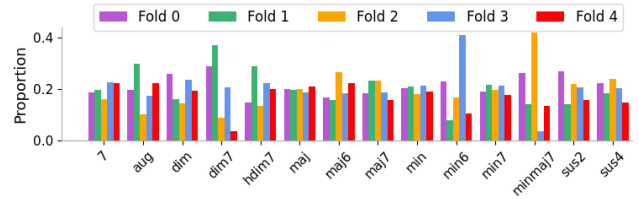
return $folds$

portion of q chords in track t over BRIM, for each $q \in Q$. The algorithm iteratively builds up a fold profile F_i for each fold i , where $F_i[q]$ is the proportion of q chords in fold i over BRIM, for each $q \in Q$. At each iteration, one track is added to each fold. At iteration 1, we take the 5 tracks with the highest proportion of the rarest chord quality q_r and add one track to each fold. Each subsequent iteration can be viewed as a ‘‘correction step,’’ whereby one track is added to each fold to minimize the variance of the highest-variance chord quality over the folds. The result of the 5-fold chord quality stratification is shown in Figure 2b.

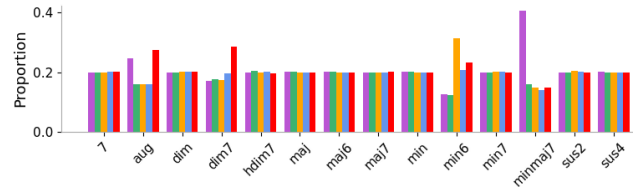
4. METHODS

4.1 Convolutional Transformer (CT)

The proposed system uses the log-power Constant Q-Transform (CQT) spectrogram as its input feature representation. We apply a convolutional-residual encoder shown in Figure 3b to capture short-term context and induce sufficient temporal smoothing of the spectrogram. The encoder first applies batch-normalization (BN) [28] to the input CQT. We then apply a series of convolutional layers and add 3 skip connections [29] to ease the training process. Each convolutional layer is zero-padded to preserve the spatial dimension of the CQT. After each convolutional layer, a BN layer followed by a Rectified Linear Unit (ReLU) activation is applied. The output of the



(a) 5-fold split used in [9, 11, 26].



(b) 5-fold Stratified Split.

Figure 2: 5-fold splits of BRIM. The height of each bar at chord quality q corresponds to the proportion of q chords contained in the corresponding fold.

convolutional encoder is passed through a stack of N bi-directional self-attention layers proposed in [19]. Details of this layer can be found in [19]. We replace the absolute positional encoding used in [19] with relative positional encoding [16], which has been shown to offer better generalization capabilities by taking into account the relative positions between frames in the self-attention mechanism.

The model facilitates structured training of the root note, bass note, and pitch classes as is done in [9]. Unlike in [9], we multiply the pitch structured loss by $\gamma > 1$ to assign more priority to the pitch structured component. The model learns to jointly minimize the cross-entropy chord label loss L_{label} and the cross-entropy structured loss L_{struct} , where L_{struct} is the sum of the cross entropy losses for the pitch, root, and bass structured components. For each frame t , the model outputs a softmax distribution $\hat{y}^{(t)} \in [0, 1]^{|V|}$ over V . At evaluation time, the system predicts the label with the highest activation in $\hat{y}^{(t)}$ for each frame t . An overview of the model is shown in Figure 3.

4.2 Curriculum Learning

The idea of CL is to train the *easy* samples before the *hard* samples. Loosely, we define an easy sample as a frame where the ground-truth chord quality is a *base* chord quality, and a hard sample as a frame where the ground-truth chord quality is an *extended* chord quality. We want to ensure that for each extended chord quality, the system first sufficiently learns its corresponding base chord quality. A critical observation in Figure 1 is that each extended chord quality is rarer than its corresponding base chord quality. Based on this observation, we propose a CL reweighting scheme that gradually converges from the training distribution to a balanced chord quality distribution, similar to the scheme proposed by Wang *et al.* in [25]. The proposed scheme enables the model to put emphasis on the frequently-labelled chord qualities at the beginning of training and put increasingly more emphasis on the rare chord qualities as training converges. Since the pitch-

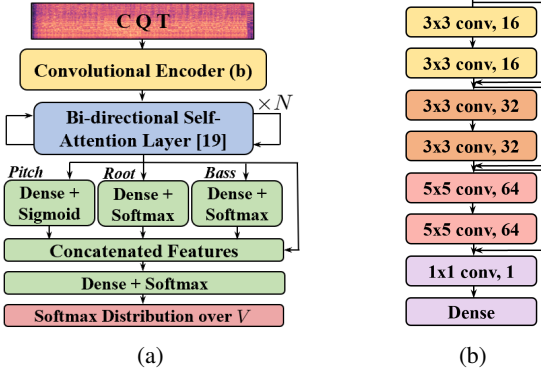


Figure 3: Proposed CT architecture. (a) outlines the end-to-end architecture and (b) outlines the layers of the proposed convolutional encoder.

shifting data augmentation eliminates the root bias in the training data, then balancing the chord-quality distribution also balances the chord-type distribution during training.

The proposed scheme runs for E epochs. At epoch $e = 1$, the target distribution is the training distribution (*i.e.* an imbalanced chord quality distribution). As e approaches E , the system gradually modifies the target chord quality distribution to be more balanced; this is achieved by reweighting samples by placing higher weights on the frames where the ground-truth chord quality is sparsely-labelled and lower weights on the frames where the ground-truth chord quality is frequently-labelled. At epoch $e = E$, the chord quality distribution is balanced.

Let $Q' = Q \cup \{N, X\}$. Let C_q be the number of frames in the training set with chord quality $q \in Q'$ and let $q_{min} \in Q'$ be the chord quality with the least number of frames in the training set. We define the chord quality training distribution D_{train} by $D_{train,q} = \frac{C_q}{C_{q_{min}}}$ for all $q \in Q'$.

Let D_e be the target chord quality distribution at epoch e . Then $D_1 = D_{train}$. During model training, the target chord quality distribution gradually transfers to a balanced distribution with the following function:

$$D_{e,q} = (D_{train,q})^{g(e)} \quad \forall q \in Q' \quad (1)$$

where e is the epoch number and $g(e)$ is the curriculum scheduler function. The scheduler function is a monotonically decreasing function from 1 to 0 that sets the pace of the curriculum. We experiment with three scheduler functions (visualized in Figure 4): $g(e) = 0$ (baseline, fixed balanced chord quality distribution), $g(e) = 1 - \frac{e-1}{E-1}$ (linear schedule), and $g(e) = \phi^e - \phi^E$ (convex schedule), where ϕ is a hyperparameter. Observe that for all three scheduler functions, $g(E) = 0$ and thus $D_{E,q} = 1$ for all q ; *i.e.* the target chord quality distribution is balanced.

At epoch e , to facilitate training with target chord quality distribution D_e , we reweight the samples such that for chord class $i \in V$ having chord quality $q \in Q'$, the class weight w_i assigned to i in L_{label} is defined by:

$$w_i = D_{e,q} / D_{train,q} \quad (2)$$

As the training set chord quality distribution is extremely

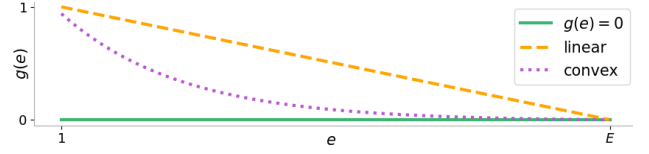


Figure 4: Proposed curriculum scheduler functions.

imbalanced, the variation in the magnitude of the weights w_i across the different chord classes $i \in V$ is extremely large at epoch $e = E$. We hypothesize that this may cause the temporal smoothness of the output predictions to be impaired. Therefore, we train an additional CRF decoder on top of the output logits to smooth the output predictions, in the same way as [8].

The proposed CL scheme differs from [25] in three critical ways. First, the training set statistics are used for computing the reweighting terms w_i , whereas [25] uses batch statistics. Using batch statistics is suboptimal for ACR since batches typically consist of a small number of sequences, and the frames of each sequence are highly interdependent. Therefore, we believe that the training statistics are more representative of the true chord quality imbalance. Second, in [25], the frequently-labelled samples are *down-sampled* by setting some samples to have weight 0 and the remaining to have weight 1. Down-sampling is ill-advised for ACR as this would disrupt the temporal coherence of the training sequence. Therefore, the proposed scheme instead employs a fully reweighted approach so that the continuity of the training sequences are preserved. Third, the proposed scheme employs a CRF decoder to smooth the output predictions at model convergence.

5. EXPERIMENTS

5.1 Model Evaluation

To compare the proposed CT model with previous methods, evaluation is conducted using `mir_eval` [30]. We obtain Weighted Chord Symbol Recall (WCSR) scores for: Root, Thirds, Triads, Sevenths, Tetrads, Maj-Min, and MIREX. We average the results of each metric across the folds, as is done in [19]. For the methods addressing the imbalanced class-learning problem, we utilize two evaluation metrics proposed in [11]: the mean frame-wise accuracy ($\text{acc}_{\text{frame}}$) and mean class-wise accuracy ($\text{acc}_{\text{class}}$) over V . $\text{acc}_{\text{frame}}$ is defined by:

$$\text{acc}_{\text{frame}} = \frac{\sum_{i=1}^n C_i}{\sum_{i=1}^n F_i} \quad (3)$$

where n is the number tracks for evaluation, F_i is the number of frames in track i , and C_i is the number of correctly-predicted frames in track i over vocabulary V . $\text{acc}_{\text{class}}$ is defined by:

$$\text{acc}_{\text{class}} = \frac{1}{|V|} \sum_{v \in V} \frac{\sum_{i=1}^n C_i^v}{\sum_{i=1}^n F_i^v} \quad (4)$$

where F_i^v is the number of frames in track i with ground-truth chord label v and C_i^v is the number of frames in track

i that are correctly predicted as v . Further, we define an additional metric termed the mean quality-wise accuracy ($\text{acc}_{\text{quality}}$) over V to be used in the CL experiments. We define $\text{acc}_{\text{quality}}$ by:

$$\text{acc}_{\text{quality}} = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{i=1}^n C_i^q}{\sum_{i=1}^n F_i^q} \quad (5)$$

where F_i^q is the number of frames in track i with ground-truth chord quality q and C_i^q is the number of frames in track i that are correctly predicted as having ground-truth chord quality q . As previously outlined in [11, 14], when addressing the imbalanced class-learning problem within the task of large vocabulary ACR, we want to maximize the $\text{acc}_{\text{class}}$ while still maintaining the $\text{acc}_{\text{frame}}$.

5.2 Implementation Details

Using `librosa` [31], audio is transformed into a log-power constant-Q spectrogram spanning 6 octaves with 36 bins per octave. The sample rate is 44100 Hz, and the hop size is 4096. We tune the hyperparameters of the bi-directional self-attention layers (optimized to BRIM): number of self-attention layers $N = 6$, number of self-attention heads $n_h = 8$, hidden dimension $d = 512$, and dropout probability $p = 0$. The CT model is trained using the Adam optimizer [32] with initial learning rate 1e-4. The learning rate is reduced by a factor of 10 when the validation L_{label} loss does not improve after 10 consecutive epochs. Model training terminates when the validation L_{label} loss does not improve after 20 consecutive epochs. We save the model weights with the lowest validation L_{label} loss for evaluation. In each epoch, a contiguous 248-frame segment is randomly sampled from each training track, and a mini-batch consists of 32 such segments. Structured training is conducted in the same way as [9], with the exception that we set $\gamma = 7$. We set the L_{label} class weights to $w_i = 1$ for all $i \in V$.

For the CL experiments, we set $E = 90$ and $\phi = 0.95$. Since only the *chord qualities* of the 5-fold split are stratified, the imbalance in the root distribution across the folds may cause $\text{acc}_{\text{class}}$ to be an unreliable validation metric. Thus, we instead use $\text{acc}_{\text{quality}}$. The training details remain the same with a few exceptions. Namely, the learning rate is reduced by a factor of 10 when the validation $\text{acc}_{\text{quality}}$ has not improved for 10 consecutive epochs. For the linear and convex scheduler functions, we save the model weights at convergence (epoch $e = E$) for evaluation. For the baseline scheduler function $g(e) = 0$, we save the model weights with the highest validation $\text{acc}_{\text{quality}}$ for evaluation. We train the CRF using Adam with a learning rate of 1e-2. We terminate training the CRF once the validation $\text{acc}_{\text{quality}}$ stops improving. Our implementation is available at <https://github.com/RLuke22/curriculum-learning-acr>.

5.3 Methods under Comparison

We compare our proposed CT architecture with CR2S+A [9], BTC [19] and the best-performing model of [11],

Metric	CT	CT _{-RE}	CT _{-RES}	CT _{-RE,S,C} (BTC) [19]	CR2S+A [9]	CRNN* [11]
Root	0.838	0.836	0.831	0.829	0.821	0.837
Thirds	0.809	0.807	0.802	0.798	0.784	0.803
Triads	0.767	0.764	0.759	0.754	0.742	0.759
Sevenths	0.714	0.711	0.709	0.700	0.677	0.694
Tetrads	0.650	0.646	0.644	0.638	0.615	0.630
Maj-Min	0.826	0.823	0.819	0.813	0.802	0.822
MIREX	0.832	0.828	0.825	0.820	0.803	0.812

Table 1: WCSR scores averaged across 5 folds. _{-RE} denotes removal of relative positional encoding. _{-S} denotes removal of structured training. _{-C} denotes removal of the convolutional encoder. *operates over a larger chord-vocabulary V' consisting of 301 chord types [11].

Method	$\text{acc}_{\text{frame}}$	$\text{acc}_{\text{class}}$
CT	0.677	0.347
CT+CL _{Baseline}	0.647	0.427
CT+CL _{Linear}	0.658	0.439
CT+CL _{Convex}	0.657	0.449
CT+EC [14]	0.650	0.379
CRNN _{0.5,10} [11]	0.630	0.321

Table 2: $\text{acc}_{\text{frame}}$ and $\text{acc}_{\text{class}}$ scores over V for all data-balancing methods using stratified split over BRIM.

which we call CRNN. As BRIM is significantly larger than the dataset used to train the BTC model, the BTC model is re-trained on BRIM as described in Section 5.2. We evaluate these models using the WCSR metrics and the same 5-fold split of BRIM that is used in [9, 11, 26].

All CL experiments are trained and evaluated with the CT model. We denote the models with convex and linear scheduler functions as CT+CL_{Convex} and CT+CL_{Linear}, respectively. The baseline model with scheduler function $g(e) = 0$ is denoted CT+CL_{Baseline}. We also evaluate the even-chance training scheme proposed in [14], which we call CT+EC. Specifically, we adjust the CT model training procedure so that each chord type $v \in V$ has an even chance of being selected at the beginning of each training segment. $\text{acc}_{\text{quality}}$ is used as the validation metric for the CT+EC model. Further, we evaluate the reweighting scheme of [11] on the CRNN model. We experiment with the best-reported reweighting configuration $(\gamma, w_{\text{max}}) = (0.5, 10.0)$, which we call CRNN_{0.5,10}. For evaluation we use the $\text{acc}_{\text{frame}}$ and $\text{acc}_{\text{class}}$ metrics using the stratified 5-fold split of BRIM.

5.4 Results

The WCSR scores for all models considered are shown in Table 1. Table 1 also includes an ablation study that outlines the performance degradation with the removal of each novel component in the CT architecture; *i.e.* the convolutional encoder (C), structured training (S), and relative positional encoding (RE). Note that the CT architecture without all three novel components (and with the inclusion of global z-normalization) is equivalent to BTC [19]. Table 1 shows that the CT model outperforms existing ACR systems across all WCSR metrics. Further, the ablation study indicates that each novel component offers a gradual, yet consistent improvement to the CT model.

Table 2 shows the results of the methods that address

Method	Chord Quality Accuracy													
	maj	min	7	min7	maj7	sus4	maj6	min6	sus2	dim	aug	hdim7	dim7	minmaj7
CT	0.801	0.637	0.518	0.576	0.570	0.277	0.102	0.134	0.034	0.365	0.236	0.357	0.031	0.031
CT+CL _{Convex}	0.735	0.626	0.537	0.595	0.634	0.406	0.275	0.288	0.261	0.395	0.517	0.476	0.181	0.229
CT+CL _{Baseline}	0.727	0.618	0.512	0.581	0.604	0.411	0.250	0.264	0.235	0.386	0.460	0.450	0.209	0.151

Table 3: Chord quality accuracies of various CT models over BRIM at evaluation. Chord quality accuracy is defined as the proportion of frames where the predicted chord quality matches the ground-truth chord quality.

imbalanced class-learning including a baseline CT model (*i.e.* no reweighting). Unsurprisingly, the CT model performs the best in the $\text{acc}_{\text{frame}}$ metric. This is consistent with previous works that have shown that optimizing the class-wise accuracy typically harms the frame-wise accuracy [11, 14]. The best-performing CL model CT+CL_{Convex} provides substantial improvement (10.2%) in the class-wise accuracy, with only a modest degradation (2.0%) in the frame-wise accuracy. This indicates that the CL scheme considerably suppresses the learning bias in the model induced by the imbalanced chord quality distribution without significantly impairing the performance of the frequently-labelled classes. Moreover, both CL configurations CT+CL_{Convex} and CT+CL_{Linear} offer improvements in the $\text{acc}_{\text{frame}}$ and $\text{acc}_{\text{class}}$ metrics over the baseline CT+CL_{Baseline}. This indicates that by having the model sufficiently learn the base chord qualities prior to the corresponding extended chord qualities, the model better generalizes on both the frequently-labelled and sparsely-labelled chord qualities. This is further confirmed in Table 3, which shows that CT+CL_{Convex} outperforms CT+CL_{Baseline} in chord quality accuracy on every chord quality except for *sus4* and *dim7*.

In Figure 5, we evaluate the chord quality accuracies of the CT+CL_{Convex} model at different epochs in the curriculum. Note that the *dim* base chord quality accuracy (in dark green) improves substantially from epochs 1 to 10, followed by an improvement in the *hdim7* (in red) and *dim7* (in purple) extended chord qualities from epochs 10 to 20 and 10 to 40, respectively. Similar trends can be observed for the *maj* and *min* base chord qualities. This indicates that sufficient learning of the base chord qualities leads to performance improvements in the corresponding extended chord qualities. We hypothesize that the convex scheduler function outperforms the linear scheduler function in class-wise accuracy because the extreme imbalance in the chord quality distribution warrants a faster curriculum pace at the beginning of training. As shown in Table 2, the proposed CT+CL_{Convex} model convincingly outperforms previous methods in the $\text{acc}_{\text{class}}$ metric. Note that the CRNN_{0.5,10} results in Table 2 differ from the results reported in [11] as we run CRNN_{0.5,10} over a different chord vocabulary V than the one used in [11].

To validate the inclusion of the CRF decoder in the CL scheme, we count the number of chord changes in BRIM predicted by the CT model, the CT+CL_{Convex} model, and the CT+CL_{Convex} model without the CRF (denoted CT+CL_{Convex}-CRF). Table 4 shows that the model weights at CL convergence disrupt the smoothness of the output chord-label predictions, as evidenced by the substan-

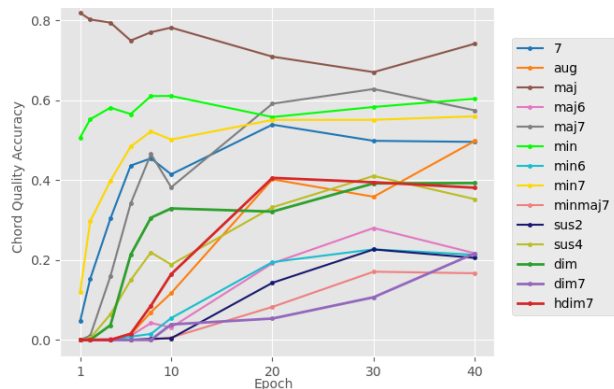


Figure 5: Chord quality accuracies of CT+CL_{Convex} evaluated at different points along the curriculum.

Method	Chord Changes
CT	170,287
CT+CL _{Convex}	123,350
CT+CL _{Convex} -CRF	235,885
Ground-Truth	122,231

Table 4: Predicted chord changes over BRIM.

tially larger number of predicted chord changes by the CT+CL_{Convex}-CRF model.

6. CONCLUSION

We propose a convolutional Transformer architecture for ACR and a novel CL reweighting scheme to handle the imbalanced chord quality distribution. The proposed scheme exploits the hierarchical relationships between chord qualities by gradually converging from the initial distribution to a balanced chord quality distribution. The proposed curriculum outperforms existing methods and non-CL baselines in improving the classification performance of rare chord qualities without significantly degrading the classification performance of the frequently-labelled chord qualities. Although the proposed method considerably diminishes the model-learning bias induced by the imbalanced chord quality distribution, the model still generally favors the frequently-labelled chord qualities. We believe this is primarily an issue of data scarcity. Therefore, a promising future direction to handle the imbalanced class-learning problem for ACR is to generate more annotated data either synthetically or by leveraging the vast amount of publically-available unannotated audio tracks.

7. ACKNOWLEDGEMENTS

We would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for their generous funding and support of this work. We would also like to thank Brian McFee for providing us with the BRIM dataset, Quinton Yong for his helpful insights on the stratification algorithm, and the anonymous reviewers for their insightful and constructive feedback.

8. REFERENCES

- [1] T. Fujishima, “Real-time chord recognition of musical sound: A system using common lisp music,” in *Proceedings of the International Computer Music Conference (ICMC)*, 1999, pp. 464–467.
- [2] C. Weiß, F. Brand, and M. Müller, “Mid-level chord transition features for musical style analysis,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 341–345.
- [3] T. Carsault, J. Nika, and P. Esling, “Using musical relationships between chord labels in automatic chord extraction tasks,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 18–25.
- [4] S. Sigtia, N. Boulanger-Lewandowski, and S. Dixon, “Audio chord recognition with a hybrid recurrent neural network,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 127–133.
- [5] X. Zhou and A. Lerch, “Chord detection using deep learning,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 52–58.
- [6] F. Korzeniowski and G. Widmer, “Feature learning for chord recognition: The deep chroma extractor,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 37–43.
- [7] E. Humphrey and J. P. Bello, “Rethinking automatic chord recognition with convolutional neural networks,” in *11th International Conference on Machine Learning and Applications (ICMLA)*, 2012, pp. 357–362.
- [8] F. Korzeniowski and G. Widmer, “A fully convolutional deep auditory model for musical chord recognition,” in *26th IEEE International Workshop on Machine Learning for Signal Processing*, 2016, pp. 1–6.
- [9] B. McFee and J. P. Bello, “Structured training for large-vocabulary chord recognition,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 188–194.
- [10] Y. Wu and W. Li, “Automatic audio chord recognition with midi-trained deep feature and BLSTM-CRF sequence decoding model,” *IEEE ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 355–366, 2019.
- [11] J. Jiang, K. Chen, W. Li, and G. Xia, “Large-vocabulary chord transcription via chord structure decomposition,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 644–651.
- [12] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, “Audio chord recognition with recurrent neural networks,” in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 335–340.
- [13] J. Deng and Y. Kwok, “A hybrid gaussian-HMM-deep learning approach for automatic chord estimation with very large vocabulary,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 812–818.
- [14] —, “Large vocabulary automatic chord estimation with an even chance training scheme,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017, pp. 531–536.
- [15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [16] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” in *Proceedings of the 57th Conference of the Association for Computational Linguistics*, 2019, pp. 2978–2988.
- [17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.
- [18] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*.
- [19] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, “A bi-directional transformer for musical chord recognition,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 620–627.
- [20] T. Chen and L. Su, “Harmony transformer: Incorporating chord segmentation into harmony recognition,” in

- Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 259–267.
- [21] —, “Attend to chords: Improving harmonic analysis of symbolic music using transformer-based models,” *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, pp. 1–13, 2021.
- [22] H. He and E. Garcia, “Learning from imbalanced data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [23] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009, pp. 41–48.
- [24] M. McVicar, Y. Ni, T. D. Bie, and R. Santos-Rodriguez, “Leveraging noisy online databases for use in chord recognition,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 639–644.
- [25] Y. Wang, W. Gan, J. Yang, W. Wu, and J. Yan, “Dynamic curriculum learning for imbalanced data classification,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 5016–5025.
- [26] E. Humphrey and J. P. Bello, “Four timely insights on automatic chord estimation,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 673–679.
- [27] B. McFee, E. Humphrey, and J. P. Bello, “A software framework for musical data augmentation,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 248–254.
- [28] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on Machine Learning, (ICML)*, 2015, pp. 448–456.
- [29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [30] C. Raffel, B. McFee, E. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. Ellis, “Mir_eval: A transparent implementation of common MIR metrics,” in *Proceedings of the 15th International Society for Music Information Retrieval Conference, (ISMIR)*, 2014, pp. 367–372.
- [31] B. McFee, V. Lostanlen, M. McVicar, A. Metsai, S. Balke, C. Thomé, C. Raffel, D. Lee, F. Zalkow, K. Lee, O. Nieto, J. Mason, D. Ellis, R. Yamamoto, E. Battenberg, R. Bittner, K. Choi, J. Moore, Z. Wei, S. Seyfarth, P. Friesch, F. Stöter, D. Hereñú, T. Kim, M. Vollrath, and A. Weiss, “librosa/librosa: 0.7.1,” Oct. 2019. [Online]. Available: <https://doi.org/10.5281/zenodo.3478579>
- [32] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, (ICLR)*, 2015.

DEEP EMBEDDINGS AND SECTION FUSION IMPROVE MUSIC SEGMENTATION

Justin Salamon Oriol Nieto Nicholas J. Bryan

Adobe Research, San Francisco, CA, USA

salamon@adobe.com

ABSTRACT

Music segmentation algorithms identify the structure of a music recording by automatically dividing it into sections and determining which sections repeat and when. Since the desired granularity of the sections may vary by application, *multi-level* segmentation produces several levels of segmentation ordered by granularity from one section (the whole song) up to N unique sections, and has proven to be a challenging MIR task. In this work we propose a multi-level segmentation method that leverages deep audio embeddings learned via other tasks. Our approach builds on an existing multi-level segmentation algorithm, replacing manually engineered features with deep embeddings learned through audio classification problems where data are abundant. Additionally, we propose a novel section fusion algorithm that leverages the multi-level segmentation to consolidate short segments at each level in a way that is consistent with the segmentations at lower levels. Through a series of experiments we show that replacing handcrafted features with deep embeddings can lead to significant improvements in multi-level music segmentation performance, and that section fusion further improves the results by cleaning up spurious short sections. We compare our approach to two strong baselines and show that it yields state-of-the-art results.

1. INTRODUCTION

Audio-based music structure analysis, also known as music segmentation, is one of the most widely studied and challenging tasks in Music Information Retrieval [1]. The goal of this task is to obtain a series of non-overlapping sections (segments) defined by a set of temporal boundaries, and to identify and label which sections are repetitions of each other. Automatic segmentation could enable efficient intra-track navigation [2], assisted music creation [3], and section-based music retrieval and recommendation [4], to name some applications.

A key challenge in music segmentation is to capture the different possible levels of temporal granularity with which a song can be segmented. From an application perspective,

a *multi-level* segmentation¹ that produces multiple segmentations ranging from coarse (e.g., 1-3 unique sections and their repetitions) to granular (e.g., 8-12) would allow application designers and/or end users to choose the level(s) of segmentation that best fits their needs. To facilitate this, datasets such as SALAMI [5] and SPAM [6] have been manually annotated with multiple segmentation levels based on length (e.g., long-scale sections, short-term motives) or functional role (e.g., “sax solo,” “outro”). Metrics to evaluate multi-level segmentation have also been proposed recently [7,8] and adopted by the community [9].

Most segmentation methods yield just one level. An early approach identified sharp differences in time series of audio features related to timbre and harmony by running a checkerboard kernel along the diagonal of a self-similarity matrix [10]. More sophisticated handcrafted features were later proposed, yielding superior boundary detection [11]. Currently, the best boundary detection is obtained with deep learning models, such as a deep convolutional neural network (CNN) [12] or deep metric learning which yields an effective feature space for boundary detection [13]. Multi-level approaches appeared more recently, and just a handful have been proposed to date. McFee and Ellis apply spectral clustering to a self-similarity matrix obtained via a simple combination of DSP features [14], an approach later enhanced by Tralie and McFee by adding harmonic embeddings from a convolutional-recurrent neural network and using Similarity Network Fusion (SNF) to combine features [9]. Supervised approaches include ordinal linear discriminant analysis [15] and a CNN that outputs two segmentation levels [16].

We propose an approach² that builds on the work of McFee and Ellis [14]. We summarize our contributions as follows: (1) we propose replacing or augmenting the handcrafted features with deep audio embeddings that can robustly capture various similarity and repetition cues; (2) we introduce a multi-level section fusion algorithm that leverages the different segmentation levels, consolidating short sections to produce a cleaner and more consistent segmentation across levels; (3) through a series of experiments and qualitative analysis, we demonstrate the effectiveness of each of our contributions. We compare our approach to strong baselines and show that it produces state-of-the-art results for multi-level music structure segmentation.



© J. Salamon, O. Nieto, N. J. Bryan. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Salamon, O. Nieto, N. J. Bryan, “Deep Embeddings and Section Fusion Improve Music Segmentation”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ Also known as hierarchical structure segmentation, but since the levels do not strictly form a hierarchy, we use multi-level segmentation.

² Code: github.com/justinsalamon/musicseg_deepemb

2. LAPLACIAN STRUCTURAL DECOMPOSITION

We start with an overview of Laplacian Structural Decomposition (LSD) [14], which forms the basis for our method.

A *recurrence matrix* captures the similarity between feature frames of a track, and can expose song structure [11]. It is a binary, squared, symmetrical matrix R such that $R_{ij} = 1$ if frames i and j are similar—for a specific metric, e.g., cosine distance—and $R_{ij} = 0$ otherwise. McFee and Ellis [14] treat the recurrence matrix as an unweighted, undirected graph, where each frame is a vertex and 1’s in the recurrence matrix represent edges. Then they apply *spectral clustering* [17] (unrelated to audio spectrograms), yielding a per-frame cluster assignment. Sections are derived by grouping frames by their cluster assignment. Sections with the same cluster ID represent repetitions.

In the original approach, R is obtained by combining two recurrence matrices obtained from audio features: R^{loc} computed from mel-frequency cepstral coefficients (MFCC) to identify local similarity between consecutive frames, and R^{rep} computed from Constant-Q transform (CQT) features to capture repetition across the entire track. The goal is to detect sudden sharp changes in timbre with R^{loc} , while capturing long-term harmonic repetition with R^{rep} . These matrices are combined via a weighted sum controlled by a hyper-parameter $\mu \in [0, 1]$, which can be set manually or automatically [14]:

$$R = \mu R^{\text{rep}} + (1 - \mu) R^{\text{loc}} \quad (1)$$

The number of unique sections produced by the segmentation is equal to the number of clusters N used for spectral clustering. By clustering with increasing $N = 1 \dots M$ we obtain a multi-level segmentation: the higher M is, the finer the resulting segmentation [18]. The clustering uses an eigenvalue decomposition, such that for a given M the data are projected onto the first M eigenvectors (ordered by their eigenvalues) of the symmetrical normalized Laplacian of R and then clustered. The key takeaway is that the same eigenvectors are reused for increasing M (each time adding one more), meaning cluster assignments at different levels are related. This property is essential for the multi-level section fusion algorithm we present later.

3. DEEP AUDIO EMBEDDINGS

We replace or augment the handcrafted features in LSD with deep audio embeddings learned via other tasks, making this a transfer learning approach. We propose to: (1) replace the MFCC features with deep embeddings learned via Few-Shot Learning (FSL) [19], and (2) augment the CQT features with deep embeddings learned via a state-of-the-art music auto-tagging model designed to capture music similarity across genre, mood, tempo, and era [20].

3.1 Few-shot Learning Embeddings

The purpose of the MFCC features used in the LSD method is to capture local (short-term) timbre similarity, with the goal of identifying sharp transitions as potential

boundary locations. However, MFCC have been shown to be sensitive to noise [21], and so we hypothesize that an audio feature that captures short-term timbre similarity more robustly could lead to better boundary detection.

To this end, we employ the Few-Shot Sound Event Detection model recently proposed by Wang et al. [22]. Few-shot learning (FSL) is an area of machine learning which aims to train models that are able, once trained, to robustly recognize a new class given a handful of examples of the new class at inference time [19]. Wang et al. showed that Prototypical Networks [19], a metric-based approach originally proposed for FSL on images, can be successfully applied to the audio domain given the right adaptations. Importantly, Prototypical Networks do not require fine-tuning or retraining. Rather, they are used to embed audio such that perceptually similar sounds are also close in the embedding space, as shown by Wang et al. [22, 23]. As such, these embeddings, which are computed from a 0.5 second window, can be viewed as a general-purpose, short-term, timbre similarity feature. Wang et al. focused on the task of sound event detection (SED), training and evaluating the model on few-shot word recognition via an annotated speech corpus. We refer the reader to this study for further details about the model architecture and training [22].

As this model was trained on hundreds of thousands of audio samples, we hypothesize that the resulting audio embedding will be more robust compared to MFCC for capturing short-term timbre similarity. We replace MFCC with these embeddings, henceforth FSL, to compute R^{loc} . We use the model trained by Wang et. al, courtesy of the authors. Even though the model was trained on speech audio data, preliminary experiments indicated it captures timbre similarity for music too—a form of transfer learning.

3.2 Music Similarity Embeddings for Repetition

In LSD the repetition recurrence matrix R^{rep} is obtained using Constant-Q transform (CQT) features [24] computed from the audio signal after applying Harmonic-Percussive Source Separation (HPSS) [25] to enhance the harmonic components of the audio signal. Still, not all songs exhibit harmonic repetition, e.g., an EDM song may exhibit repetition of timbre (presence/absence of a beat, a high- or low-pass filter that is applied in specific sections, etc.). Many Western popular music songs use the same harmonic progression for both the verse and chorus, with only the instrumentation and lyrics indicating a section change.

We propose to use, in addition to CQT, deep audio embeddings that can capture other complementary music qualities that may be indicative of repetition, such as instrumentation, tempo, and mode. To achieve this, we leverage the deep music auto-tagger presented by Lee et al. [20]. In their work, the authors contrast classification and metric learning for training a deep music embedding that can be used for similarity-based music retrieval. Of the approaches compared, disentangled multi-task classification yielded an embedding that gave the best music retrieval results, in addition to producing state-of-the-art results for music auto-tagging. Here, *disentangled* means that the em-

bedding space is divided into sub-spaces that capture different dimensions of music similarity. The full embedding of size 256 is divided into four disjoint subspaces, each of size 64, where each subspace captures similarity along one musical dimension: genre, mood, tempo, and era. We refer the reader to Lee et al. [20, 26] for further details about the model architecture and optimization.

We hypothesize that this embedding, which is obtained from a 3-second context window and was trained on the Million Song Dataset [27], captures musical qualities that can be complementary to those captured by the CQT: genre is often a reasonable proxy for instrumentation; mood can be a proxy for tonality and dynamics; tempo is an important low-level quality in itself; and era, in addition to being related to genre, can be indicative of mixing and mastering effects. Combined, the full embedding, henceforth referred to as DEEPSIM, may surface repetitions along dimensions that are not captured by the CQT.

3.3 Fusing Similarity Matrices

In Section 2 we explained that the LSD method uses two matrices: R^{loc} (from MFCC) and R^{rep} (from CQT features). Now, we replace MFCC with FSL features for computing R^{loc} . For computing R^{rep} we do not replace the CQT features but rather combine them with the DEEPSIM embeddings, since they are potentially complementary. We do this via another weighted sum controlled by a hyperparameter $\gamma \in [0, 1]$, leading to the following equation:

$$R = \mu (\gamma R^{\text{DEEPSIM}} + (1 - \gamma) R^{\text{CQT}}) + (1 - \mu) R^{\text{FSL}} \quad (2)$$

All three matrices are normalized prior to being combined to ensure their values are in the same $[0, 1]$ range. This simplistic approach to feature fusion may not be optimal, and indeed more advanced fusion techniques such as Similarity Network Fusion [9] have been proposed. However, this approach has the advantage of allowing us to easily and clearly study the relative importance of the features we are proposing to use: μ controls the relative importance of local versus repetition similarity, while γ controls the relative importance of CQT versus DEEPSIM features for repetition similarity. We set our initial parameterization to $\mu = 0.5, \gamma = 0.5$, meaning we give equal weight to local similarity obtained via FSL features and repetition similarity, which is given by the simple average of the R^{CQT} and R^{DEEPSIM} matrices. Later on we will explore the impact of varying these parameters on two different datasets to gain insight about feature relevance for different data.

4. MULTI-LEVEL SECTION FUSION

In Section 2 we explained how segmentation is achieved by clustering each frame of the audio signal. This assigns each frame a cluster ID, and then consecutive frames with the same cluster ID are grouped to form sections. This process can sometimes result in a small number of consecutive frames having a different cluster ID to those around them, leading to very short sections. These often do not represent actual sections in the song, and even when they do, they may not be helpful to the end user or application.

The LSD method attempts to alleviate this issue via smoothing: it applies median filtering to R^{rep} to enhance diagonals in the matrix before it is combined with R^{loc} , and also applies median smoothing to the vectors obtained via spectral clustering. Even with this smoothing, we found that our approach (and LSD) can still produce spurious short sections. Smoothing more aggressively would deteriorate the temporal accuracy of the boundaries between sections, and is thus undesirable. What is more, it is unlikely for there to be a single “best” minimal section duration for all use cases. Rather, the appropriate lower bound on section duration will depend on the application.

We address this challenge in a way that allows the user to define their desired minimal section duration. Given the desired minimal duration in seconds, we now need to remove sections whose duration is below this value, henceforth “short sections”, by fusing them with the previous section, the next section, or both. But, how do we determine which section(s) to fuse with? We propose an algorithm that leverages multi-level segmentation to solve this.

4.1 Multi-level Section Fusion Algorithm

Our method leverages two heuristics: (1) section IDs should mostly be consistent with overlapping sections at lower segmentation levels, since we are reusing the same eigenvectors for clustering (cf. Section 2); (2) section boundaries should be mostly consistent across segmentation levels, and thus a boundary that overlaps with boundaries at lower levels of the hierarchy is more likely to be a real boundary compared to one that does not.

The algorithm works as follows: say we need to fuse a short section at level n . If the section is the first or last of the song, we merge it to the next or previous section respectively, as that is our only option. If the section is in the middle of the song *and* both the previous and next sections have the same ID, we merge all three together. These three simple scenarios are depicted in Figure 1a. Otherwise, we need to determine whether to merge the short section with the previous *or* the next section at level n . So, we look one level down in the hierarchy ($n - 1$) and find the section that overlaps the most with our short section. If the ID of the overlapping section at level $n - 1$ matches the ID of either the previous or next sections at level n , we merge the short section with the matching section (Figure 1b). If the overlapping ID at level $n - 1$ does not match the ID of neither the previous nor the next section at level n , we go down to level $n - 2$ and try again (Figure 1c), and so on until we find an overlapping section whose ID matches the ID of either the previous or next section at level n .

If we reach the bottom level and still do not find a match, we turn to our second multi-level cue: the boundaries (Figure 1d). Our short section at level n has two boundaries, let’s call them “start” and “end.” For each of the two we count how many boundaries they overlap with at all lower segmentation levels, where we consider boundaries at different levels to overlap if they are within one second of each other. Whichever of the two (start or end) overlaps with the most boundaries at lower levels is more

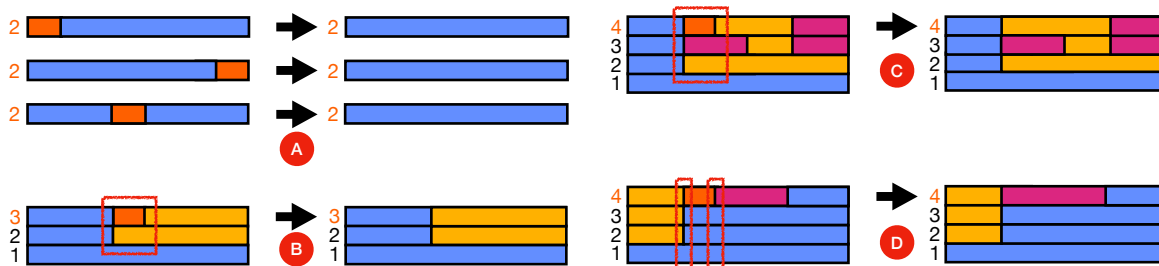


Figure 1: Multi-level section fusion algorithm: in all plots the short orange section needs to be fused. The number to the left of each segmentation represents its level, with the number in orange indicating the level being cleaned: (A) three simple fusion scenarios, (B) fusion via multi-level segmentation IDs: level $n - 1$ gives us the answer, (C) same as (B) but this time level $n - 2$ gives us the answer, (D) fusion via multi-level segmentation boundaries: the start boundary is consistent with boundaries at levels $n - 1$ and $n - 2$, whereas the end boundary is not consistent with any boundaries at lower levels.

likely to be a real boundary, so we keep that boundary and remove the other by fusing the short section to the adjacent section separated by the “losing” boundary.

We repeat the entire process until there are no short sections left at level n . We iterate over the sections in a double loop: the outer loop iterates over section IDs, from the highest to the lowest. The section ID corresponds to the eigenvector to which the section was clustered. By iterating in this way, we are more likely to keep sections with lower IDs, which in turn are more likely to appear at lower levels of the hierarchy. Within each section ID, our inner loop iterates over the sections from shortest to longest. We have found that this approach leads to more coherent section fusion across the entire hierarchy.

5. EXPERIMENTAL DESIGN

5.1 Datasets

We experiment with two datasets: Harmonix [28] and SALAMI [5], which are the largest datasets published to date with structural segmentation annotations. They are notably different in terms of audio content and annotations.

The Harmonix set contains 912 tracks of Western popular music that have been manually annotated with functional sections (“intro,” “chorus,” “solo,” etc.). As such, the annotations are not multi-level (i.e., they are “flat”) and represent a single segmentation level with one annotator per track. This dataset was compiled by the video game company Harmonix with the goal of incorporating music segmentation into some of its music games (e.g., Rock Band, Guitar Hero). This makes it highly relevant for evaluating algorithms that will be applied in real-world applications focusing on Western music.

The SALAMI set is comprised of 1,355 tracks spanning a wide range of musical genres including classical, jazz, non-Western music, live performances, etc. Each track is manually annotated with three levels: (1) functional segments representing sections such as “guitar solo,” “verse,” etc.; (2) larger-scale segments representing longer structural sequences, annotated with upper-case letters, e.g., A, B, C’; (3) small-scale segments capturing shorter time scales in the song that may include melody lines or motifs, annotated with lower-case letters, e.g., a, b’, c. To be com-

parable to previous work [9], we evaluate against the large-(2) and small-scale (3) annotations, limiting our test set to tracks that have two or more annotations (884 tracks). The remainder (471) are used for hyper-parameter tuning.

5.2 Metrics

The L-Measure (L-M) is the preferred metric for evaluating multi-level segmentations [7]. It treats music segmentation as a similarity ranking problem, capturing both boundary alignment (otherwise evaluated as a binary classification problem) and segment labeling (otherwise evaluated as a clustering problem). To compute L-M, we divide the reference annotation into time points (frames), and compare each point t against all other time points. If t is closer to point u than point v when considering all segmentation levels, we represent it as a triplet (t, u, v) . We repeat the same process for the algorithm’s estimate segmentation. We then define the L-Precision (L-P) as the fraction of estimate triplets that match reference triplets, the L-Recall (L-R) as the fraction of reference triplets that match estimate triplets, and L-M as their harmonic mean.

Our method and the baselines we compare it against produce much deeper segmentations than the two levels annotated in SALAMI or the single level annotated in Harmonix, meaning the L-Precision may not be sufficiently reliable [7, 9]. Conversely, the L-Recall, which captures how well the structure defined in the reference is retrieved in the estimation is, in this context, a more trustworthy metric, and so we focus on it in our study. Still, for completeness we report all three L metrics (L-P, L-R, L-M).

We also report the standard metrics used to evaluate flat music segmentation: Hit Rate for boundary retrieval at 0.5 and 3 second tolerance windows, $HR_{0.5}$ and HR_3 , and the Pairwise Frame Clustering (PFC) [29] and Normalized Conditional Entropies (NCE) [30] for segment labeling. Since our application scenario assumes the preferred segmentation level will be preset by the application designer or set by the end user based on their needs, we simulate this scenario in our evaluation by computing these metrics for each track using the segmentation level that maximizes the metrics. For conciseness, we only report the aggregated harmonic mean for each of the flat metrics.

Deep Embs.	Sec. Fusion	L-P	L-R	L-M
No	No	36.70	65.77	46.82
No	Yes	38.07	66.68	47.92
Yes	No	40.91	76.56	53.01
Yes	Yes	43.50	76.47	55.01

Table 1: Ablation results on the Harmonix dataset.

5.3 Baselines

We compare our approach against two baselines: the original LSD method [14], and its improved variant that adds deep harmonic embeddings and uses Similarity Network Fusion (SNF) to combine the recurrence matrices [9]. While the latter baseline also leverages deep embeddings, they are only designed to capture harmony, unlike our multiple deep embeddings which capture a variety of musical properties. We use the LSD implementation from MSAF [6] and the SNF implementation released by the authors.³ LSD and our method use beat-aligned features, for which we use the beat tracker by Korzeniowski et al. [31] implemented in the madmom package [32], as it has been shown to yield better segmentation results [28] compared to the default beat tracker in Librosa [33]. SNF does not rely on beat tracking.

5.4 Ablations

To demonstrate the effectiveness of our contributions, we perform a systematic ablation compared to LSD: we fix $\mu = 0.5$ for LSD and $\mu = \gamma = 0.5$ for our approach, and then compare LSD, LSD + section fusion, our method (LSD + deep embeddings) without section fusion, and finally our full method (LSD + deep embeddings + section fusion). For section fusion we set the minimum section duration to 8 seconds (=4 bars at 120 bpm) as a reasonable lower bound. In a real world scenario this value could be chosen by an end user or preset by the application designer.

6. RESULTS

6.1 Ablations

In Table 1 we present the results of the ablations described above on the Harmonix dataset. We see that our deep embeddings and section fusion independently improve the baseline. Noteworthy is the dramatic increase in L-Recall due to our proposed deep embeddings. Combining the deep embeddings with section fusion improves L-Precision and thus the overall L-M. Though omitted from Table 1 for conciseness, we also confirmed that just replacing MFCC with FSL (without DEEPSIM) improves over the baseline, strengthening our hypothesis from Section 3.1.

6.2 Feature Importance

To understand the relative importance of the features used in our approach, we run a grid search over μ and γ , focusing on L-Recall (cf. Sec. 5.2). Remember that μ controls the ratio of local similarity (FSL embeddings) to repetition,

³ <https://github.com/ctralie/GraphDitty>

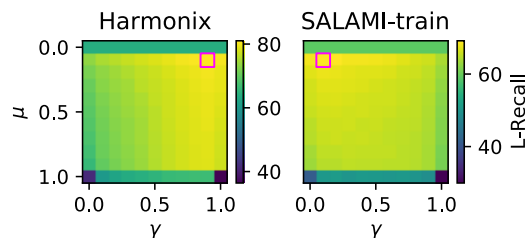


Figure 2: Grid search for μ and γ , red square is best.

while γ controls the relative contribution of the DEEPSIM embeddings versus the CQT features to repetition.

We present the results in Figure 2 for Harmonix and SALAMI, with the L-Recall maxima marked by a square. First, we note that results always worsen when $\mu = 0$ or 1, illustrating the importance of combining both local similarity and repetition matrices. While less pronounced, the same is true of γ , showing that both datasets benefit from combining DEEPSIM and CQT features for repetition.

For Harmonix, performance is maximized when $\mu = 0.1, \gamma = 0.9$: most weight goes to local repetition via FSL, with most of the remainder going to DEEPSIM features for repetition. On the other hand, for SALAMI performance is maximized when $\mu = \gamma = 0.1$, i.e., most of the weight goes to FSL with the remainder going to CQT features.

The difference in optimal parameter values per dataset warrants discussion. A small μ in both cases highlights the importance of local similarity information (FSL), regardless of music genre. On the other hand, we see the DEEPSIM embeddings are preferred when segmenting Western popular music (Harmonix), while CQT features are favored for SALAMI which is more diverse. One possible explanation is that DEEPSIM was trained on a subset of MSD that leans more heavily toward Western popular music compared to SALAMI [34]. Still, it is always beneficial to use a combination of both features.

6.3 Multi-level and Flat Results

We compare our approach against two strong baselines representing the state-of-the-art in multi-level music structure segmentation [9, 14]. We compute the baselines using the same setup reported by their authors. LSD sets μ automatically per-track in a data-driven fashion. For our approach, Deep Embeddings with section Fusion (DEF), we report results for three parameter configurations: (1) $\mu = \gamma = 0.5$, (2) optimal values obtained via grid search on SALAMI-train (μ^S, γ^S), (3) optimal values obtained via grid search on Harmonix (μ^H, γ^H)⁴.

The full multi-level segmentation results are presented in Table 2. For SALAMI, we see that the parameter values obtained by optimizing DEF over SALAMI-train generalize well to the test set, beating all other methods in terms of multi-level segmentation (L-P, L-R, L-M) and setting a new state of the art. Turning to Harmonix, we see that DEF outperforms the baseline for all three parameter configurations, setting a new state of the art for this dataset too. Most

⁴ These may be artificially inflated due to the lack of train/test splits for the Harmonix dataset.

Method	SALAMI			Harmonix		
	L-P	L-R	L-M	L-P	L-R	L-M
LSD [7]	41.89	63.60	49.77	39.05	69.33	49.63
SNF [9]	43.08	66.82	51.65	36.38	67.47	47.01
DEF _{0.5, 0.5}	42.43	64.51	50.38	43.50	76.47	55.01
DEF _{μ^S, γ^S}	43.46	67.30	52.02	42.63	75.43	54.06
DEF _{μ^H, γ^H}	41.64	66.06	50.38	43.23	81.03	56.04

Table 2: Multi-level segmentation results.

Method	SALAMI				Harmonix			
	HR _{0.5}	HR ₃	PFC	NCE	HR _{0.5}	HR ₃	PFC	NCE
LSD [7]	31.99	47.46	56.13	59.00	40.69	56.50	61.21	57.43
SNF [9]	29.17	45.59	56.73	59.98	26.57	51.42	57.38	54.86
DEF _{0.5, 0.5}	33.78	55.65	59.44	62.16	45.74	68.84	70.11	66.48
DEF _{μ^S, γ^S}	32.07	53.91	59.99	62.39	43.18	67.34	69.34	65.44
DEF _{μ^H, γ^H}	31.79	56.35	58.56	61.48	41.61	71.17	71.49	67.59

Table 3: Flat segmentation results.

sections in the “lowercase” level of SALAMI are shorter than 8 s which, given our section fusion, may explain why the improvement is moderate compared to Harmonix. Furthermore, SALAMI contains various tracks with limited inter-annotator agreement [6], making it harder for an algorithm to match the reference annotations.

Finally, we report the flat results in Table 3 (SALAMI is evaluated against the “uppercase” level). Note that these results mimic the behavior of a user choosing their desired segmentation level, as described in Section 5.2. Similar to the multi-label results, vanilla DEF also outperforms the baselines on all flat metrics in both datasets, including HR_{0.5} which is the strictest metric for boundary retrieval.

6.4 Qualitative Analysis

To gain further insight into how our approach compares to the LSD baseline, we examine the multi-level segmentations they produce for a particular track in the Harmonix dataset: track 199, “The Number of the Beast” by Iron Maiden (we encourage the reader to listen to this track to better follow this section). The multi-level segmentations are shown in Figure 3 (we use vanilla DEF_{0.5, 0.5}) with the reference boundaries overlaid as vertical magenta lines and the reference section labels printed at the top of each plot.

It is apparent that the baseline method produces many more noisy segments (i.e., too short, not pertinent) compared to our approach. Particularly relevant is the “breakdown” segment, where there are multiple changes in terms of instrumentation: the drums stop suddenly, the rhythm and bass guitars change riffs drastically, and all this occurs between two different guitar solos (“solo” and “solo2”). The baseline method detects these short changes starting at level 3, without being able to detect the whole “breakdown” as a single whole section. This also prevents it from recognizing other important sections at levels 3 and 4 such as “inst,” which is where the drums kick in along with a loud and long scream, or the “verse” and “chorus” sections, since the new unique sections introduced at these levels are “cannibalized” by the changes in the “breakdown.”

In contrast, our method detects the breakdown as a segment starting from level 6, labeling it similarly to the “bridge,” which makes musical sense given that both parts are instrumental and quite different to all others parts. The

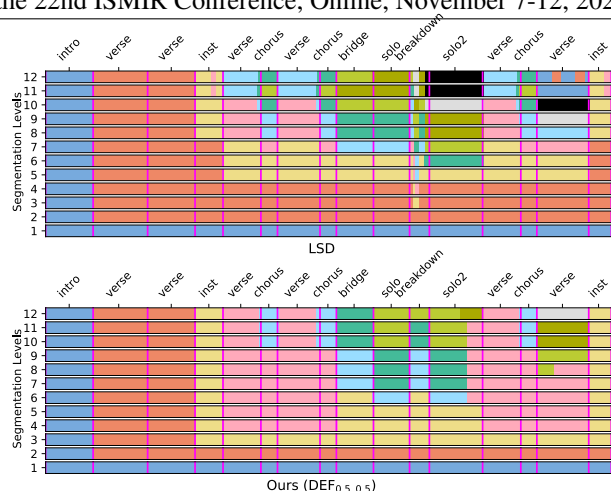


Figure 3: Segmentation of Harmonix track 199: LSD (top) and DEF_{0.5, 0.5} (bottom). Ground truth in vertical lines.

absence of noisy short segments in our approach can be attributed, in all likelihood, to our proposed section fusion algorithm. Our method successfully captures the drum entrance in level 3, identifying three highly differentiated long segments: spoken word intro (blue), music with minimal drums (orange), and music with full drums (yellow). Successfully capturing these key changes in timbre can be attributed to our introduction of the proposed deep embeddings. Overall, it is apparent in this example that our method obtains notably cleaner sections that better align to the reference annotations thanks to both the deep embeddings and the multi-level section fusion algorithm.

7. CONCLUSION

In this work we introduced a multi-level segmentation method that leverages deep audio embeddings learned via other tasks. Building on an existing multi-level segmentation algorithm based on spectral clustering, we replaced MFCC features with deep embeddings trained via Few-Shot Learning for computing local timbre similarity. We also augmented the CQT features used to identify section repetition with deep embeddings from a state-of-the-art music auto-tagging model that captures similarity along different music dimensions. Next, we introduced a novel section fusion algorithm that leverages the multi-level segmentation to consolidate short segments. Through a series of experiments we showed that our two key contributions—replacing the handcrafted features with our proposed deep embeddings and applying multi-level section fusion—lead to significant improvements in multi-level music segmentation, outperforming two strong baselines and yielding state-of-the-art results. Finally, we complemented our quantitative results with a qualitative analysis to gain further insight into how our proposed enhancements improve segmentation performance. Future work includes evaluating a broader range of deep embeddings with our segmentation approach such as OpenL3 [35] and VGGish [36], exploring advanced feature fusion approaches such as SNF [9], and investigating automated strategies for determining the optimal minimum section duration for section fusion.

8. ACKNOWLEDGMENTS

We would like to thank the authors of our baselines, Brian McFee, Daniel P.W. Ellis, and Christopher Tralie for making their code publicly available and reproducible.

9. REFERENCES

- [1] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, “Audio-Based Music Structure Analysis: Current Trends, Open Challenges, and Applications,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 246–263, 2020.
- [2] M. Schedl, “Intelligent user interfaces for social music discovery and exploration of large-scale music repositories,” in *Proceedings of the 2017 ACM Workshop on Theory-Informed User Modeling for Tailoring and Personalizing Interfaces*, ser. HUMANIZE ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 7–11. [Online]. Available: <https://doi.org/10.1145/3039677.3039678>
- [3] H. Jhamtani and T. Berg-Kirkpatrick, “Modeling self-repetition in music generation using generative adversarial networks,” 2019.
- [4] A. Bozzon, G. Prandi, G. Valenzise, and M. Tagliasacchi, “A music recommendation system based on semantic audio segments similarity,” in *Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications*, ser. EuroIMSA ’08. USA: ACTA Press, 2008, p. 182–187.
- [5] J. B. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, “Design and Creation of a Large-Scale Database of Structural Annotations,” in *Proc. of the 12th International Society of Music Information Retrieval*, Miami, FL, USA, 2011, pp. 555–560.
- [6] O. Nieto and J. P. Bello, “Systematic Exploration of Computational Music Structure Research,” in *Proc. of the 17th International Society for Music Information Retrieval Conference*, New York City, NY, USA, 2016, pp. 547–553.
- [7] B. McFee, O. Nieto, M. M. Farbood, and J. P. Bello, “Evaluating hierarchical structure in music annotations,” *Frontiers in Psychology*, vol. 8, no. 1337, 2017.
- [8] B. McFee and K. M. Kinnaird, “Improving Structure Evaluation Through Automatic Hierarchy Expansion,” in *Proc. of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 152–158.
- [9] C. J. Tralie and B. McFee, “Enhanced Hierarchical Music Structure Annotations via Feature Level Similarity Fusion,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May, pp. 201–205, 2019.
- [10] J. Foote, “Automatic Audio Segmentation Using a Measure Of Audio Novelty,” in *Proc. of the IEEE International Conference of Multimedia and Expo*, New York City, NY, USA, 2000, pp. 452–455.
- [11] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos, “Unsupervised Music Structure Annotation by Time Series Structure Features and Segment Similarity,” *IEEE Transactions on Multimedia, Special Issue on Music Data Mining*, vol. 16, no. 5, pp. 1229 – 1240, 2014.
- [12] K. Ullrich, J. Schlüter, and T. Grill, “Boundary Detection in Music Structure Analysis Using Convolutional Neural Networks,” in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 417–422.
- [13] M. C. McCallum, “Unsupervised Learning of Deep Features for Music Segmentation,” in *Proc. of the 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, 2019.
- [14] B. McFee and D. P. W. Ellis, “Analyzing Song Structure with Spectral Clustering,” in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 405–410.
- [15] ———, “Learning to Segment Songs With Ordinal Linear Discriminant Analysis,” in *Proc. of the 39th IEEE International Conference on Acoustics Speech and Signal Processing*, Florence, Italy, 2014, pp. 5197–5201.
- [16] T. Grill and J. Schlüter, “Music Boundary Detection Using Neural Networks on Combined Features and Two-level Annotations,” in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Málaga, Spain, 2015.
- [17] U. von Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, aug 2007.
- [18] H. Grohganz, M. Clausen, N. Jiang, and M. Müller, “Converting Path Structures into Block Structures using Eigenvalue Decomposition of Self-Similarity Matrices,” in *Proc. of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.
- [19] J. Snell, K. Swersky, and R. Zemel, “Prototypical networks for few-shot learning,” in *Advances in Neural Information Processing Systems*, 2017.
- [20] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, “Metric learning vs classification for disentangled music representation learning,” in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [21] C. V. Cotton and D. P. W. Ellis, “Spectral vs. spectrotemporal features for acoustic event detection,” in *IEEE Worksh. on Apps. of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, Oct. 2011, pp. 69–72.

- [22] Y. Wang, J. Salamon, N. J. Bryan, and J. P. Bello, “Few-shot sound event detection,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 81–85.
- [23] Y. Wang, J. Salamon, M. Cartwright, N. J. Bryan, and J. P. Bello, “Few-shot drum transcription in polyphonic music,” in *21st International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2020.
- [24] J. C. Brown, “Calculation of a constant Q spectral transform,” *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.
- [25] D. Fitzgerald, “Harmonic/percussive separation using median filtering,” in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, vol. 13, 2010.
- [26] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, “Disentangled multidimensional metric learning for music similarity,” in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6–10.
- [27] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” in *12th Int. Soc. for Music Info. Retrieval Conf.*, Miami, USA, Oct. 2011, pp. 591–596.
- [28] O. Nieto, M. McCallum, M. E. Davies, A. Robertson, A. Stark, and E. Egozy, “The Harmonix Set: Beats, Downbeats, and Functional Segment Annotations of Western Popular Music,” in *Proc. of the 20th International Society for Music Information Retrieval Conference*, Delft, The Netherlands, 2019, pp. 565–572.
- [29] M. Levy and M. Sandler, “Structural Segmentation of Musical Audio by Constrained Clustering,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 318–326, feb 2008.
- [30] H. Lukashevich, “Towards Quantitative Measures of Evaluating Song Segmentation,” in *Proc. of the 10th International Society of Music Information Retrieval*, Philadelphia, PA, USA, 2008, pp. 375–380.
- [31] F. Korzeniowski, B. Sebastian, and G. Widmer, “Probabilistic Extraction of Beat positions from a Beat Activation Function,” in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 513–518.
- [32] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: a new Python Audio and Music Signal Processing Library,” in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.
- [33] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and Music Signal Analysis in Python,” in *Proc. of the 14th Python in Science Conference*, Austin, TX, USA, 2015, pp. 18–25.
- [34] K. Choi, G. Fazekas, M. Sandler, and K. Cho, “Convolutional recurrent neural networks for music classification,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2392–2396.
- [35] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, listen and learn more: Design choices for deep audio embeddings,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, May 2019, pp. 3852–3856.
- [36] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, New Orleans, LA, USA, Mar. 2017, pp. 131–135.

MULTI-TASK LEARNING OF GRAPH-BASED INDUCTIVE REPRESENTATIONS OF MUSIC CONTENT

Antonia Saravanou^{1†} Federico Tomasi² Rishabh Mehrotra² Mounia Lalmas²

¹ University of Athens, ² Spotify

antoniasar@di.uoa.gr, {federicot,rishabhm,mounial}@spotify.com

ABSTRACT

Music streaming platforms rely heavily on learning meaningful representations of tracks to surface apt recommendations to users in a number of different use cases. In this work, we consider the task of learning music track representations by leveraging three rich heterogeneous sources of information: (i) organizational information (e.g., playlist co-occurrence), (ii) content information (e.g., audio and acoustics), and (iii) music stylistics (e.g., genre). We advocate for a multi-task formulation of graph representation learning, and propose MUSIG: MUlti-task Sampling and Inductive learning on Graphs. MUSIG allows us to derive generalized track representations that combine the benefits offered by (i) the inductive graph based framework, which generates embeddings by sampling and aggregating features from a node’s local neighborhood, as well as, (ii) multi-task training of aggregation functions, which ensures the learnt functions perform well on a number of important tasks. We present large scale empirical results for track recommendation for the playlist completion task, and compare different classes of representation learning approaches, including collaborative filtering, word2vec and node embeddings, as well as graph embedding approaches. Our results demonstrate that considering content information (i.e., audio and acoustic features) is useful and that multi-task supervision helps learn better representations.

1. INTRODUCTION

Recent advancements in recommendation technology [1–7] have fueled music listening on on-demand music streaming apps (e.g., Spotify, Pandora, Apple Music). Playlists form the backbone of how music is consumed, with users relying on curated or user generated playlists to discover and consume music from a massive pool of millions of songs. Personalization models built for selection of tracks, generation of playlists and subsequent recom-

mendation of playlists to users, rely heavily on representing tracks in a meaningful way, to best capture the various intricacies and differences across musical tracks.

When learning track representations, one can leverage various types of heterogeneous information encoded in music data to benefit downstream tasks of music recommendation: (i) *organizational information*: tracks organized into playlists; (ii) *content information*: audio and acoustic features extracted from tracks; and (iii) *musical stylistics*: musical domain characteristics like music genres. Further, such representations are used by system designers for many different downstream tasks, e.g., track recommendation for playlist completion, ranking tracks within a playlist and suggesting tracks in sequential sessions (i.e., track radios). Unfortunately, the learnt representations are often ill-suited for such tasks, because of mismatch between the original learning and downstream task. Instead, training the representation learning system on multiple, complementary tasks would enable learning richer representations, allowing for an increased adoption of the representations for a variety of newer downstream tasks, which is important in an industrial setting.

Motivated by the above aspects, we propose a MUlti-task based Sampling and Inductive Graph learning approach (MUSIG) for learning track representations, that combines information from heterogeneous sources and benefits from supervision signals from a number of tasks. Instead of training a distinct embedding vector for each node, following recent advancements in graph based learning [8], we train a set of aggregator functions. These functions aggregate information from different nodes in the local neighborhood, and are trained via pairwise multi-task supervision. For each pair of nodes, we consider three tasks: (i) playlist co-occurrence, (ii) genre prediction, and (iii) regression of tracks’ audio and acoustic properties.

Furthermore, the trained aggregator functions afford the inductive ability to the model. Indeed, we can generate embeddings for unseen nodes by applying the learned aggregation functions. Finally, jointly leveraging organizational, content and stylistics information helps us cover individual track level information (e.g., audio/acoustic features) as well as information from across various groupings of track such as music stylistics based grouping (e.g., genres) and user consumption based grouping (e.g., playlists).

We present a case-study on music recommendations and conduct large scale analysis to compare different techniques across several qualitative and quantitative measures.

[†]This work was done while the first author was an intern at Spotify.



We make a number of contributions, including algorithmic and qualitative insights of music data at scale. Our findings suggest that extracting audio and acoustic features from music content is useful, and the addition of such content attributes better drives the representation of the tracks, especially when large amount of consumption data is not available, e.g., when launching in new markets. Furthermore, we show that training the model on multiple tasks results in performance improvements and enables learning more generalizable representations. We contend that our findings have implications on the design and development of representation learning approaches not only for music, but also for other types of data.

2. RELATED WORK

Music representation learning. Recent works on music representation learning rely on deep neural networks to generate music embeddings, using various groups of features: sequence of notes [9], music signals [10], pitch sequences, temporal dependencies [11, 12], and artist features [13]. Using music signals/notes from a track to generate track embeddings has shown various degrees of success, and research in this area is still ongoing. In this work, we use similar features and compare how graph representation learning models can incorporate them.

Word2Vec style embeddings. Text representations have been extensively studied in the last years. Word embeddings refer to low-dimensional real-valued vector representations for each term in the input vocabulary. Word2Vec [14] and GloVe [15] are two well-known word embedding algorithms that learn embedding vectors based on the idea that similar words appear in similar contexts. Word embeddings have been applied in various contexts, such as item recommendations [16–18], music recommendations [19] and query modeling and expansion [20, 21]. We also use Word2Vec to generate track embeddings based on the idea that tracks appearing together in a playlist should be closer in the embedding space. However, these approaches are limited as they only consider the sequence in which the items appear, and could not include additional information on the actual content.

Graph based embeddings. In recent years, variations of Word2Vec working on graph structured data were developed. Examples include Deepwalk [22] and Node2Vec [23], which generate random walks in a specified neighborhood of the target node, to compute the node embedding. Significant advancements of learning on graph structures for recommendation applications include GraphSAGE [8], PinSAGE [24], PinnerSAGE [25], IntentGC [26], MEIRec [27]. Most of these methods are based on Graph Convolutional Networks (GCNs) [28], which combine the graph information from the neighborhood of a node (graph structure) and node features (content information) in the creation of the embeddings. Graph-based embeddings are especially useful when nodes and edges have different types [29–32]. Graph representation learning exploit the structure and the features of the

Feature	Description
Genre	One-hot encoded vector of the top-50 popular genres
Popularity	2-dimensional vector with the global and the region popularity
Audio	42-dimensional vector that includes: <i>danceability</i> , <i>energy</i> , <i>liveness</i> , <i>acousticness</i> , <i>loudness</i> , <i>tempo</i> , <i>instrumentalness</i> , <i>valence</i> , etc
Acoustic	8-dimensional vector, corresponding to audio characteristics

Table 1. Track features (values are normalized in 0–1).

data. However, the embeddings are learnt by optimizing the model on a single task, which makes the embeddings not easily generalizable to additional downstream tasks.

3. MULTI-TASK GRAPH EMBEDDINGS

Recommender systems rely heavily on learning meaningful representations of users and content, to offer personalized recommendations piquing users’ interest. With an explicit focus on streaming music platforms, we briefly discuss few important characteristics of representation learning and describe our proposed method, MUSIG, for node representation learning with multi-task supervision.

3.1 Music Graph Data

We work with data consisting of track and playlist information from Spotify, a popular music streaming platform¹. Tracks are organized into *playlists*. Playlists provide information on how users *organize* their music. We represent playlist-track information as a graph and create a (weighted) homogeneous graph containing all tracks in our dataset. Let the graph be $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with nodes \mathcal{V} be the tracks and edges \mathcal{E} be the connections between tracks co-appearing in the same playlist. We set the weight of an edge to be the number of distinct playlists in which the connected tracks co-appear. We keep edges with weight ≥ 10 . Our graph contains 5.2M edges and 15.9K nodes, from a collection of 95K playlists (albums and movies).

Following the approach outlined in [33], we extract various content features from the music recording of the track, including acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, valence and tempo, etc., which we refer to as *audio features* (Table 1). Furthermore, we train a deep neural model on the music recording of each track (via 30-second windows) for a binary classification task of playlist co-occurrence and we use the last layer projected to 8 dimensions as the *acoustic features* of the track.

3.2 Desired Characteristics

We identify few desirable characteristics for representation learning approaches and motivate their need for inclusion with brief supporting analysis. First, many industrial applications, especially in the music domain, require representations to be promptly available for new tracks. Figure 1 (left) plots the increase in new content on a daily

¹ <https://www.spotify.com>

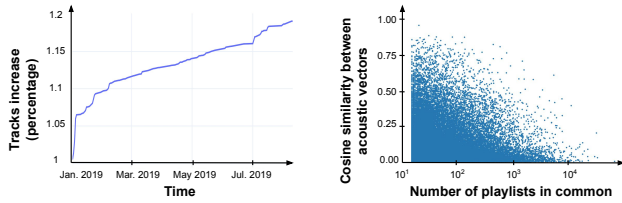


Figure 1. (Left) Percentage of tracks per day in our sample. (Right) Correlation between number of common playlists and acoustic similarity on 50K pairs of tracks.

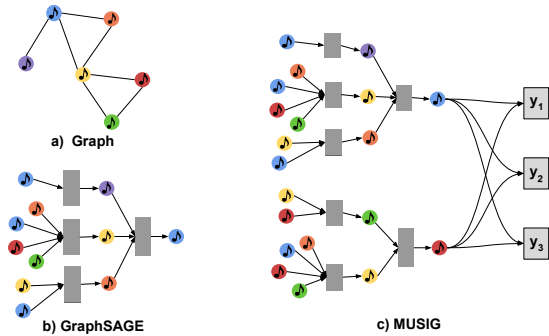


Figure 2. (a) Input graph (each color represents a different track). (b) GraphSAGE model, aggregating the *blue* node’s neighbors to generate the node embedding. (c) Our multi-task model (MUSIG) computing *blue* and *red* nodes’ embeddings while optimizing for three different tasks.

basis; hundreds of tracks are added every day² and learning their representations as early as possible is crucial for production machine learning systems.

Second, a good representation learning approach should leverage all available information, including both playlist co-occurrence and content features. Figure 1 (right) presents the relationship between playlist co-occurrence and acoustic features between randomly selected track pairs. For each pair, we compute the similarity between tracks using their acoustic features, and plot those against the percentage of playlists in which the two tracks co-occur. We observe very low correlation between the two modalities (-0.029), and low density of the scatter-plot in the high similarity co-occurrence region, which highlights that these modalities (graph structure and additional features) capture different information.

Finally, the learnt track representations are employed in a number of use cases across multiple product features. Usually, methods that compute representations are optimized for a specific task. We hypothesize that training the representation learning modules on multiple tasks would enable learning generic representations which would help in a wide variety of downstream recommendation tasks.

3.3 MUSIG Overview

The key idea behind MUSIG is multi-task supervision of neighborhood aggregator functions that aggregate infor-

mation from a node’s neighborhood. The idea of exploiting node’s neighborhood has shown to provide state-of-the-art results [8, 34]. MUSIG adopts a multi-task based learning of aggregator functions, which enables it to learn parameters of the functions based on feedback from multiple, complimentary tasks. Specifically, the algorithmic computations performed by MUSIG are divided into two key steps: (i) **Neighborhood Aggregator Step**, which generates embeddings by aggregating information from different nodes in multiple hops away from a given node (based on search depth), and (ii) **Multi-Task Supervision Step**, which trains the parameters of the aggregation functions by jointly predicting multiple tasks, and back-propagates the combined losses to the aggregator function parameters.

3.4 Neighborhood Aggregator Step

Unlike traditional representation learning approaches, which train a specific embedding for each item in an end-to-end neural model, MUSIG relies on local neighbourhood information and learns aggregator functions that can digest local information to obtain a representation of any given node (Figure 2). For any depth d , the aggregator function recursively aggregates information from all nodes in the d -depth neighborhood of a node, and uses a set of weight matrices \mathbf{W}^k , $\forall k \in \{1, \dots, K\}$ to propagate information between layers arising for each depth. At each iteration, or search depth, the nodes aggregate information from their local neighbors, and as this process iterates, the nodes incrementally gain more information from further reaches of the graph. We follow an iterative approach to aggregate information. First, each node $v \in \mathcal{V}$ aggregates the representations of the nodes in its neighborhood $\mathcal{N}(v)$, $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$, into a single vector $\mathbf{h}_{\mathcal{N}(v)}^{k-1}$:

$$\mathbf{h}_{\mathcal{N}(v)}^{k-1} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \quad (1)$$

The representations at step k depend on the representations generated at $k - 1$, with representations at $k = 0$ being encoded by the default node features provided to the graph. By incorporating node features in the learning algorithm, the model simultaneously learns the topological structure of each node neighborhood as well as the distribution of the node features in the neighborhood. To extract all adjacent nodes, we uniformly sample a fixed-size set of neighbors and thereby keep the computational footprint of each batch fixed. Following [34] we use a permutation invariant aggregator function which implements the mean operator, taking the element-wise mean of the vectors in $\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}$:

$$\text{AGGREGATE}_k(\mathbf{h}_v^{k-1}) \leftarrow \text{MEAN}(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}) \quad (2)$$

We then concatenate the node’s current representation, with the aggregated neighborhood vector. This concatenated vector is fed through a fully connected layer with nonlinear activation function, which transforms the representations to be used at the next step of the algorithm (i.e., \mathbf{h}_v^k , $\forall v \in \mathcal{V}$):

$$\mathbf{h}_v^k \leftarrow \sigma(\text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^{k-1})) \quad (3)$$

² In 2019, there is an average 0.09% daily increase of the catalog, resulting in a 19% increase from the start of the year.

The final representations output at depth K is denoted as $\mathbf{s}_v \equiv \mathbf{h}_v^K$, which is used in the next step during training.

3.5 Multi-Task Supervision

The performance of representations learnt by MUSIG model relies heavily on how well the aggregator functions are trained and what tasks they are trained on. Most graph-based representation learning approaches are trained only on link prediction tasks, and hence learn function parameters only to do well on link prediction task. We hypothesize that training these parameters in a multi-task setting would make the parameters (and in turn, output representations) generalizable across multiple downstream applications. Learning under multi-task supervision offers various benefits, including imparting inductive bias via auxiliary tasks, which cause the model to prefer hypotheses that explain more than one task. This improves generalization by sharing the domain information between complimentary tasks, which is achieved by using a shared representation to learn multiple tasks — what is learned from one task can help learn other tasks.

Considering the output from the previous step, we denote by s_i a track from the input space S and a collection of task spaces $\{Y^t\}_{t \in [T]}$. To train the aggregator functions in a multi-task learning setup, we consider large sample of i.i.d. data points $\{\langle s_i, s_{j \in I(i)} \rangle, y_i^1, y_i^2, \dots, y_i^T\}$, where $\langle s_i, s_{j \in I(i)} \rangle$ represents a pair of nodes (i.e., tracks) with $s_{j \in I(i)}$ being a track derived either from neighborhood of s_i , $\mathcal{N}(s_i)$, or negatively sampled from elsewhere. T is the number of tasks, N is the number of such node pairs sampled, and y_i^t is the label of the t -th task for the i -th track pair. Essentially, for each pair of tracks sampled from the graph, we consider labels obtained via different tasks. We further consider a parametric hypothesis class per task as $f^t(\langle s_i, s_{j \in I(i)} \rangle; \theta^{sh}) : S \rightarrow Y^t$, such that the parameters (θ^{sh}) are shared between tasks. We also consider task-specific loss functions $\mathcal{L}^t(\cdot, \cdot) : S^t \times S^t \rightarrow R^+$.

We employ an empirical risk minimization formulation of multi-task learning, and minimize the loss function:

$$\min_{\theta^{sh}} \sum_{t=1}^T c^t \mathcal{L}^t(\theta^{sh}) \quad (4)$$

for some static or dynamically computed weights c^t per task. In essence, losses from all tasks are combined into a single surrogate task via linear weighted scalarization, with each task having c^t weight. We perform grid search over the space of the parameters to estimate the final set of task weights used to report results. $\mathcal{L}^t(\theta^{sh})$ is the empirical loss of the task t , defined as:

$$\mathcal{L}^t(\theta^{sh}) \triangleq \frac{1}{N} \sum_i \mathcal{L}(f^t(\langle s_i, s_{j \in I(i)} \rangle; \theta^{sh}), y_i^t) \quad (5)$$

We apply a multi-task supervision based loss function to the output representations, $\mathbf{z}_u, \forall u \in \mathcal{V}$, and tune the weight matrices $\mathbf{W}^k, \forall k \in \{1, \dots, K\}$, and parameters of the aggregator functions via stochastic gradient descent.

3.5.1 Identifying Supervision Tasks

The multi-task supervision of the model encourages nodes to have representations that help them solve all tasks for which the model is trained on. Our choice of supervision tasks is guided by our focus on leveraging the heterogeneous information encoded in music data, specifically, around three types of information (i) *organizational information*: tracks organized into playlists, (ii) *content information*: audio and acoustic features extracted from tracks, and, (iii) *musical stylistics*: musical domain characteristics like music genres. To have a representative set of tasks to train the model on, we select one task from each of these three categories of information:

1. **Playlist prediction:** binary classification task, where we predict whether or not the two tracks co-occur in same playlists. This task encapsulates the organization structure embedded in music playlists: two tracks sharing a playlist would make their representations similar to each other.
2. **Genre prediction:** binary classification task, where we predict whether two tracks belong to the same genre. Genres are useful for the categorization, and training on them ensures that tracks with the same genre have similar representations.
3. **Acoustic or audio similarity prediction:** regression task, where we encourage the embeddings to capture similarities in the music content space. We define track similarity as the inner product between acoustic or audio vectors and, thus, force the learnt space to encode music characteristics. This training task enforces representations to rediscover audio and acoustic distances between the tracks.

To better highlight that these tasks contribute heterogeneous information during representation learning, we compute the label correlation across them, and observe that the playlist prediction task has very little correlation with the other two tasks, and that the genre and acoustic similarity prediction tasks do share some commonality, but still differ enough (Figure 1). We use binary cross-entropy loss for the classification task and RMS loss for the regression task. Overall, the final loss function combines the losses from each of the three tasks as:

$$\mathcal{L}_{\text{Final}} = c_1 \mathcal{L}_{\text{Playlist}} + c_2 \mathcal{L}_{\text{Genre}} + c_3 \mathcal{L}_{\text{AudioDist}}. \quad (6)$$

Importantly, unlike previous approaches, the representations s_i that are fed into this loss function are generated from the features contained within a node’s local neighborhood, rather than training a unique embedding for each node (via an embedding look-up).

4. EXPERIMENTAL SETUP

To study the quality of the track embeddings, we conduct an empirical evaluation using data from Spotify.

Method	Features	HR	MRR	Prec@1	Prec@5	Prec@10	NDCG@1	NDCG@5	NDCG@10
CF	-	0.2715	0.1350	0.0119	0.0136	0.0119	0.0610	0.0890	0.0921
TRACK2VEC	-	0.5616	0.0174	0.0123	0.0218	0.0225	0.0123	0.0520	0.0813
NODE2VEC	-	0.4739	0.0128	0.0001	0.0005	0.0025	0.0001	0.0008	0.0065
TRACK2VEC	GENRE	0.5017	0.0137	0.0126	0.0133	0.0139	0.0126	0.0378	0.0575
TRACK2VEC	POPULARITY	0.5029	0.0138	0.0125	0.0140	0.0137	0.0125	0.0388	0.0564
TRACK2VEC	AUDIO	0.5020	0.0133	0.0156	0.0132	0.0128	0.0156	0.0368	0.0539
TRACK2VEC	ACOUSTIC	0.5014	0.0137	0.0145	0.0145	0.0140	0.0145	0.0425	0.0610
GRAPHSAGE	GENRE	0.4981	0.0132	0.0038	0.0129	0.0135	0.0038	0.0318	0.0499
GRAPHSAGE	POPULARITY	0.5038	0.0149	0.0239	0.0219	0.0180	0.0239	0.0624	0.0808
GRAPHSAGE	AUDIO	0.5914	0.0162	0.0143	0.0155	0.0154	0.0143	0.0420	0.0615
GRAPHSAGE	ACOUSTIC	0.5382	0.0163	0.0164	0.0217	0.0186	0.0164	0.0594	0.0795
MUSIG	GENRE	0.7077	0.0359	0.0278	0.0362	0.0393	0.0278	0.0724	0.1150
MUSIG	POPULARITY	0.7505	0.0358	0.0024	0.0193	0.0327	0.0024	0.0312	0.0776
MUSIG	AUDIO	0.7203	0.0324	0.0661	0.0494	0.0415	0.0661	0.1240	0.1614
MUSIG	ACOUSTIC	0.7305	0.0308	0.0412	0.0429	0.0372	0.0412	0.1047	0.1404

Table 2. Results of the comparison methods using the 50% of playlist’s tracks as seed track list.

4.1 Downstream Task: Playlist Completion

Playlists are the backbone of how music content is consumed, with over one-third consumption resulting from user-generated playlists.³ To assist users in selecting music for their playlists from the massive music catalog of million tracks, platforms rely on track recommendation services for playlist completion. Good track representations are crucial for the playlist completion task to be effective. Given a number of tracks in a playlist, our goal is to recommend related tracks. We construct this experiment in an offline fashion. We randomly select 10K playlists that have at least 40 tracks. We keep the top $x\%$ tracks to be the seedlist and we calculate the average embedding from those. Then, we mix the bottom $(100 - x)\%$ tracks with the same number of tracks from a random pool and we call them candidate tracks C . We calculate the *cosine similarity* of all pairs (z_s, z_c) , where z_s is the average seedlist embedding and z_c is the embedding of candidate $c \in C$, and we rank them in descending order. Finally, we recommend the top $(100 - x)\%$ ranked tracks.

4.2 Baselines

We compare the proposed MUSIG with representative models from the three different classes of representation learning approaches: collaborative filtering, word2vec based models and graph embedding based model.

1. Collaborative Filtering. We compare with a collaborative filtering matrix factorization method trained with WARP loss [35], which aims at maximizing the rank of positive examples by repeatedly sampling negative ones.

2. Track2Vec. We compare a Word2Vec-based model, considering tracks co-occurring in the same playlists. We also concatenate the normalized features of Table 1 to the final embeddings of the model for fair comparison (TRACK2VEC-FEATURE).

3. Node2Vec. This approach takes into account random walks in the neighborhood of the node to create embeddings. Tracks connected by links in the graph are encouraged to be closer in the embedding space.

4. GraphSAGE [8]. This is a node representation model that produces embeddings based on the structure (i.e., node neighborhood) as well as the feature vector.

5. GraphSAGE-Feature. To investigate the performance of the model when adding features, we run the GraphSage model with all four different groups of features. To tune the model, we use the same parameters as before.

Our MUSIG/MUSIG-Feature model. Since the proposed MUSIG model affords multiple supervision, it is trained on genre prediction and audio/acoustic feature similarity tasks in addition to playlist co-occurrence task. We modulated the balance between the three tasks by empirically selecting the best performing triple (c_1, c_2, c_3) of Eq. (6) across the following set: $\{(1, 1, 1), (1, 0, 0), (0.7, 0.1, 0.2), (0.4, 0.2, 0.4)\}$, to evaluate different properties of the single loss functions (i.e., when all losses weight the same; when only the first is non-zero, which is equivalent to a single task GraphSAGE).

4.3 Evaluation Metrics

We use four metrics to compare the approaches on the playlist completion task. Firstly, we include the standard versions for Precision at k ($P@k$) and Normalized Discounted Cumulative Gain at k ($NDCG@k$). We define *hit rate* (HR) as the fraction of tracks that were ranked in the top K candidates for a specific playlist P . This metric directly measures the probability that the track recommendations are the correct ones. In our experiments, K dynamically changes based on the size of the playlist, and it is defined as $K = (100 - x)/|P|$, where x is the size of the seedlist tracks and $|P|$ the size of the playlist. We also use (scaled) mean reciprocal rank (MRR), which takes into account the rank of the track u among recommended tracks for playlist P , defined as [24]:

$$MRR = \frac{1}{n} \sum_{u \in P} \frac{1}{|R_u/10|},$$

where n is the number of all pairs and R_u is the rank of the track u among all recommended tracks for playlist P .

³ <https://www.businessofapps.com/data/spotify-statistics/>

4.4 Comparison across approaches

We compare all representation learning approaches to ours (MUSIG), on the playlist completion task (Table 2). Results are calculated using the 50% of the playlist as seedlist. In Section 4.6, we discuss more on the performance of the models using all features. MUSIG trained on the Music Graph using the multi-task training and including the POPULARITY in the node attributes, outperforms all other models. More specifically, we observe that MUSIG-POPULARITY achieves the best HR=75.05%, while the best performance from any of the comparison models is achieved by GRAPHSAGE-AUDIO with an HR=59.14%. All results were tested for statistical significance and proven significant ($p \ll 0.01$).

4.5 Impact of training on multiple tasks

MUSIG improves the hit-rate score of the best existing model by 27%. This is an important indicator that embeddings generated by optimizing on multiple tasks are able to significantly improve the performance of the downstream task of playlist completion. Furthermore, this indicates that imparting the representations to perform well on genre classification and acoustic/audio distance similarity tasks enriches them further, improving the performance. We leave for future work further validation of other tasks for training, and the impact on other downstream tasks.

4.6 Impact of Features

We extensively investigate the importance of leveraging content features while learning embeddings. We select the best performing track (TRACK2VEC) and node embedding (GRAPHSAGE) models, and we evaluate the performance of these models and MUSIG using different groups of features node attributes (GENRE, POPULARITY, AUDIO, ACOUSTIC). For fair comparison, in TRACK2VEC we use aggregations of features and track embeddings.

In Table 2 we observe that TRACK2VEC achieves best performance when trained *without* the content features, which was expected since the model is designed to leverage only *organization* information. Second, we observe that in MUSIG the GENRE⁴, AUDIO and ACOUSTIC features achieve lower hit rate scores, when compared to the POPULARITY features. An explanation is that all three features are already included as tasks in MUSIG, while popularity enriches further the learning phase. Intuitively, popularity does have a relationship to content, as it is related to more “mainstream” or “alternative” track types. However, in all groups of content features, our model outperforms all other models when trained with the same content features. The improvements of our model for each group of content features are: GENRE: 42%, POPULARITY: 49%, AUDIO: 22%, and, ACOUSTIC: 36% compared to the second best model. This highlights that the information contained by the audio and the acoustic features extracted from mu-

⁴ An explanation for the limited performance offered by genre could be our restriction to the most 50 popular genres (Table 1).

Regr. Task	Features	HR	MRR	Prec@10	NDCG@10
AUDIO	ACOUSTIC	0.7305	0.0308	0.0372	0.1404
AUDIO	AUDIO	0.5518	0.0186	0.0164	0.0581
ACOUSTIC	ACOUSTIC	0.7203	0.0324	0.0415	0.1614
ACOUSTIC	AUDIO	0.4339	0.0134	0.0085	0.0295

Table 3. Regression task and node features interplay.

sic recordings is indeed informative, and leveraging them while learning embeddings is useful.

We also evaluate the models for the playlist completion task when using *all* four feature categories as node attributes. For MUSIG, we only add the POPULARITY feature in the node attributes, as information from the rest of the available features is already used in the tasks during the training. The best performing model is MUSIG, which achieves $HR=0.75$. The second best performance, GRAPHSAGE with all features as node attributes; achieves $HR=0.50$. This highlights that the multi-task learning in our MUSIG model achieves better results in all cases, for each feature separately but also in combination, further motivating multi-task learning methods for node classification tasks.

4.7 Variations in Multi-task Learning

To leverage music audio and acoustic properties, the proposed model not only considers them as node features, but also as regression tasks in the multi-task setting. We investigate the interplay between using such content information, and we compare multi-task models trained on both audio and acoustic features and tasks (Table 3). Among all combinations of tasks and features, using AUDIO similarity as the regression task with ACOUSTIC features as node features gives the best hit rate, while the reverse model, i.e., ACOUSTIC similarity task with AUDIO features, outperforms other combinations for all other metrics. AUDIO features capture rich information about music content. This further motivates the usefulness of hybrid representation learning approaches, that combines playlist organization information with content information.

5. CONCLUSION

We propose a multi-task graph-based learning model for music recommendation. Our method learns the track representations based on *content* features and *structural* graph neighborhoods, while the multi-task training is aggregating multiple functions and learning representations based on supervision from multiple training tasks. The inductive aspect of MUSIG helps in reducing the wait time to surface new, fresh content in front of users from days to few hours, while the multi-task supervision enables the use of these representations for tasks that could not directly benefit from playlist co-occurrence only. Empirical results demonstrate the benefits of our method, wherein we show the value of the multi-task over the single-task learning. Furthermore, we show that extracting content features (such as audio or acoustic) improves the performance in existing methods, achieving the best improvements when those features are used in the multi-task setting.

6. REFERENCES

- [1] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li, "Personalized recommendation via parameter-free contextual bandits," in *SIGIR '15*. ACM, 2015.
- [2] H. P. Vanchinathan, I. Nikolic, F. De Bona, and A. Krause, "Explore-exploit in top-n recommender systems via gaussian processes," in *RecSys '14*. ACM, 2014.
- [3] E. Christakopoulou and G. Karypis, "Local item-item models for top-n recommendation," in *RecSys '16*. ACM, 2016.
- [4] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," in *SIGKDD*. ACM, 2016.
- [5] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, 2019.
- [6] R. Katarya and O. P. Verma, "Efficient music recommender system using context graph and particle swarm," *Multimedia Tools and Applications*, 2018.
- [7] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based recommendation system for cold start items," *Expert Systems with Applications*, 2017.
- [8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS '17*, 2017.
- [9] R. de Volk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *ISMIR '18*, 2018.
- [10] Y.-J. Luo and L. Su, "Learning domain-adaptive latent representations of music signals using variational autoencoders," in *ISMIR '18*, 2018.
- [11] S. Lattner, M. Grachten, and G. Widmer, "Learning transposition-invariant interval features from symbolic music and audio," 2018.
- [12] —, "A predictive model for music based on learned interval representations," 2018.
- [13] J. Park, J. Lee, J. Park, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," *arXiv:1710.06648*, 2017.
- [14] T. Mikolov, I. Sutskever, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.
- [15] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation. 2014," 2018.
- [16] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2vec applied to recommendation: Hyperparameters matter," in *RecSys '18*. ACM, 2018.
- [17] N. Ben-Lhachemi and E. H. Nfaoui, "Using tweets embeddings for hashtag recommendation in twitter," *Procedia Comput. Sci.*, 2018.
- [18] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *ACM SIGKDD*. ACM, 2015.
- [19] Z. Cheng, J. Shen, L. Zhu, M. S. Kankanhalli, and L. Nie, "Exploiting music play sequence for music recommendation," in *IJCAI*, 2017.
- [20] P. D. Bruza and D. Song, "Inferring query models by computing information flow," in *CIKM '02*. ACM, 2002.
- [21] F. Diaz, B. Mitra, and N. Craswell, "Query expansion with locally-trained word embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.
- [22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*. ACM, 2014.
- [23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD '19*. ACM, 2016.
- [24] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*. ACM, 2018.
- [25] A. Pal, C. Eksombatchai, Y. Zhou, B. Zhao, C. Rosenberg, and J. Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.
- [26] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, and X. He, "Intentgc: A scalable graph convolution framework fusing heterogeneous information for recommendation," in *KDD '19*. ACM, 2019.
- [27] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li, "Metapath-guided heterogeneous graph neural network for intent recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. ACM, 2019.
- [28] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.
- [29] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

- [30] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, “Heterogeneous graph attention network,” in *The World Wide Web Conference*, 2019, pp. 2022–2032.
- [31] L. Yang, Z. Xiao, W. Jiang, Y. Wei, Y. Hu, and H. Wang, “Dynamic heterogeneous graph embedding using hierarchical attentions,” *Advances in Information Retrieval*, vol. 12036, p. 425, 2020.
- [32] H. Xue, L. Yang, V. Rajan, W. Jiang, Y. Wei, and Y. Lin, “Multiplex bipartite network embedding using dual hypergraph convolutional networks,” in *Proceedings of the Web Conference 2021*, 2021, pp. 1649–1660.
- [33] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” 2011.
- [34] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation learning on graphs: Methods and applications,” *arXiv:1709.05584*, 2017.
- [35] J. Weston, S. Bengio, and N. Usunier, “Wsabie: Scaling up to large vocabulary image annotation,” in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

DADAGP: A DATASET OF TOKENIZED GUITARPRO SONGS FOR SEQUENCE MODELS

Pedro Sarmento¹

Adarsh Kumar^{2,3}

CJ Carr⁴

Zack Zukowski⁴

Mathieu Barthet¹

Yi-Hsuan Yang^{2,5}

¹ Queen Mary University of London, UK

² Academia Sinica, Taiwan

³ Indian Institute of Technology Kharagpur, India

⁴ Dadabots

⁵ Taiwan AI Labs

{p.p.sarmiento, m.barthet}@qmul.ac.uk, emperorcj@gmail.com, yhyang@ailabs.tw

ABSTRACT

Originating in the Renaissance and burgeoning in the digital era, tablatures are a commonly used music notation system which provides explicit representations of instrument fingerings rather than pitches. GuitarPro has established itself as a widely used tablature format and software enabling musicians to edit and share songs for musical practice, learning, and composition. In this work, we present DadaGP, a new symbolic music dataset comprising 26,181 song scores in the GuitarPro format covering 739 musical genres, along with an accompanying tokenized format well-suited for generative sequence models such as the Transformer. The tokenized format is inspired by event-based MIDI encodings, often used in symbolic music generation models. The dataset is released with an encoder/decoder which converts GuitarPro files to tokens and back. We present results of a use case in which DadaGP is used to train a Transformer-based model to generate new songs in GuitarPro format. We discuss other relevant use cases for the dataset (guitar-bass transcription, music style transfer and artist/genre classification) as well as ethical implications. DadaGP opens up the possibility to train GuitarPro score generators, fine-tune models on custom data, create new styles of music, AI-powered songwriting apps, and human-AI improvisation.

1. INTRODUCTION

Historically, tablatures' proliferation is closely linked to the lute repertoire, compositions that roughly span from the 16th century onwards, and are still available today [1]. In opposition to standard notational practices (usually referred to as staff notation), in a tablature system for string instruments each staff line on the score represents a string of the instrument, substituting a representation of pitch by a given location on said instrument (i.e. a fingering) [2]. Tablatures are a *prescriptive* type of notation, where the

Figure 1. An excerpt from a GuitarPro song notation using tablatures and score for two guitars, bass and drums.

emphasis is on the action (symbol-to-action), contrary to *descriptive* forms of notation, which establishes a symbol-to-pitch relationship. This characteristic makes tablatures an intuitive and inclusive device for music reading and learning, which can explain their large prevalence for music score sharing over the Internet [3,4]. Often represented as non-standardised text files that require no specific software to read or write, tablatures' online dissemination has surpassed more sophisticated music notation formats, such as Music XML or MIDI [3]. However, tablature representations that rely solely on text have limitations from a user perspective. For example, it is common that rhythm indications are discarded, preventing a comprehensive transcription of the music and automatic playback. Tablature edition software (e.g. GuitarPro¹, PowerTab², TuxGuitar³) can be regarded as a solution for this problem, keeping the *prescriptive* approach, and supporting rhythm notations and playback. By supporting the annotation of multiple instruments, as observable in Figure 1, these tools account for an interactive music experience, either for songwriting or music learning purposes.

The release of this dataset intends to leverage the GuitarPro format used by the before-mentioned software to support guitar and bands/ensembles' related research within the MIR community, focusing specifically on the



© P. Sarmento, A.Kumar, CJ Carr, Z. Zukowski, M. Barthet and Y. Yang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** P. Sarmento, A.Kumar, CJ Carr, Z. Zukowski, M. Barthet and Y. Yang, "DadaGP: A Dataset of Tokenized GuitarPro Songs for Sequence Models", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ <https://www.guitar-pro.com/>

² <http://www.power-tab.net/guitar.php>

³ <https://sourceforge.net/projects/tuxguitar/>

task of symbolic music generation. The contributions of this paper are: (1) a dataset of over 25,000 songs in GuitarPro and token format, together with statistics on its features and metadata, (2) an algorithm and Python software to convert between any GuitarPro file and a dedicated token format suitable for sequence models⁴, (3) results from its main use case, the task of symbolic music generation, and (4) a discussion about further applications for DadaGP and its ethical implications.

In this paper, we first present some relevant background concerning previously released music datasets in symbolic format. In Section 3, we discuss advantages of tab-based datasets for MIR research. We then describe, in Section 4, the details of the DadaGP dataset, its encoder/decoder support tool, the features it encompasses and the ones it lacks. Within Section 5 we present a use case of symbolic music generation using our proposed dataset, supported by previous approaches concerning databases of symbolic music. Section 6 proposes additional applications for the dataset. Finally, in Section 7 we explain the steps needed in order to acquire the dataset, further pointing out some ethical considerations in Section 8.

2. BACKGROUND

Since its release in 1983, the MIDI (Music Instrument Digital Interfaces) standard has remained highly ubiquitous. Unsurprisingly, MIDI has been the most recurrent option in terms of musical notation formats, concerning datasets released within the MIR community, either targeting music generation purposes, that lately have boomed by leveraging deep learning approaches, or aiming for musical analysis, musicology or purely information retrieval ends. A comprehensive overview of previously released datasets in symbolic format is presented in [5]. The authors present MusPy, a toolkit for symbolic music generation, that natively supports a total of eleven datasets. Considering cumulative song duration, the top five datasets are the Lakh MIDI dataset [6], the MAESTRO dataset [7], the Wikifonia Lead Sheet dataset⁵, the Essen Folk Song database [8], and the NES Music database [9]. With respect to music notation formats, these datasets employ MIDI, MusicXML and ABC. Recently, the GiantMIDI-Piano dataset [10], comprising 10,854 unique piano solo pieces, the POP909 dataset [11] and the Ailabs.tw Pop1K7 dataset [12], containing respectively piano arrangements of 909 and 1,748 popular songs, were also released, all relying on MIDI format. This standardisation around MIDI is useful for there are several Python libraries to work with this format, such as music21 [13], mido [14], pretty_midi [15], and jSymbolic [16].

Regarding guitar-oriented research, previous dataset releases have not particularly targeted automatic music generation goals, instead focusing on guitar transcription or playing technique detection. The GuitarSet consists of 360 excerpts of acoustic guitar along with annotations for

string and fret positions, chords and beats [17]. Furthermore, the Guitar Playing Techniques dataset [18] contains 6,580 clips of notes together with playing technique annotations. Likewise, the IDMT-SMT-Guitar dataset [19] also comprises short excerpts that include annotations of single notes, playing techniques, note clusters, and chords. Lately, Chen et al. compiled a dataset of 333 tablatures of fingerstyle guitar, created specifically for the purpose of music generation [20].

To the authors best knowledge, there exists no multi-instrument dataset that is able to combine the ease of use of symbolic formats whilst providing guitar (and bass) playing technique information. Such expressive information is lacking in other formats, and GuitarPro appears as a viable resource for music analysis and generation.

3. MOTIVATIONS: WHY GUITARPRO?

GuitarPro is both a software and a file format, widely used by guitar and bass players, but also by bands. It is mostly utilized for tasks such as music learning and practicing, where musicians simply read or play along a given song, and for music notation, in which composers/bands use the software to either support the songwriting process, or simply as a means for ease of distribution once compositions are done. As an example of the software's widespread dissemination, the tablature site Ultimate Guitar⁶ hosts a catalogue of over 200,000 user-submitted GuitarPro files, containing notations of commercial music, mostly from the genres of rock and metal. One of the main motivations for the creation of DadaGP is to engage the MIR community into research that leverages the expressive information, instrumental parts and song diversity in formats such as GuitarPro. Although GuitarPro is a paid software, free alternatives such as TuxGuitar are capable of editing/exporting into GuitarPro format. Moreover, GuitarPro files can be easily imported into MuseScore⁷, a free software notoriously known for music notation, which also possesses tablature features. However, using MuseScore might present some occasional incompatibilities, specifically those regarding the selection of instruments (e.g. drums are often imported as piano, and the corresponding MIDI instruments need to be manually switched). Another important motivation for the release of this dataset is that it is possible to make conversions between GuitarPro and MIDI files. This can be done inside any of the aforementioned software, by simply exporting into MIDI, or by scripting. Thus, by converting the dataset's GuitarPro files into MIDI, MIDI-based music feature extraction functions available (e.g. Python libraries referenced in Section 2) can be applied. Finally, we believe that our dataset is able to provide researchers with the information present in standard MIDI datasets, while including at the same time prescriptive information useful for guitar-oriented research.

⁴ Available at: <https://github.com/dada-bots/dadaGP>

⁵ No longer available.

⁶ <https://www.ultimate-guitar.com/>

⁷ <https://musescore.com/>

4. DADAGP DATASET

Leveraging the proliferation of music transcriptions available online as GuitarPro files, we compiled DadaGP, a dataset containing 26,181 songs. We also devised an encoding/decoding tool to convert GuitarPro files into tokens, which is described in Section 4.1. In total, it contains 116M tokens, which is about the size of WikiText-103 [21]. In terms of duration, the dataset amounts to over than 1,200 hours (average song length of 2:45 minutes).

4.1 Encoding/Decoding Tool

4.1.1 Feature Extraction with PyGuitarPro

PyGuitarPro [22] is a Python library which reads, writes and manipulates GuitarPro files⁸. Our encoding/decoding tool explores its feature extraction functions, in order to convert much of the information into a tokenized text format. With PyGuitarPro it is possible to acquire information regarding music-theoretic features (e.g. pitch, rhythm, measure, instrument) and playing technique information.

4.1.2 Tokenization

The token format takes inspiration from event-based MIDI encodings used in previous music generation works, such as MuseNet [23], REMI [24] and CP [12]. The tool consists of a Python script that utilizes PyGuitarPro to process GuitarPro files into/from token format. Syntactically, every song begins with `artist`, `downtune`, `tempo` and `start` tokens. A depiction of the conversion process can be seen in Figure 2. Notes from pitched instruments are represented by a combination of tokens in the format of `instrument:note:string:fret` and rests by `instrument:note:rest`. For the drumset, the representation is done by `drums:note:type`, leveraging GuitarPro 5 percussion MIDI maps (e.g. `drums:note:36` for a kick drum, `drums:note:40` for a snare). Every note or rest is separated in time by `wait` tokens. This is sufficient for the decoder to figure out note durations. There is no need to use note-off tokens, because new notes silence old notes, unless a *ghost note* or *let ring* effect is used. Every new measure, note effect, beat effect, and tempo change is registered as a token. Effect tokens are applied to the preceding note token. A histogram containing the most common tokens in DadaGP is available in Figure 4(g).

Furthermore, the DadaGP token format is resilient to syntax errors, such that random token sequences will still produce decodable music. We believe this is helpful when creatively pushing generators to make out-of-distribution sequences using high temperatures, early epochs, extreme latent dimension values, interpolated style conditioning, and other experimental practices.

4.2 Repertoire

Each song is labelled with artist and genre information, although genre tags are absent within original GuitarPro

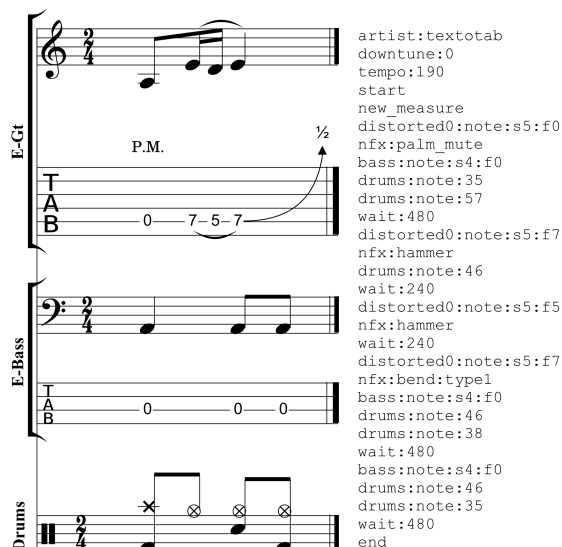


Figure 2. A measure with a distorted guitar, bass and drums in GuitarPro’s graphical user interface (left), and its conversion into token format (right).

files. To this end, we compiled a genre list, with information acquired from the Spotify Web API⁹, querying by artist and song title, resulting in genre metadata for each composition. It is worth mentioning that a given song can have more than one genre attached to it. Information about the most prevalent genres within DadaGP can be seen in Figure 3. While its emphasis is on genres and sub-genres from rock and metal, its corpus is diverse, also including stylistically distinct genres such as jazz, classical, pop and EDM. From Figure 4(a) we observe that most of the songs in DadaGP contain four instrumental parts, usually two guitars, a bass and drums.

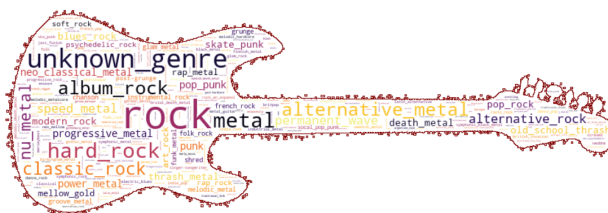


Figure 3. Word cloud representation of the musical genres in DadaGP. Tag size increases with amount of songs.

4.3 Instruments

Regarding instrumentation, for DadaGP a maximum of nine instruments were chosen: three distorted or over-driven guitars, two clean or acoustic guitars, one bass, one drumset, one lead (for instruments with sharp attacks, e.g. piano), and one pad (for instruments used more ambiently, like a choir or a string ensemble). Multiple drum tracks are combined into one. Rare instruments are combined into

⁸ Currently, it supports GP3, GP4 and GP5 files.

⁹ Available at: <https://developer.spotify.com/documentation/web-api/>

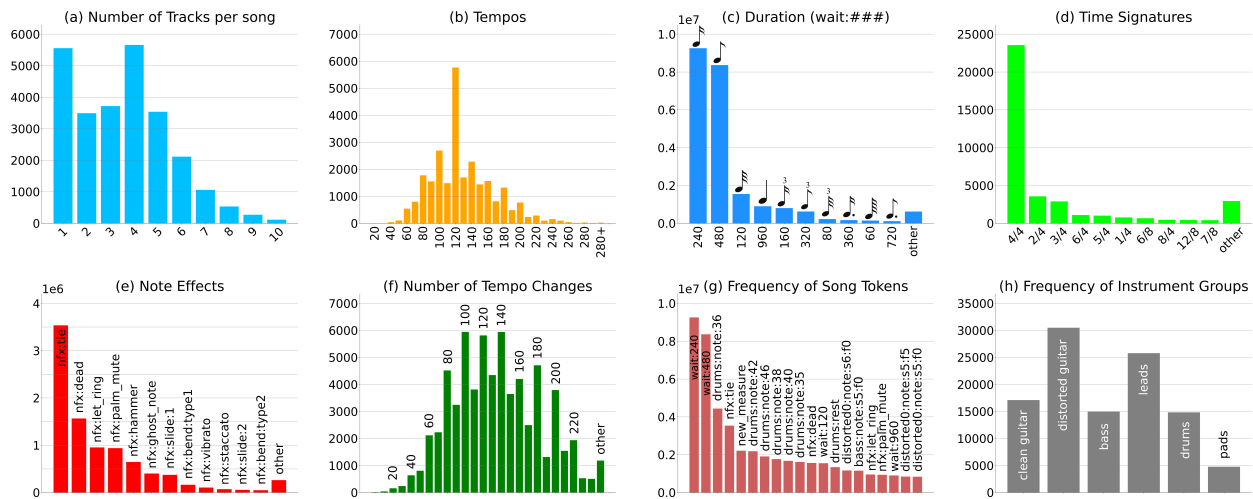


Figure 4. Statistical information about the DadaGP dataset. Histograms of tracks per song (a), initial tempos (b), most common note durations in token and staff notation format (c), time signatures (d), note effects (e), amount of tempo changes (f), most frequent tokens (g) and instruments (h).

the lead and pad tracks. In Figure 4(h) we can notice a predominance of distorted guitars in the dataset. Intuitively this is justified by the presence of two distorted guitars (often one rhythmic and one lead) on most of the songs in DadaGP, due to the predominance of the rock/metal genre. Concerning guitar and bass, 7 string guitars are supported, as are 5 and 6 string basses. Downtuning is supported only if all instruments downtune the same amount, and common tunings such as *Drop D*¹⁰ and *Drop AD* are also included. Rare tunings were dropped from the dataset as the encoder does not support them.

Guitar playing technique notations are represented by note effect tokens (`nfx`), although this family of tokens also holds information about other instruments (e.g. `nfx:tie`, which acts as a link between two adjacent notes). On Figure 4(e) we present a histogram of the most frequent occurrences of these in our dataset, namely *palm mute* (a technique often used with distortion guitars where the guitar player dampens the strings with the right hand palm), bends and vibratos, slides, hammer-ons and pull-offs (both under `nfx:hammer`).

4.4 Meter

As clarified before, each note/rest event is followed by a `wait` token which specifies the number of ticks between it and the succeeding event. In DadaGP, tick resolution uniformly corresponds to 960 ticks per quarter note. For a tempo of 100 bpm, a tick corresponds to $60/(100 * 960) = 0.000625$ seconds. Referring to the excerpt in Figure 2, eighth note events are separated by `wait:480` tokens, and sixteenth note ones by `wait:240`. A histogram with the most common durations in DadaGP is presented in Figure 4(c), in both token and standard staff notation formats, to ease visualization.

¹⁰ A tuning in which only the lowest string is downtuned by one whole step, usually from E to D.

Usually, in a GuitarPro file a default tempo is specified for the entire song, although it supports the inclusion of tempo changes throughout the piece. This is addressed by our encoder/decoder with the tokens `tempo` and `bfx:tempo_change` respectively, which affects note/rest duration. In Figure 4(b) and Figure 4(f) are presented plots corresponding to the most frequent tempos and tempo changes.

The encoder/decoder also supports the representation of measure repetitions with the `measure:repeat` token. Although time signatures are not tokenized, they are inferred by summing the `wait` tokens between the occurrences of `new_measure`. However, this method is insufficient to distinguish between 3/4 and 6/8 measures, for example. To circumvent this, for the plot presented in Figure 4(d) we leveraged PyGuitarPro functions to extract accurate information about the most prevalent time signatures for each measure in our dataset.

4.5 What is Missing?

Information regarding key signature is not provided as part of the dataset. Although key signature can be represented in GuitarPro format, it is rarely present within files. Similarly to the results presented in [6] for the Lakh MIDI dataset, 93.7% of the songs in DadaGP were automatically assigned the key of C Major, rendering these statistics inaccurate.

GuitarPro does not include note velocity information as in MIDI. However, in GuitarPro loudness between notes and musical phrases is notated using traditional dynamic instructions (e.g. *forte*, *pianissimo*, *mezzo-forte*). In its token format, DadaGP does not yet support this, but there is a possibility of accentuating notes at two levels with `nfx:accentuated_note` and `nfx:heavy_accentuated_note`.

Concerning vocals, a common practice with GuitarPro files is to select MIDI wind instruments to notate singing

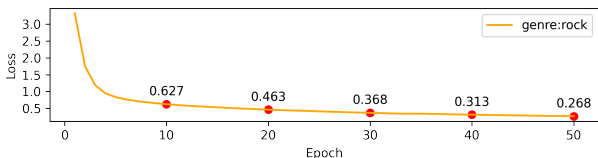


Figure 5. Training loss of the rock subset model, per epoch.

melodies. Currently, our dataset is not well-suited to handle vocals, for these get converted into the `leads` instrument, which may also contain information about other instruments, such as the piano. Lyrics are also possible to include in GuitarPro, but that feature is currently not supported by our encoder/decoder tool.

5. USE CASE: SYMBOLIC MUSIC GENERATION

Recently, the field of symbolic music generation has witnessed consistent progress. Considering works that target symbolic music generation with Transformer-based models, MusicTransformer [25] is a MIDI generator trained on piano performances with improved long-term coherence over vanilla RNNs due to the use of the Transformer [26]. Similarly, MuseNet [23] is a generative Sparse Transformer [27] trained on a larger dataset of MIDI including over 600 styles. An API for the model was launched by OpenAI, which powers the songwriting app MuseTree [28]. However, the model was not released, so it cannot be fine-tuned on custom data. In [29] the author trained a charRNN generator on dozens of GuitarPro songs encoded as a sequence of ASCII characters. It only supported one instrument, and its verbose character-sequence format opened up the possibility for syntax errors.

We tested the DadaGP dataset for a symbolic music generation use case by using the Pop Music Transformer model [24], in which the authors devised a Transformer-XL [30] architecture to generate pop piano compositions in MIDI format. The reason for the choice of this architecture is because this work considers metrical structure in the input data, allowing for an increased awareness in terms of beat-measure structure. We chose the Transformer-XL model as it is able to learn dependencies that are 450% longer than vanilla Transformers, thus well-suited for our task. As per the settings, similarly to the original paper, we used $M = 8$ attention heads and $N = 12$ self-attention layers.

As a proof-of-concept, we collected a subset from our dataset, retrieving 6,910 songs labelled as `genre:rock`. We generated a list of all the unique tokens in this subset, creating a vocabulary with 2,104 entries.

Training was set to run for 50 epochs. With around 43M parameters, this model took around 10 days to perform this task on a Tesla K80 GPU. We consider this to be impractical in terms of reproducibility, so we intend to release pre-trained models from epochs 40 and 50, for which losses can be seen in Figure 5.

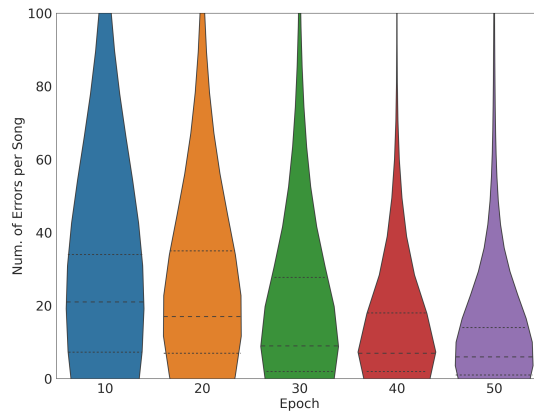


Figure 6. Violin plot of number of errors per song at different epochs.

Regarding inference, we conditioned the model by prompting it with an initial list of tokens comprising `artist`, `downtune`, `tempo` and `start`, necessary for the DadaGP decoder. Furthermore, in an attempt to guide the model towards the generation of music comprising specific instruments, we included tokens for a single note of a distorted guitar, bass guitar and drums. Through experimentation, we set on a limit of 1,024 tokens for each generated song, using 1.2 as temperature parameter. Finally, we manually appended an `end` token in order for the decoder to be able to convert it to GuitarPro format, as this is the instruction which tells the decoder when the song finishes.

As a simple evaluation metric, we focused on the notion of *grammar errors*, namely repetitions of the tokens that should only occur once (`artist:`, `downtune:`, `tempo:`, `start` and `end`), or adjacent repetitions of the same token. Using this, we estimated the number of errors per song, for a corpus of 1,000 generated songs from the model at epochs 10, 20, 30, 40 and 50. As observable in Figure 6, not only the median of the number of errors per song is smaller in later epochs, but also the occurrence of outliers is diminished, as expected.

Despite the limitations of the current evaluation, it allowed us to notice a predominance of a specific error, namely the repetition of the token `end`. This is problematic, because the decoder immediately stops the conversion when an `end` token appears, ultimately shortening songs when in GuitarPro format. To counter this effect, we devised a condition that, during inference, would force the model to sample a different token in the event that an `end` token is selected. Results of generated songs without any curation or post-processing have been made available¹¹.

6. PROSPECTIVE APPLICATIONS

Although primarily tailored for symbolic music generation, below we describe further applications for DadaGP.

¹¹ Available at: <https://drive.google.com/drive/folders/1USNH8olG9uy6vodslM3iXInBT725zult?usp=sharing>

6.1 Guitar-Bass Transcription

The task of guitar-bass transcription from audio recordings is still mostly done manually by musicians, requiring expertise and being both effort and time consuming. In order to automate this, previous research has focused on both solo bass guitar [31, 32] and solo guitar [33–35] transcription. As a contribution to solve this problem, we anticipate that DadaGP can be used to create a synthetic dataset for training guitar-bass transcription models, by rendering its corpus from tablatures into audio, using a DAW and appropriate sound fonts. Such a synthetic dataset can be used to pre-train a model, which can then be fine-tuned afterwards using realistic sounds with aligned scores. This argument is supported by the promising results shown by the Slakh dataset, a synthesized version of the Lakh MIDI dataset, on the task of music source separation [36].

6.2 Music Style Transfer

Recently, the task of style transfer, the process of changing the style of an image, video, audio clip or musical piece so as to match the style of a given example, has been the subject of much attention. First investigated in applications that target computer vision, music style transfer has recently shown promising results in both the audio [37] and symbolic domains [38–40]. As a prospective application of DadaGP, we envisage that genre information can be leveraged in segregating the dataset across different genres, rendering it suitable for the task of musical genre style transfer, as proposed in [41] for the specific morphing between Bach chorales and Western folk tunes. Furthermore, besides musical genre, artistic information can also be used towards the task of composer style transfer, once again by filtering DadaGP across distinct artists.

6.3 Artist/Genre Classification

Another task for which artistic and musical genre information present in DadaGP is useful is artist/genre classification. We hypothesize that these features can be used to train classification models, in order to predict composer style and genre related information from the symbolic representation of the songs itself, similarly to what has been implemented in [42–44]. A thorough survey of the most important approaches regarding music genre classification in the symbolic domain can be consulted in [45]. Furthermore, there is a symbiosis between this task and the one present in the previous subsection, since the models trained for artist/genre classification can be prospectively used in composer style-based feature extractions, which can be further utilized in tasks like composer style conditioned generation and music style transfer.

7. DISTRIBUTION

To ensure reproducibility and facilitate the usage of the dataset, we allow researchers to access DadaGP from a Zenodo repository¹², on application by request. Here

¹²<https://zenodo.org>

we include the token format versions of the songs, the encoder/decoder Python script in order to convert them into/from GuitarPro format, and the statistical data presented on this paper.

8. ETHICAL CONSIDERATIONS

Training large models has a carbon footprint. Some cloud services are carbon neutral, others are not. This should be considered when training large models on this data. Releasing pre-trained models reduces impact, and we intend to do so with the models present in this paper.

Many questions regarding production and consumption of music created with AI are still unanswered. For example: Is it wrong to train machine learning models on copyrighted music? Should this be protected by fair use for artists and scientists? What about commercial use? How to acknowledge, reward and remunerate artists whose music has been used to train models? What if an artist does not want to be part of a dataset? Should creators have a monopoly on their style and exclude others from using their style? Or is style communal? Some of these questions were also raised upon the release of Jukebox [46], an audio model trained on more than 7,000 artists. However, OpenAI made the case that "*Under current law, training AI systems constitutes fair use (...)*" and that "*Legal uncertainty on the copyright implications of training AI systems imposes substantial costs on AI developers and so should be authoritatively resolved*" [47].

9. CONCLUSION AND FUTURE WORK

In this paper we presented DadaGP, a dataset of songs in GuitarPro and token formats, together with its encoding/decoding tool. We discussed the features, strengths and weaknesses of the dataset. Moreover, we presented a symbolic music generation use case entailing a novel approach for multi-instrument music generation in tablature format. Finally, we pointed out additional research applications for DadaGP and discussed some ethical implications. We intend to improve the DadaGP dataset, namely the possibility of removing `measure:repeat` tokens. During generation, we discovered that these tokens were often hard for the model to interpret, sometimes leading to disproportionate measure repetitions. Also, we plan to include note and phrase dynamics information, and the support for vocal instrumental parts. Regarding music generation, we envision to (1) release a pre-trained model which can be fine-tuned on new music, (2) collaborate with artists that use GuitarPro, (3) explore genre/style transfer, (4) and attempt to play the generated songs in social performances.

10. ACKNOWLEDGMENTS

This work is supported by the EPSRC UKRI Centre for Doctoral Training in Artificial Intelligence and Music (Grant no. EP/S022694/1). Thanks to Colin Raffel, Brian McFee, and Sviatoslav Abakumov for discussions and advice.

11. REFERENCES

- [1] R. De Valk, R. Ahmed, and T. Crawford, “JosquIntab: A Dataset for Content-based Computational Analysis of Music in Lute Tablature,” in *Proc. of the 20th International Society for Music Information Retrieval Conference*, 2019.
- [2] T. Magnusson, *Sonic Writing: Technologies of Material, Symbolic & Signal Inscriptions*. Bloomsbury Academic, 2019.
- [3] R. Macrae and S. Dixon, “Guitar Tab Mining, Analysis and Ranking,” in *Proc. of the 12th International Society for Music Information Retrieval Conference*, 2011.
- [4] M. Barthes, A. Anglade, G. Fazekas, S. Kolozali, and R. Macrae, “Music Recommendation for Music Learning: Hotttabs, a Multimedia Guitar Tutor,” in *Workshop on Music Recommendation and Discovery*, 2011, pp. 7–13.
- [5] H. W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, “MusPY: A Toolkit for Symbolic Music Generation,” in *Proc. of the 21st International Society for Music Information Retrieval, ISMIR*, 2020.
- [6] C. Raffel and D. P. W. Ellis, “Extracting Ground Truth Information from MIDI Files: A MIDIfesto,” in *Proc. of the 17th International Society for Music Information Retrieval Conference*, 2016.
- [7] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. Anna Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck Google Brain, “Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset,” 2019.
- [8] “Essen Folk Song Database.” [Online]. Available: <http://www.esac-data.org/>
- [9] C. Donahue, H. H. Mao, and J. Mcauley, “The NES music database: A Multi-Instrumental Dataset with Expressive Performance Attributes,” in *Proc. of the 19th International Society for Music Information Retrieval Conference*, 2018.
- [10] Q. Kong, B. Li, J. Chen, and Y. Wang, “GiantMIDI-Piano: A Large-Scale MIDI Dataset for Classical Piano music,” in *Transactions of the International Society for Music Information Retrieval*, 2020.
- [11] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, “POP909: A Pop-Song Dataset for Music Arrangement Generation,” in *Proc. of 21st International Conference on Music Information Retrieval*, 2020.
- [12] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, “Compound Word Transformer: Learning to Compose Full-Song Music Over Dynamic Directed Hypergraphs,” in *Proc. of the AAAI Conference on Artificial Intelligence*, 2021.
- [13] M. S. Cuthbert and C. Ariza, “music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data,” in *Proc. of the 11th International Society for Music Information Retrieval Conference*, 2010.
- [14] O. M. Bjørndalen, “Mido: Midi objects for python.” [Online]. Available: <https://github.com/mido/mido>
- [15] C. Raffel and D. P. W. Ellis, “Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi,” in *Late-Breaking Demos of the 15th International Society for Music Information Retrieval Conference*, 2014.
- [16] C. McKay and I. Fujinaga, “jSymbolic: A Feature Extractor for MIDI Files,” in *Proc. of the International Computer Music Conference*, 2006.
- [17] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, “GuitarSet: A Dataset for Guitar Transcription,” in *Proc. of the 19th International Society for Music Information Retrieval Conference*, 2018.
- [18] L. Su, L.-F. Yu, and Y.-H. Yang, “Sparse Cepstral, Phase Codes for Guitar Playing Technique Classification.” in *Proc. of the 15th International Society for Music Information Retrieval Conference*, 2014.
- [19] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, “Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score and Instrument-related Parameters,” in *Proc. of the 17th Int. Conference on Digital Audio Effects*, 2014.
- [20] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, “Automatic Composition of Guitar Tabs by Transformers and Groove Modelling,” in *Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020.
- [21] S. Merity, C. Xiong, J. Bradbury, and R. Socher, “Pointer Sentinel Mixture Models,” *Proc. of the 5th International Conference on Learning Representations*, 2016.
- [22] S. Abalumov, “PyGuitarPro.” [Online]. Available: <https://github.com/Perlence/PyGuitarPro>
- [23] C. Payne, “Musenet,” 2019. [Online]. Available: openai.com/blog/musenet
- [24] Y.-S. Huang and Y.-H. Yang, “Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions,” in *Proc. of the 28th ACM International Conference on Multimedia*, 2020.
- [25] C.-Z. Anna Huang and A. M. Vaswani Jakob Uszkoreit Noam Shazeer Ian Simon Curtis Hawthorne Andrew Dai Matthew D Hoffman Monica Dinculescu Douglas Eck Google Brain, “Music Transformer: Generating Music with Long-term Structure,” in *Proc. of the 7th International Conference on Learning Representations*, 2019.

- [26] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *Proc. of the 31st Conference on Neural Information Processing Systems*, 2017.
- [27] R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating Long Sequences with Sparse Transformers,” *arXiv preprint arXiv:1904.10509*, 2019.
- [28] S. Waterman, “Musetree,” 2019. [Online]. Available: <https://stevenwaterman.uk/musetree/>
- [29] M. Moocarme, “Deep learning metallica with recurrent neural networks,” 2016. [Online]. Available: <https://www.mattmoocar.me/blog/tabPlayer/>
- [30] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.
- [31] J. Abeßer and G. Schuller, “Instrument-centered music transcription of solo bass guitar recordings,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 9, 2017, pp. 1741–1750.
- [32] J. Abeßer, S. Balke, K. Frieler, M. Pfeleiderer, and M. Müller, “Deep Learning for Jazz Walking Bass Transcription,” in *Proc. of the AES International Conference on Semantic Audio*, 2017.
- [33] A. Wiggins and Y. E. Kim, “Guitar Tablature Estimation with a Convolutional Neural Network,” in *Proc. International Conference on Music Information Retrieval*, 2019, pp. 284–291.
- [34] S. Rodríguez, E. Gómez, and H. Cuesta, “Automatic transcription of Flamenco guitar falsetas,” in *Proc. International Workshop on Folk Music Analysis*, 2018.
- [35] T.-W. Su, Y.-P. Chen, L. Su, and Y.-H. Yang, “TENT: Technique-embedded Note Tracking for Real-World Guitar Solo Recordings,” in *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, 2019, p. 15–28.
- [36] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, “Cutting Music Source Separation Some Slakh: A Dataset to Study the Impact of Training Data Quality and Quantity,” in *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2019.
- [37] Y.-N. Hung, I.-T. Chiang, Y.-A. Chen, and Y.-H. Yang, “Musical Composition Style Transfer via Disentangled Timbre Representations,” in *Proc. of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 4697–4703.
- [38] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, “Symbolic Music Genre Transfer with CycleGAN,” in *Proc. of the IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 786–793.
- [39] O. Cífka, U. Şimşekli, and G. Richard, “Groove2Groove: One-Shot Music Style Transfer With Supervision From Synthetic Data,” in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, 2020, pp. 2638–2650.
- [40] S.-L. Wu and Y.-H. Yang, “MuseMorphose: Full-song and fine-grained music style transfer with just one Transformer VAE,” *arXiv preprint arXiv:2105.04090*, 2021.
- [41] Y.-Q. Lim, C. S. Chan, and F. Y. Loo, “Style-Conditioned Music Generation,” in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.
- [42] T. J. Tsai and K. Ji, “Composer Style Classification of Piano Sheet Music Images Using Language Model Pre-training,” in *Proc. of the 21st International Society for Music Information Retrieval Conference*, 2020.
- [43] S. Kim, H. Lee, S. Park, J. Lee, and K. Choi, “Deep Composer Classification Using Symbolic Representation,” in *Late-Breaking Demo Session of the 21st International Society for Music Information Retrieval Conference*, 2020.
- [44] A. Kotsifakos, E. E. Kotsifakos, P. Papapetrou, and V. Athitsos, “Genre Classification of Symbolic Music with SMBGT,” in *Proc. of the 6th International Conference on Pervasive Technologies Related to Assistive Environments*. New York, NY, USA: Association for Computing Machinery, 2013.
- [45] D. C. Corrêa and F. A. Rodrigues, “A Survey on Symbolic Data-based Music Genre Classification,” *Expert Systems with Applications*, vol. 60, pp. 190–210, 2016.
- [46] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, “Jukebox: A Generative Model for Music,” 2020. [Online]. Available: <https://github.com/openai/jukebox>.
- [47] OpenAI, “USPTO Comment Regarding Request for Comments on Intellectual Property Protection for Artificial Intelligence Innovation,” 2019. [Online]. Available: https://www.uspto.gov/sites/default/files/documents/OpenAI_RFC-84-FR-58141.pdf

DOES TRACK SEQUENCE IN USER-GENERATED PLAYLISTS MATTER?

Harald Schweiger Emilia Parada-Cabaleiro Markus Schedl

Institute of Computational Perception, Johannes Kepler University Linz (JKU), Austria

Human-centered AI Group, AI Lab, Linz Institute of Technology (LIT), Austria

harald.schweiger@jku.at emilia.parada-cabaleiro@jku.at markus.schedl@jku.at

ABSTRACT

The extent to which the sequence of tracks in music playlists matters to listeners is a disputed question, nevertheless a very important one for tasks such as music recommendation (e. g., automatic playlist generation or continuation). While several user studies already approached this question, results are largely inconsistent. In contrast, in this paper we take a data-driven approach and investigate 704,166 user-generated playlists of a major music streaming provider. In particular, we study the consistency (in terms of variance) of a variety of audio features and metadata between subsequent tracks in playlists, and we relate this variance to the corresponding variance computed on a position-independent set of tracks. Our results show that some features vary on average up to 16% less among subsequent tracks in comparison to position-independent pairs of tracks. Furthermore, we show that even pairs of tracks that lie up to 11 positions apart in the playlist are significantly more consistent in several audio features and genres. Our findings yield a better understanding of how users create playlists and will stimulate further progress in sequential music recommenders.

1. INTRODUCTION

Over the last decade, online streaming services have substantially changed the way people consume music. As a result, research on automatic playlist generation (APG) and automatic playlist continuation (APC) has gained attraction and contributed to improving machine-based creation and extension of item sequences (most commonly, music tracks), respectively. All the more as users nowadays spend over 36 % of their online music listening time on user-generated playlists, 17 % on playlists personalized by recommendation engines, and 15 % on the ones created by professional playlist curators.¹ Together with the fact that users create and share massive amounts of playlists on mu-

¹ <https://www.goodwatercap.com/thesis/understanding-spotify>



© H. Schweiger, E. Parada-Cabaleiro, and M. Schedl. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** H. Schweiger, E. Parada-Cabaleiro, and M. Schedl, “Does Track Sequence in User-generated Playlists Matter?”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online*, 2021.

sic streaming platforms,² this raises the question of how well current research understands the underlying semantics of user-generated playlists.

Most APG and APC approaches include algorithms that are capable of learning sequences [1–5] while other focus on smooth transitions [6]. However, contradictory findings from user-centered studies [7, 8], as well as from offline evaluations of sequence-aware recommenders [9, 10], impair a clear understating of whether tracks’ sequential order has a meaningful role in users’ listening experiences.

To narrow this research gap, the work at hand investigates directly, in a multifaceted manner, various properties shared across subsequent tracks in user-generated playlists. In contrast to other works, we argue that our conducted in-depth statistical analysis of a large set of real user-generated playlists complement findings over conclusions previously drawn from other indirect approaches, such as:

- measuring differences in recommendation accuracy for shuffled playlists [9, 10],
- comparing different machine learning approaches such as sequence aware vs. only context-aware recommenders [3],
- analyzing the effects of adding an additional re-ranking stage to the model [2, 4],
- evaluating feedback from user studies [7, 8, 11].

Against this background, we investigate the following research questions:

RQ1: Does the sequence of tracks matter in user-generated playlists? We approach this question by comparing the variance of subsequent tracks to the overall playlist variance, in terms of a variety of properties, concerning track metadata and audio features.

RQ2: For how long do the properties of one track persist on its successors? We study this question by evaluating the number of tracks that are affected by the previous ones concerning the aforementioned properties.

2. RELATED WORK

Related work can be categorized into (i) user studies investigating the quality criteria of user-generated playlists, (ii) research analyzing the difference between sequential and order-agnostic algorithms for APG or APC, and (iii) works that consider APG and APC as sequential problems, thereby, indirectly assuming the importance of track order.

² For instance, Spotify reports having over 4 billion playlists (<https://newsroom.spotify.com/company-info/>).

Concerning user studies, in the works by Kamehkhosh et al. [8, 11] users were asked to identify quality criteria of playlists. In both works participants ranked (out of 7 options) the track order as the fourth and sixth most important criteria, respectively. Although this might indicate that track order has less relevance to users than other properties, one third of the participants reordered their tracks during one of the experiments by Kamehkhosh et al. [11], which shows that (even unconsciously) track order is, to some extent, relevant to the users. Differently, in the user study conducted by Tintarev et al. [7], participants did not experience their track recommendations to be ordered. Some participants even reported that they generally use randomization for listening to their songs.

Concerning sequence-aware music recommender systems, Bonnin and Jannach [3] showed that algorithms based on sequential patterns outperform association rules. Chen et al. [12] trained a Latent Markov Embedding capable of reproducing coherency of playlists, thereby, outperforming n-gram models. Yang et al. [13] proposed an autoencoder architecture which performed better when track order was not manipulated. In contrast, Vall et al. [9, 10] investigated a recurrent neural network trained once on actual playlists and once on shuffled playlists. They showed that rank-based accuracy did not significantly change between the two settings.

Furthermore, some research acknowledged the importance of track order by directly implementing methods capable of learning track sequences. Bittner et al. [5] identified a vast support for the creation of smooth transitions in commercial DJing software, which led them to implement a system that fosters such transitions. Amongst other works related to the topic [6, 14], Jannach et al. [4] presented a two-stage approach for APC to re-rank candidates coherently with recent tracks. Similarly, Volkovs et al. [2] used a two-stage model including temporal and pairwise interactions which achieved the best score in the 2018 ACM Recommender Systems Challenge.³

Finally, since previous work on APG and APC mostly focus on western music, considering theoretical principles from tonal music is important when investigating tracks' transitions. Yet, in previous works the *mode* is typically considered [15] while the *key* (essential to represent *tonality* besides the *mode*), is often disregarded. Indeed, the role of *tonality*, despite its importance in the hierarchical relationships inherent of Western music,⁴ has been rarely considered in the context of playlist sequentiality [5, 18].

3. DATA AND METHODOLOGY

3.1 Dataset

In order to answer the research questions, we considered the *Million Playlist Dataset* (MPD) provided by Spo-

tify for the ACM Recommender Systems Challenge 2018. It encompasses one million user-generated playlists from US-citizens, with a length between 5 and 250 tracks, and an average length of 66.35 tracks. Overall, the playlists in the dataset contain about 2.3M unique tracks by 296K artists. The dataset includes only publicly shared playlists with at least 5 followers; thus, minimizing the risk of including collections of tracks without any musical theme which are just enjoyed by the creator.

One additional advantage of using this dataset is the coverage of high-level audio features, i. e., descriptors derived from low-level acoustic properties, that can be retrieved by the Spotify API.⁵ These features have been used frequently in the literature [19–22] to analyze or recommend music. In this work, we investigate the following audio features: *acousticness* (confidence that a track contains non electronic instruments); *danceability* (how suitable a track is for dancing); *energy* (measure representing tracks' intensity and activity according to perceptual features such as dynamic range or loudness); *instrumentalness* (probability that a track does not contain vocals); *key* (indicates the tonality of the track without referring to the mode, i. e., the pitch-class); *liveness* (confidence value that indicates whether the track has been performed in presence of an audience); *loudness* (average loudness of the track in decibel); *speechiness* (measures the presence of spoken words); *mode* (indicates the scale of the track, i. e., major or minor, to which the key refers to); *tempo* (pace of the track in beats per minute); *valence* (indicates a track's hedonistic value, i. e., whether it sounds positive or negative).

In addition to the described audio features, we also take into account other three related to metadata, i. e., *artist*, *genre*, and *popularity*. As MPD provides only the main artist per track, we enrich the set of artists by retrieving for each of the 2.3M tracks, also through the Spotify API, all artists which have contributed to a track. This has been done to account for artist collaborations as possible effect of smooth transitions inside playlists. For 136,854 of the 402,867 artists in the enriched artist set, a set of *genres* is available.⁶ We link these genres to the tracks of the playlists in order to analyze whether a shift in genres over time can be observed. Finally, the *popularity* of a track, which describes the recent average number of listening events, is retrieved by the same query as the artists.

All in all, 9 continuous features, i. e., *acousticness*, *danceability*, *energy*, *instrumentalness*, *liveness*, *loudness*, *speechiness*, *tempo*, and *valence*, as well as 5 discrete, i. e., *key*, *mode*, *genre*, *artist*, and *popularity*, are considered.

Note that some features, i. e., *acousticness*, *instrumentalness*, *liveness*, and *speechiness*, describe confidence levels rather than meaningful musical characteristics. Nevertheless, we include these features as they might still be insightful, even with their skewed distribution, towards values of 0 and 1.

From the one million playlists provided by the dataset, we filter out all playlists which have less than 30 tracks:

⁵ <https://developer.spotify.com/documentation/web-api/reference/#category-tracks>

⁶ <https://everynoise.com/>

³ <http://www.recsyschallenge.com/2018>

⁴ From a music theory perspective tonal functionality models listeners' expectations, within and across songs, as shown by the tonal relationship between the different movements of unique compositions, e. g., sonatas (cf. Sonata A in [16]), whose movements' tonalities are typically related in terms of dominant, subdominant, relative, or modal relationships [17].

this yields 704,166 playlists with 2,165,065 unique tracks to be analyzed. The filtering is mainly done for **RQ2**, so that we can analyze tracks dependencies that lie up to 15 tracks apart, which is necessary since our method requires twice of this number of tracks to assure that all tracks are covered in the variance calculation presented in Section 3.2.2. As a side effect of the filtering procedure, we also minimize random noise caused by small playlists.⁷

3.2 Definitions

3.2.1 Playlist Variance

Let T be a list of n tracks $[t_1, \dots, t_n]$ forming an arbitrary playlist of our dataset. Each track t_i is assigned to a set of features where x_i denotes a single feature value, representing any of the considered Spotify features (i. e., the audio features and *popularity*). Besides, the genres and artists of each track are defined as discrete feature sets, G_i and A_i , respectively, through a bag-of-words representation.

The variance according to a feature \mathbf{x} across all tracks, independently of the track order inside a playlist, is calculated as the sum of differences between the average of the feature \bar{x} and all feature values x_i , as given by Equation (1). To avoid ambiguity, from now on we call this the *playlist variance*.

$$\text{pl_var}(T) = \frac{1}{n-1} \sum_i^n (x_i - \bar{x})^2 \quad (1)$$

This formula works in cases for which the mean can be computed. However, calculating the overlap between genres and artists, e. g., with the Jaccard distance, does not provide a mean value. Similarly, the discrete features *key* and *mode* need also a different distance measurement to capture the similarities across tracks' tonalities. In these cases, the *playlist variance* can be calculated by averaging the differences of all pairwise combinations. This has been demonstrated by Zhang and Cheng [23] and it is shown in Equation (2).

$$\text{var}(T) = \frac{1}{n} \sum_i^n (x_i - \bar{x})^2 = \frac{1}{2n^2} \sum_i^n \sum_j^n (x_i - x_j)^2 \quad (2)$$

Equation (2) can now be extended by any arbitrary distance measurement \mathcal{D} and since the distance w. r. t. the same track is always zero, a degree of freedom $n-1$ is considered to compute the variances, as shown in Equation (3).

$$\text{pl_var}(T) = \frac{1}{2n(n-1)} \sum_{i \neq j}^n \mathcal{D}(x_i, x_j)^2 \quad (3)$$

In order to eliminate possible correlations between repeating artists, we prevent some pairwise track combinations to be considered for calculating the playlist variance. The corresponding filter function $\mathcal{F}(A_i, A_j)$ returns 1 if all artists of A_i are different from those of A_j and 0 otherwise.

Adding the filter function $\mathcal{F}(A_i, A_j)$ to the playlist variance results in the *constrained playlist variance*, as defined by Equation (4).

$$\text{cpl_var}(T) = \frac{\sum_{i \neq j} \mathcal{F}(A_i, A_j) \mathcal{D}(x_i, x_j)^2}{2 \sum_{i \neq j} \mathcal{F}(A_i, A_j)} \quad (4)$$

3.2.2 Sequential Variance

To answer the RQs we need to compare the playlist variance with a variance eligible to account for the track order inside playlists. We will refer to this as *sequential variance*, which is the variance of a pair of tracks occurring at a fixed distance (number of tracks) apart in a given playlist. The sequential variance for all track combinations that lie d tracks apart is defined by Equation (5). Note that $d=1$ means that the two tracks are direct neighbors.

$$\text{seq_var}(T) = \frac{1}{2(n-d)} \sum_i^{n-d} \mathcal{D}(x_i, x_{i+d})^2 \quad (5)$$

Similarly as for the constrained playlist variance, to compute the *constrained sequential variance* we apply again the filter function \mathcal{F} on the sequential variance, thus ignoring pairs of tracks by the same artist(s), as defined in Equation (6).

$$\text{cseq_var}(T) = \frac{\sum_i^{n-d} \mathcal{F}(A_i, A_{i+d}) \mathcal{D}(x_i, x_{i+d})^2}{2 \sum_i^{n-d} \mathcal{F}(A_i, A_{i+d})} \quad (6)$$

3.2.3 Proportional Variance

To analyze the aggregated differences between playlist and sequential variance for all considered playlists in the dataset, we calculate for each track list $T \in \mathbf{D}$, where \mathbf{D} denotes to the dataset, the ratio of the playlist variance to the sequential variance. From now on, we refer to it as the *unconstrained proportional variance* (UPV), by this denoting that repeating artists were not excluded. As a minor part of the tracks might present very homogeneous features, sequential variances with values close to zero can occur. Since dividing the playlist variance through these variances may yield proportional variances converging to infinity, we use the median instead of the mean to reduce the UPV values of all playlists to one average value, as shown in Equation (7).

$$\text{prop}(\mathbf{D}) = \text{median}_{T \in \mathbf{D}} \frac{\text{pl_var}(T)}{\text{seq_var}(T)} \quad (7)$$

Note that the *constrained proportional variance* (CPV) is calculated as shown in Equation (7) but considering the constrained versions of the playlist and sequential variance instead.

3.2.4 Feature-specific Distance Measurement

In this section, we summarize the three different distance measurements, previously denoted as \mathcal{D} , to calculate the variances, both the playlist variance and the sequential one:

⁷ It seems that playlists in the dataset are stratified by their track size. The smallest playlists is 5 tracks long and the largest 250.

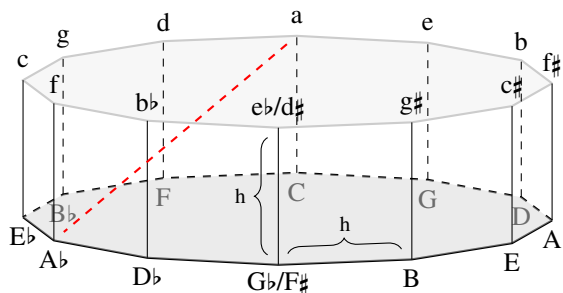


Figure 1. Visualization of the Euclidean distance (marked with the red dashed line) between ‘Ab’ and ‘a’ in \mathbb{R}^3 . Major and minor tonalities are indicated with upper- and lower-case, respectively; # stand for sharps, ♭ for flats.

(i) For all the continuous features and the discrete *popularity*, the variance is calculated by Euclidean distance.

(ii) For the discrete features *key* and *mode*, related in terms of tonality from a music theory perspective, these were considered together, as proposed by Bittner et al. [5], who maps *mode* and *key* into the three dimensional space \mathbb{R}^3 . Keys, i. e., the pitch classes, are mapped according to the circle of fifths and represented as points onto a two dimensional unit circle. The third dimension is added by the mode (major or minor) so that *key* and *mode* are equidistant. In Figure 1 the representation of *key* and *mode* in the three dimensional space is shown. From now on, we will refer to the combination of these two features as *tonality*.

(iii) For the overlap in artists (or genre) between two tracks, the variance is calculated by the Jaccard distance, as shown in Equation (8), where A_i and A_j represent the artist (or genre) sets of track t_i and t_j , respectively.

$$\mathcal{J}(A_i, A_j) = \frac{|A_i \cup A_j| - |A_i \cap A_j|}{|A_i \cup A_j|} \quad (8)$$

3.3 Method

To investigate the RQs, we first calculate for each playlist in the dataset the playlist variance, as defined in Section 3.2.1. The playlist variance is our baseline, which represents the variance of features irrespective of the order of the tracks. Then, the sequential variance is computed for each playlist as defined in Section 3.2.2. In contrast to the playlist variance, the sequential variance considers only tracks which lie exactly d tracks apart from each other inside the playlist.

To answer **RQ1** we choose $d = 1$, so that only features of direct neighbors, i. e., $(x_1, x_2), \dots, (x_{n-1}, x_n)$, are considered for the variance calculation. If for the majority of playlists the sequential variance is lower than the playlist variance, thus yielding a high proportional variance, i. e., above 1.0, we can conclude that users, consciously or unconsciously, create playlists with smooth transitions between tracks for the given feature under investigation. In contrast, if the sequential variance is higher than the playlist variance for a certain feature, thus yielding a low proportional variance, i. e., below 1.0, we can

conclude that users tend to prefer a more rapid change for that feature. Reasons for rapid changes can be multifarious. For instance, in playlists with the purpose of dancing, a change towards slower or different music style might be used to give listeners a recovery break.

We also investigate the effects of repeating or partially overlapping artists across tracks. Assuming that artists tend to produce tracks with similar features, sequences of tracks by the same artist might bias the variances of other features, especially when correlations between artists and features are strong. Therefore, we adapted the sequential variance and playlist variance as defined by Equation (4) and Equation (6) with the constraint of excluding subsequent tracks for which artists repeat.

To answer **RQ2** we compare the playlist variance with the sequential variance for different track distances d . This enables us to assess how the features of a given track persist on the neighboring ones in relation to the distance between them. We interpret the changes in the UPV w. r. t. different track distances as defined in Section 3.2.2 and Section 3.2.3. We also compute a series of Welch’s two-tailed t-tests between the playlist variances and sequential variances to identify how many consecutive tracks of a given track are affected w. r. t. the feature under consideration. Generally, track distance and significance are inversely proportional, i. e., when the former increases, the latter decreases. As soon as the two-tailed t-test returns a p -value larger than .001, we conclude that there is no significant difference between sequential and playlist variance.⁸

4. RESULTS AND DISCUSSION

For **RQ1** we first investigate in Section 4.1 the variation of subsequent tracks in comparison to the order-independent playlist variance. Next, in Section 4.2, we focus on the distribution of *loudness*, i. e., the audio feature with the largest UPV. For **RQ2** the number of tracks affected by the previous one, i. e., those for which properties characteristic of previous tracks still persist, are assessed in Section 4.3.

4.1 Quantitative Analysis of Proportional Variances

In Figure 2, the unconstrained and constrained proportional variances, i. e., UPV and CPV, respectively, as defined in Section 3.2.3, are shown.

The statistical analysis shows that *genre* seem to be the most important property influencing users in the selection of neighboring tracks, as displayed by the highest UPV, i. e., 1.159 (meaning that playlist variance exceeds sequential variance by 15.88 %); cf. UPV for *genre* in Figure 2. A high UPV indicates a low variance for neighboring tracks in comparison to the overall playlist variance for a given feature. However, as explained in Section 3.1, the tracks’ genres, being the union of the corresponding artists’ gen-

⁸ Note that the reported results are comparable to those obtained from the non-parametric alternatives Mann-Whitney U rank and Wilcoxon signed-rank test.

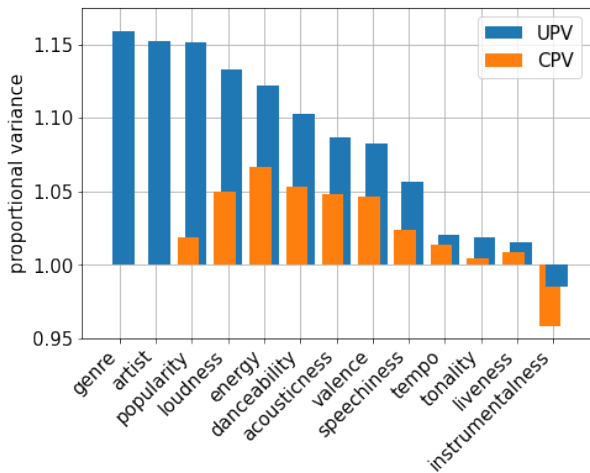


Figure 2. Bar chart representing the unconstrained proportional variances (UPV) and the constrained proportional variances (CPV) for all the analyzed features.

res,⁹ are often sparse, leading to large Jaccard distances when repeating artists are filtered out, which yields a very low CPV (cf. no visible CPV bar for *genre* in Figure 2).

The next most important properties for low sequential variance (in relation to playlist variance) are *artist* and *popularity*, as shown by their high UPV: 1.153 and 1.151, respectively (cf. UPV for *artist* and *popularity* in Figure 2). This indicates that user-generated playlists often repeat the same artists in subsequent tracks. The proportional variance for *popularity* drops considerably, i. e., -0.133 , after filtering out repeating artists: compare UPV (1.151) w. r. t. CPV (1.019) for *popularity* in Figure 2.¹⁰ We assume this is due to tracks from famous artists, which are more common in playlists, being typically more popular; therefore, repeating artists have a large effect on tracks’ popularity.

Several audio features (*loudness*, *energy*, *danceability*, *acousticness*, and *valence*) have lower but still considerable proportional variances, both in the unconstrained and the constrained setting: all of them show a $UPV \geq 1.082$ and a $CPV \geq 1.046$; while *tempo*, *tonality*, and *liveness* show $UPV \leq 1.021$, $CPV \leq 1.013$; and *speechiness* falls in between with $UPV = 1.056$, $CPV = 1.024$ (cf. UPV and CPV in Figure 2). This shows that concerning *loudness*, *energy*, *danceability*, *acousticness*, and *valence*, the majority of playlists tend to have smooth transitions between directly neighboring tracks even in cases where all artists are different from one song to another. Differently, for *tempo*, *tonality*, *liveness*, and to a lesser extent for *speechiness*, no substantial differences are displayed. Nevertheless, a deeper evaluation focusing on specific genres, such as ‘classical’ or ‘rap’, should be performed in order to understand whether the importance of these features is biased by the effect of predominant genres, e. g., ‘pop’ or ‘rock’, in which they might not have a prominent role.

Interestingly, for *instrumentalness* it can be observed

⁹ There are 5, 145 genres across the whole dataset with an average of 3.17 genres per track.

¹⁰ For obvious reasons there is no bar referring to the constrained proportion for *artist* in Figure 2.

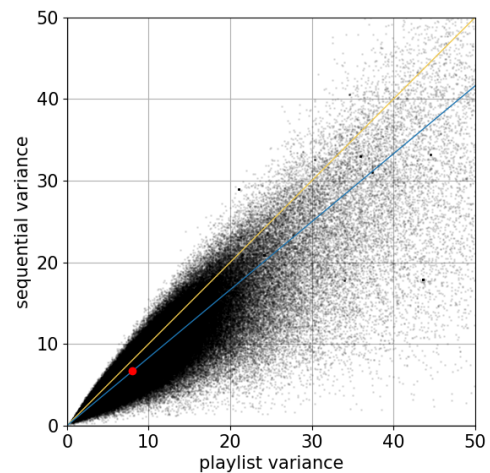


Figure 3. Scatter plot of the loudness feature. Each point represents one playlist with the playlist variance on the x-axis and the sequential variance on the y-axis. The red point marks the center of gravity. The lower line (in blue) visualizes the general differences between both variances in comparison to the line of equality (in yellow).

that the proportional variances are below the line of equality (i. e., 1.0 on the y-axis), meaning that the sequential variance is on average larger than the playlist variance. More precisely, the UPV is 0.985 (or -1.49% in relation to the playlist variance). The effect is even stronger for CPV: 0.958 (or -4.23%). This is an unexpected outcome, which might be explained by the very skewed distribution (skewness = 3.593) of this feature.

4.2 Visualization of Proportional Feature Differences

To visually explore the relationship between playlist and sequential variance over all playlists in the dataset, we represent each playlist as a point on a scatter plot with the x-axis corresponding to the playlist variance and the y-axis to the sequential variance of a chosen feature. Figure 3 shows the distribution of the feature *loudness*. We chose loudness as example as it is the audio feature with the largest UPV. For completeness, we provide the plots for all features as well as the source code to reproduce the experiments.¹¹

Figure 3 displays that the scattered points are not symmetrically distributed along the line of equality, i. e., the diagonal (upper line) considered as reference. Most of the points fall below the line of equality, as shown by the general trend of the distribution, indicated by the lower line crossing the center of mass (large dot), which has a slope of 0.83, i. e., 39.77 degrees. This indicates that directly neighboring tracks vary less arbitrary than other tracks in the playlist, in other words, there is a large imbalance between sequential and playlist variance.

Furthermore, the effect seems to be even stronger for playlists with generally large playlist variances (cf. empty area in the upper left part compared to the lower right part of Figure 3). Thus, we conclude that the majority of user-generated playlists have a smooth change in *loudness*.

¹¹ https://gitlab.cp.jku.at/haralds/spv_analysis

4.3 Proportional Variances for Increasing Track Distances

Unlike in Section 4.1, where the sequential variance was only considered for every track and its direct neighbor, in order to answer **RQ2**, we compute now the sequential variance of tracks which lie a predefined distance d apart from each other. Figure 4 depicts along the x-axis the increasing track distance considered, whereas the y-axis shows the drop in features persistence according to the UPV defined in Section 3.2.3. Dashed lines indicate non-significant results on the t-test: as significance threshold, we consider $p \leq .001$.

It can be seen that the UPV of the analyzed features, excluding *instrumentalness*, *genre*, and *artist*, drop in a similar fashion. The larger the initial UPV (i. e., the UPV at track distance $d = 1$), the longer specific characteristics of a given feature prevail on the upcoming tracks. Generally, *energy*, *loudness*, *danceability*, *acousticness*, and *valence* are properties that significantly persist on tracks which lie up to 11 tracks apart (cf. solid lines for these features in Figure 4). Differently, the UPV drop faster for *genre* and *artist* than for the audio features, which indicates that repeating artists and overlapping genres are only important for neighboring tracks lying close to each other, i. e., within a track distance of 2 or 3. Interestingly, after around 8 tracks the lines for *genre* and *artist* drop below 1.0. This suggests that after 8 tracks it is more likely that artists and genres differ than they do not.

As mentioned in Section 4.1, the audio features *tempo*, *tonality*, and *liveness* present an initial UPV ≤ 1.021 , which drops even further with increasing track distance (cf. Figure 4). Nevertheless, although these features do not generally show a high UPV for any track distance, they are still significant: especially *tonality*, whose characteristics persist even 9 tracks apart (cf. solid line for *tonality* in Figure 4). This suggest that these features might be important for specific genres or themes but not for the dominant ones, i. e., the most popular, whose weight could have hidden the role of these features for concrete genres in the investigated scenario. A similar trend (persisting up to 8 tracks) is shown for *speechiness*, falling in between audio features with high UPV and low UPV. The exact reason for the persisting significance but low UPV values is an open research question which will be investigated in future work.

The only outlier feature in this assessment is again, as expected by the findings described in Section 4.1, *instrumentalness*. Unexpectedly, the UPV continues to drop until a track distance of 8 is reached, afterwards it increases again. This might be explained by pronounced overlaps between artists or between genres, as well as by the skewed distribution (skewness = 3.6) of this feature. Investigating this behavior further will also be part of our future work.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated to which extent audio and metadata characteristics of *subsequent* tracks in user-generated playlists differ, and we related this difference

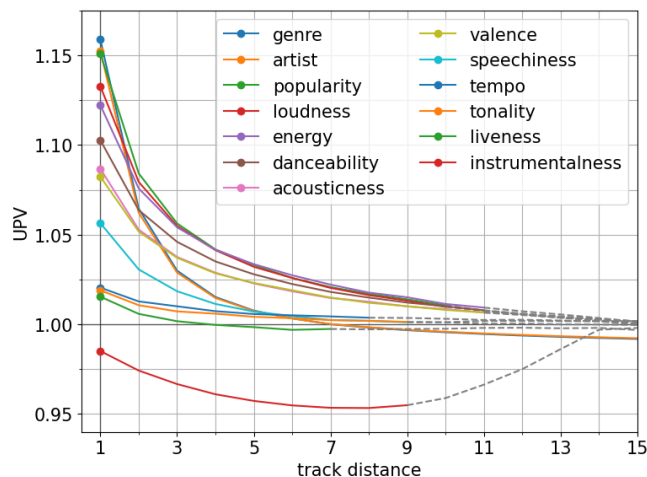


Figure 4. Visualization of the change in UPV according to track distance. The feature labels of the legend are sorted by the starting values of UVP in descending order. Gray dashed lines mark the point at which t-test between playlist and sequential variance return a p -value $\geq .001$.

to the difference of arbitrary tracks in the playlist. For this purpose, we defined variance measures on the level of subsequent tracks (sequential variance) and on the level of an entire playlist (playlist variance). Using these measures, we analyzed both direct neighbors and tracks up to a certain distance apart in the playlist. Our major findings can be summarized as follows. (i) Metadata, i. e., *genre*, *artist*, and *popularity*, vary on average by 15.10% more for the overall playlist variance than for order dependent sequential variance. (ii) The audio features *loudness*, *energy*, *danceability*, *acousticness*, and *valence* persist stronger over subsequent tracks at larger distances in the playlist than the metadata aspects *genre*, *artists*, and *popularity*. This effect is particularly pronounced for track distances ≥ 3 , and specially marked for *energy*, *danceability*, *acousticness*, and *valence*, which significantly persist on average up to 11 subsequent tracks. (iii) Filtering tracks by the same artist(s) shows similar, but less pronounced results for all features, except for *genre* and *popularity*, where the difference between playlist and sequential variance almost vanishes.

Future work will include research about the content of playlists for which very large or very small UPV values are measured. This will enable us to identify possible patterns inside playlists as well as the ‘themes’ that the creator might have had in mind. We will also focus on a more profound explanation about correlations between features and will further investigate the reasons of certain outliers, e. g., *instrumentalness*. Since we are aware that the sequential relationship between tracks for some of the evaluated features, such as *key* and *mode*, might strongly depend on the musical genre,¹² a deeper evaluation on selected musical genres will also be carried out. We will ultimately leverage our findings to improve APG and APC algorithms.

¹² For instance, in classical music the sequential relationship between pieces in terms of *tonality* is stronger than in other genres, e. g., rock.

6. ACKNOWLEDGMENTS

This research received support by the Austrian Science Fund (FWF): P33526.

7. REFERENCES

- [1] H. Zamani, M. Schedl, P. Lamere, and C. Chen, “An analysis of approaches taken in the ACM recsys challenge 2018 for automatic music playlist continuation,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 5, pp. 57:1–57:21, 2019. [Online]. Available: <https://doi.org/10.1145/3344257>
- [2] M. Volkovs, H. Rai, Z. Cheng, G. Wu, Y. Lu, and S. Sanner, “Two-stage model for automatic playlist continuation at scale,” in *Proceedings of the ACM Recommender Systems Challenge, RecSys Challenge 2018, Vancouver, BC, Canada, October 2, 2018*. ACM, 2018, pp. 9:1–9:6. [Online]. Available: <https://doi.org/10.1145/3267471.3267480>
- [3] G. Bonnin and D. Jannach, “Automated generation of music playlists: Survey and experiments,” *ACM Comput. Surv.*, vol. 47, no. 2, pp. 26:1–26:35, 2014. [Online]. Available: <https://doi.org/10.1145/2652481>
- [4] D. Jannach, L. Lerche, and I. Kamehkhosh, “Beyond “hitting the hits”: Generating coherent music playlist continuations with the right tracks,” in *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*, H. Werthner, M. Zanker, J. Golbeck, and G. Semeraro, Eds. ACM, 2015, pp. 187–194. [Online]. Available: <https://doi.org/10.1145/2792838.2800182>
- [5] R. M. Bittner, M. Gu, G. Hernandez, E. J. Humphrey, T. Jehan, H. McCurry, and N. Montecchio, “Automatic playlist sequencing and transitions,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 442–448. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/86_Paper.pdf
- [6] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer, “Playlist generation using start and end songs,” in *Proceedings of the 9th International Conference on Music Information Retrieval, ISMIR 2008, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 173–178. [Online]. Available: http://ismir2008.ismir.net/papers/ISMIR2008_143.pdf
- [7] N. Tintarev, C. Lofi, and C. C. S. Liem, “Sequences of diverse song recommendations: An exploratory study in a commercial system,” in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, July 09 - 12, 2017*, M. Bieliková, E. Herder, F. Cena, and M. C. Desmarais, Eds. ACM, 2017, pp. 391–392. [Online]. Available: <https://doi.org/10.1145/3079628.3079633>
- [8] I. Kamehkhosh, D. Jannach, and G. Bonnin, “How automated recommendations affect the playlist creation behavior of users,” in *Joint Proceedings of the ACM IUI 2018 Workshops co-located with the 23rd ACM Conference on Intelligent User Interfaces (ACM IUI 2018), Tokyo, Japan, March 11, 2018*, ser. CEUR Workshop Proceedings, A. Said and T. Komatsu, Eds., vol. 2068. CEUR-WS.org, 2018. [Online]. Available: <http://ceur-ws.org/Vol-2068/milc1.pdf>
- [9] A. Vall, M. Quadrana, M. Schedl, and G. Widmer, “Order, context and popularity bias in next-song recommendations,” *Int. J. Multim. Inf. Retr.*, vol. 8, no. 2, pp. 101–113, 2019. [Online]. Available: <https://doi.org/10.1007/s13735-019-00169-8>
- [10] —, “The importance of song context and song order in automated music playlist generation,” *CoRR*, vol. abs/1807.04690, 2018. [Online]. Available: <http://arxiv.org/abs/1807.04690>
- [11] I. Kamehkhosh, G. Bonnin, and D. Jannach, “Effects of recommendations on the playlist creation behavior of users,” *User Model. User Adapt. Interact.*, vol. 30, no. 2, pp. 285–322, 2020. [Online]. Available: <https://doi.org/10.1007/s11257-019-09237-4>
- [12] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, “Playlist prediction via metric embedding,” in *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, Q. Yang, D. Agarwal, and J. Pei, Eds. ACM, 2012, pp. 714–722. [Online]. Available: <https://doi.org/10.1145/2339530.2339643>
- [13] H. Yang, Y. Jeong, M. Choi, and J. Lee, “MMCF: multimodal collaborative filtering for automatic playlist continuation,” in *Proceedings of the ACM Recommender Systems Challenge, RecSys Challenge 2018, Vancouver, BC, Canada, October 2, 2018*. ACM, 2018, pp. 11:1–11:6. [Online]. Available: <https://doi.org/10.1145/3267471.3267482>
- [14] B. McFee and G. R. G. Lanckriet, “The natural language of playlists,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 537–542. [Online]. Available: <http://ismir2011.ismir.net/papers/PS4-11.pdf>
- [15] Z. Duan, L. Lu, and C. Zhang, “Audio tonality mode classification without tonic annotations,” in *IEEE International Conference on Multimedia and Expo. IEEE*, 2008, pp. 1361–1364.
- [16] W. Apel, *The Harvard dictionary of music*. Cambridge, MA, USA: Harvard University Press, 2003.

- [17] D. J. Grout and C. V. Palisca, *A history of Western music*. New York, NY, USA: Norton, 2001.
- [18] A. M. Sarroff and M. Casey, “Modeling and predicting song adjacencies in commercial albums,” *Proc. SMC*, 2012.
- [19] D. Kowald, P. Müllner, E. Zangerle, C. Bauer, M. Schedl, and E. Lex, “Support the underground: characteristics of beyond-mainstream music listeners,” *EPJ Data Sci.*, vol. 10, no. 1, p. 14, 2021. [Online]. Available: <https://doi.org/10.1140/epjds/s13688-021-00268-9>
- [20] E. Zangerle, M. Pichl, and M. Schedl, “User models for culture-aware music recommendation: Fusing acoustic and cultural cues,” *Trans. Int. Soc. Music. Inf. Retr.*, vol. 3, no. 1, pp. 1–16, 2020. [Online]. Available: <https://doi.org/10.5334/tismir.37>
- [21] J. S. Andersen, “Using the echo nest’s automatically extracted music features for a musicological purpose,” in *Proceedings of the 4th International Workshop on Cognitive Information Processing, CIP 2014, Copenhagen, Denmark, May 26-28, 2014*. IEEE, 2014, pp. 1–6. [Online]. Available: <https://doi.org/10.1109/CIP.2014.6844510>
- [22] M. McVicar, T. Freeman, and T. D. Bie, “Mining the correlation between lyrical and audio features and the emergence of mood,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 783–788. [Online]. Available: <http://ismir2011.ismir.net/papers/OS9-2.pdf>
- [23] Y. Zhang, H. Wu, and L. Cheng, “Some new deformation formulas about variance and covariance,” in *Proceedings of International Conference on Modelling, Identification and Control*, 2012, pp. 1042–1047.

A DIFFERENTIABLE COST MEASURE FOR INTONATION PROCESSING IN POLYPHONIC MUSIC

Simon Schwär Sebastian Rosenzweig Meinard Müller
International Audio Laboratories Erlangen, Germany

simon.schwaer@fau.de, {sebastian.rosenzweig, meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Intonation is the process of choosing an appropriate pitch for a given note in a musical performance. Particularly in polyphonic singing, where all musicians can continuously adapt their pitch, this leads to complex interactions. To achieve an overall balanced sound, the musicians dynamically adjust their intonation considering musical, perceptual, and acoustical aspects. When adapting the intonation in a recorded performance, a sound engineer may have to individually fine-tune the pitches of all voices to account for these aspects in a similar way. In this paper, we formulate intonation adaptation as a cost minimization problem. As our main contribution, we introduce a differentiable cost measure by adapting and combining existing principles for measuring intonation. In particular, our measure consists of two terms, representing a tonal aspect (the proximity to a tonal grid) and a harmonic aspect (the perceptual dissonance between salient frequencies). We show that, combining these two aspects, our measure can be used to flexibly account for different artistic intents while allowing for robust and joint processing of multiple voices in real-time. In an experiment, we demonstrate the potential of our approach for the task of intonation adaptation of amateur choral music using recordings from a publicly available multitrack dataset.

1. INTRODUCTION

The widely-used 12-tone equal temperament (12-TET) tuning system divides the octave in twelve equal semitones of the ratio $2^{1/12} \approx 1.0595$. This allows instruments with fixed pitch to play in any key at the cost of most intervals being slightly out of tune in comparison to the natural overtone spectrum of harmonic sounds. Just Intonation (JI) scales, on the other hand, are constructed from intervals with small integer ratios to a root note. As a result, the harmonic overtones of two tones in a JI scale are more congruent than those in 12-TET. However, the absolute pitches in the JI scale change for different root notes, so that the grid must be adapted to different keys and musical contexts.

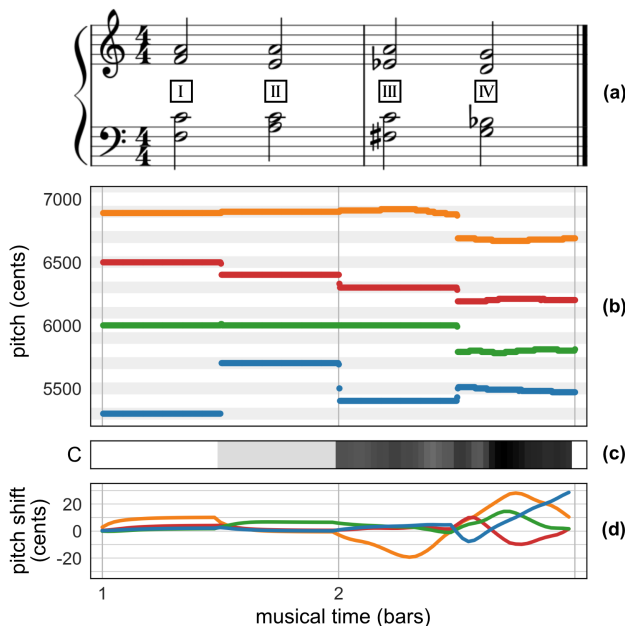


Figure 1: Joint adaptation of the voices in an example cadence. (a) Sheet music. (b) Fundamental frequencies of the original synthesized voices measured with pYIN [1] (orange: soprano, red: alto, green: tenor, blue: bass). (c) Overall intonation cost C between all voices with $w = 0.33$. (d) Pitch shift curve for all voices obtained from joint gradient descent on C with $w = 0.33$ and $\mu = 350$.

Many instruments can produce any pitch in between the 12-TET or JI grid or have considerable variance in tuning. This allows performers to dynamically change the sounding scale or chord, both intentionally and accidentally. This flexible intonation is particularly relevant in a cappella choral singing. While 12-TET is often used as an approximation for the distribution of chosen pitches by singers [2, 3], their intonation is influenced by a multitude of aspects. For example, choir singers tend to aim for JI in harmonies [4], whereas other influences may prevail in melodic or solistic phrases [5]. At the same time, singers continuously have to account for the intonation of their fellow musicians [6, 7], while pitch changes also occur during the sounding tone [8]. Depending on the singers' ability to control their voices, this complex setting often results in defects like poor local intonation or intonation drift [9, 10].

Different aspects of intonation are illustrated in the synthesized example cadence shown in Figure 1: The pitches



(where “pitch” is used as a technical term synonymously with fundamental frequency here and in the following) in chord I correspond to a JI scale with root F, while chord II is tuned to 12-TET. Continuous deviations are illustrated in chords III and IV, where the soprano in III and all voices in IV are detuned randomly between -25 and $+25$ cents around the 12-TET grid. Even with these large pitch fluctuations, the chords can still be clearly recognized¹.

When post-processing multitrack recordings, pitch-shifting the individual audio signals may mitigate some unintended intonation deviations in a performance. However, this requires a known target pitch, and similarly to intonation in a live performance, the desired target can be influenced by many aspects. Instead of quantizing to a fixed set of pitches like 12-TET or manually tweaking individual notes, we formulate intonation adaptation as a cost minimization problem. A good cost measure for this task should have a local minimum at the target pitch for each individual note.

As our main contribution, we propose a differentiable cost measure, where the local minima can be adjusted according to artistic intent. In particular, we employ two existing models to account for different aspects of intonation: The first model represents a *tonal* aspect, that most music is composed from a set of discrete pitches approximated by equal divisions of the octave [11]. The second model considers a *harmonic* aspect and uses perceptual dissonance to capture the tendency for JI in multi-part harmonies [12].

We show that our cost-based approach has several advantageous properties over existing methods for intonation processing:

- A variable weight between the terms allows for flexibly setting the local minimum anywhere between 12-TET and JI.
- In contrast to purely dissonance-based adaptation, our cost measure has a local minimum also for musically unstable voices of a chord.
- Using gradient descent, intonation can be adapted in real-time, dynamically reacting to changing inputs.

For example, the overall cost shown in Figure 1c is high when voices deviate strongly from the desired pitches. At the same time, musically dissonant chords like the diminished seventh chord in III have a higher inherent perceptual dissonance. Therefore, an adaptation should not aim to achieve zero cost, but to find the nearest local minimum.

Figure 1d shows the pitch shift curves for the voices in our example that locally minimize the cost measure. The curves were obtained using joint gradient descent, where all voices are processed at the same time and influence each other. The resulting “optimal” pitches after applying the shift lie in between 12-TET and JI, as can be seen e. g. in the major third of chord I. Its initial pitch in the present example is -14 cents w.r.t. 12-TET and it is pitched up by 10 cents to minimize the cost with the given parameters. Furthermore, the algorithm finds meaningful solutions in

the more complex situations occurring in chords III and IV.

The remainder of this article is structured as follows. In Section 2, we review existing approaches to intonation adaptation and adaptive tuning, in Section 3 we introduce the cost measure, and in Section 4, we demonstrate the applicability to local intonation adaptation in a multitrack choral music recording with amateur singers.

2. RELATED WORK

A common intonation adaptation strategy implemented in many commercial products like *Melodyne* [13] or *Auto-Tune* [14] is to measure the fundamental frequency (F0) in a monophonic recording and to pitch-shift the signal such that the F0 approaches a fixed target value. The target can be chosen manually by the user or determined automatically from a predefined grid or score.

Several approaches have been proposed to dynamically choose a target pitch based on musical assumptions. Rule-based algorithms like *Groven.Max* [15] or *Hermode Tuning* [16] choose pitches for all voices of a synthesizer by analyzing the musical structure of a chord. Aiming for JI, they implement fixed rules to compromise in chords where just intervals between all pairs of notes are not possible. This problem can also be addressed by solving a quadratic program [17]. This way, the deviation from JI is distributed evenly across the pitches and all intervals are as close as possible to a small integer ratio. Additional constraints can enable temporal continuity.

Deep learning is used in [18] to infer “good” intonation from curated training examples of monophonic vocal recordings over a backing track. The model then outputs a pitch shift curve that can match the intonation in an input recording with the characteristics of the training examples.

Sethares [19] relates the chosen scale to the timbre of the sound. Summing the perceptual dissonance [20] of all individual salient frequency pairs between two sounds, he obtains a *dissonance landscape*, in which local minima exist for small integer ratio intervals if the timbre is harmonic. This principle is also used for adaptive tuning using gradient descent [12], which achieves a tuning similar to JI without requiring explicit musical analysis of the chords. In [21], the idea was further enhanced to be stable in more complex settings by adding a proximity constraint. This limits the deviation from 12-TET to a few cents and requires the input to be in the same range.

3. INTONATION COST MEASURE

Our cost measure is based on the assumption that proper intonation in a polyphonic context is a balance between the proximity to pitches in an equal temperament tuning system most suitable for the composition and the minimization of perceptual dissonance. Mathematically, this can be expressed as the sum of a *tonal* cost C_t , indicating the distance to the pitches in an equal temperament tuning system, and a *harmonic* cost C_h , measuring the perceptual dissonance in the overall sound:

$$C := w \cdot C_t + (1 - w) \cdot C_h. \quad (1)$$

¹ Audio examples are available online:

<https://www.audiolabs-erlangen.de/resources/MIR/2021-ISMIR-IntonationCostMeasure>.

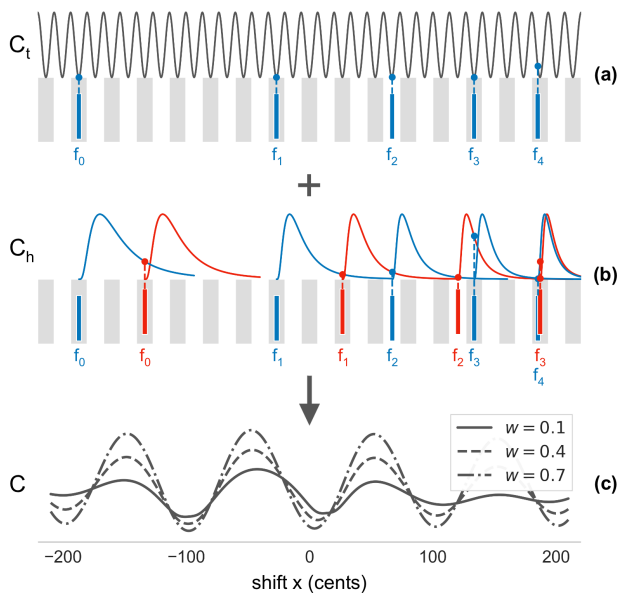


Figure 2: Conceptual overview of our cost measure. **(a)** C_t is higher when salient frequencies are far from the equal temperament grid. **(b)** C_h is higher when salient frequencies of a tone (blue) are similar but not equal to salient frequencies of a concurrent reference tone (red). **(c)** Shifting the tone shown in blue by x cents changes the overall cost C . A higher relative weight w of C_t to C_h results in local minima closer to the 12-TET grid.

A lower cost C corresponds to a “better” choice for the pitch, where the parameter $w \in [0, 1]$ controls the relative weight between the two aspects.

Figure 2 exemplifies the behavior of C for two hypothetical tones with five harmonic partials each (depicted in blue and red). The graphs in Figure 2c show the change in C when the tone represented in blue is pitch-shifted by -200 to $+200$ cents. As the two tones form a major third, a shift by $+14$ cents would result in a just interval. With a larger w , the relative weight of C_t increases, corresponding to a preference for equal temperament, whereas with decreasing w , the local minima move closer to JI intervals.

In the upcoming section, we develop differentiable expressions for C_t and C_h and illustrate their properties by continuing our example from Figure 1. In Section 3.4, we then show how the cost measure can be used to adapt the intonation using gradient descent.

3.1 Prerequisites

For a given audio signal, we assume a stationary sound in each analysis time frame n and represent it by a set

$$\mathcal{P}[n] := \{(f_m, a_m) \mid m \in \{1, \dots, M\}\}, \quad (2)$$

consisting of M salient frequencies f_m in Hz with amplitude a_m . The cost measure is defined for a signal w.r.t. to a reference (or “background”) signal represented by a set $\mathcal{P}_{\text{ref}}[n]$. In the following, we omit the frame index for \mathcal{P} and \mathcal{P}_{ref} where the time-dependency is not relevant.

To obtain this representation from audio signals, we use a short-time Fourier transform (STFT) with a frame and

hop size of 0.1 sec and detect peaks in the magnitude spectrum of each time frame that constitute the salient frequencies. Avoiding the misinterpretation of transient peaks in the spectrum, we filter the STFT representation to remove percussive components of the signal [22]. Then, we identify up to 16 peaks in the remaining spectrum of each time frame by selecting the local maxima above a threshold. To increase frequency resolution, we interpolate the exact peak frequency and amplitude by fitting a parabola to the magnitudes of neighboring bands [23]. For an inactive voice or a purely percussive signal frame, we set $\mathcal{P} = \emptyset$.

Note that, for harmonic sounds in a monophonic signal, all salient frequencies in \mathcal{P} are close to integer multiples of the lowest frequency f_0 . This assumption does not hold for inharmonic sounds and polyphonic recordings.

For the example in Figure 1, we synthesize the signals using a sawtooth waveform with 16 harmonic partials and amplitudes $a_i = 1/(i\pi)$ using a reference frequency of 440 Hz for A4.

3.2 Tonal Cost

Equal divisions of the octave are a good approximation for the distribution of pitches in many music theories [11]. Furthermore, measuring the distance to an equal temperament grid is an often used strategy to assess the intonation in a performance [3, 24].

We define the tonal distance $d_t^K(f_1, f_2)$ between two positive frequencies f_1 and f_2 in Hz on a K-TET grid as

$$d_t^K(f_1, f_2) := \frac{1}{2} \left(1 - \cos(2\pi K \log_2(f_1/f_2)) \right), \quad (3)$$

where the distance is small if the interval between f_1 and f_2 is close to a K-TET interval (i. e., $f_1/f_2 \approx 2^{k/K}$ with $k \in \mathbb{Z}$). By measuring the tonal distance of each frequency in \mathcal{P} to a given reference frequency f_{ref} , we define the tonal cost C_t as

$$C_t := \frac{\sum_{(f,a) \in \mathcal{P}} a \cdot d_t^K(f, f_{\text{ref}})}{\sum_{(f,a) \in \mathcal{P}} a}, \quad (4)$$

where frequencies with higher amplitude contribute more to C_t . The highest cost $C_t = 1$ is reached, when all salient frequencies lie exactly in the middle between two frequencies on the equal temperament grid defined by K and f_{ref} . The parameters K and f_{ref} can either be estimated from \mathcal{P}_{ref} or fixed to known values. Note that, with fixed parameters, C_t does not depend on \mathcal{P}_{ref} . Furthermore, Figure 2a shows that for integer-multiple frequencies in \mathcal{P} , not all frequencies can align with the grid, so that C_t is never 0 for such signals. However, the local minimum is reached when the loudest partials are close to the grid.

For the example, let \mathcal{P} include the salient frequencies of the soprano voice while all other voices are contained in \mathcal{P}_{ref} . By setting $K = 12$ and $f_{\text{ref}} = 440$ Hz, we obtain the cost heatmap shown in Figure 3. By pitch-shifting the soprano signal by -200 to 200 cents and evaluating the cost for each time frame, it shows for which shifts the salient frequencies in the signal fit best on the 12-TET grid. Tracking the nearest local minimum starting at a shift of 0

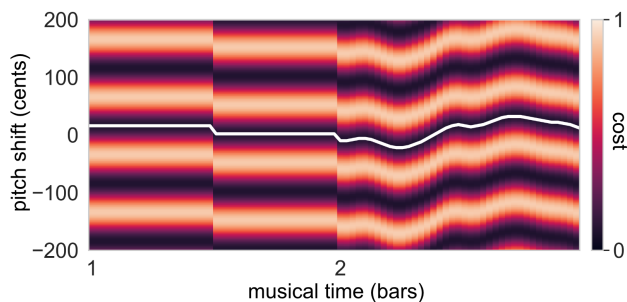


Figure 3: Tonal cost heatmap for the soprano voice in the example from Figure 1, pitch-shifted by -200 to $+200$ cents. White line indicates local minimum closest to 0.

cents, the white line corresponds to the pitch shift required to minimize the cost.

3.3 Harmonic Cost

The perceptual dissonance between concurrent sounds can be expressed in terms of the *pure-tone* dissonance between all combinations of salient frequencies present in each sound [19]. While the perceived dissonance between two pure tones was first determined experimentally [20], we quantify the dissonance between two positive frequencies f_1 and f_2 using the parametrized model from [11] (omitting a global scaling factor):

$$d_h(f_1, f_2) := \exp\left(-\ln^2\left(\frac{|\log_2(f_1/f_2)|}{w_c}\right)\right), \quad (5)$$

with $w_c := 6.7 \cdot \min(f_1, f_2)^{-0.68}$ as the frequency-dependent parameter that controls the interval of maximal dissonance and the decay of the dissonance curve. To ensure differentiability, we define $d_h(f_1, f_2) := 0$ for $f_1 = f_2$.² As illustrated in Figure 5, $d_h(f_1, f_2)$ approaches 0 from both sides when f_1 is close to f_2 . $d_h(f_1, f_2) = 1$ is maximal when the logarithmic distance between the two frequencies is $|\log_2(f_1/f_2)| = w_c$.

The sum of $d_h(f_1, f_2)$ between all pairs of salient frequencies in \mathcal{P} and \mathcal{P}_{ref} weighted by the amplitude constitutes the harmonic cost:

$$C_h := \frac{\sum_{(f,a) \in \mathcal{P}} \sum_{(f_r, a_r) \in \mathcal{P}_{\text{ref}}} \min(a, a_r) \cdot d_h(f, f_r)}{\sum_{(f,a) \in \mathcal{P}} a} \quad (6)$$

The normalization does not restrict C_h to $[0, 1]$, because the total number of salient frequency pairings is $|\mathcal{P}| \cdot |\mathcal{P}_{\text{ref}}|$, but when comparing harmonic signals, only a small fraction of pairings have $d_h(f_1, f_2) \gg 0$. By normalizing by the sum of amplitudes in \mathcal{P} , we achieve a comparable range for C_t and C_h , where C_h vanishes when the amplitudes in \mathcal{P}_{ref} are very small (i. e. no reference signal is present for which a harmonic relation can be evaluated).

The concept of the harmonic cost is illustrated in Figure 2b, where only the pairings between sets (blue and red) contribute to the cost. With the frequency-dependent w_c ,

² Proof of differentiability can be found on the accompanying website.

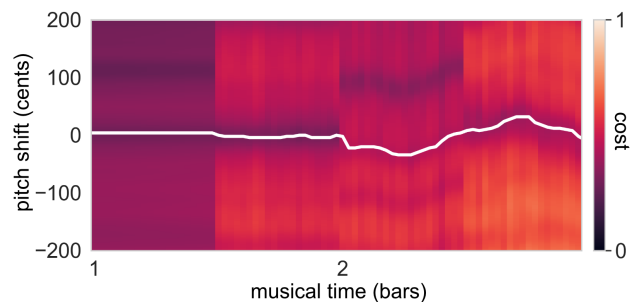


Figure 4: Harmonic cost heatmap for the soprano voice in the example from Figure 1, pitch-shifted by -200 to $+200$ cents. White line indicates local minimum closest to 0.

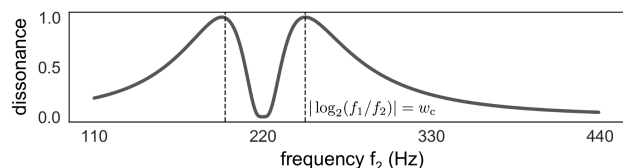


Figure 5: Pure-tone dissonance $d_h(f_1, f_2)$ from [11] with $f_1 = 220$ Hz and f_2 between 110 and 440 Hz.

the dissonance curve becomes more narrow towards higher frequencies. Applied to the soprano in our example analogous to Figure 3, this results in the heatmap in Figure 4.

3.4 Joint Intonation Adaptation

In a musical performance, the cost C may vary between time frames n and we denote the cost in each frame by $C[n]$. The goal of intonation adaptation is to obtain a pitch shift function $p : \mathbb{Z} \rightarrow \mathbb{R}$, where a pitch shift of $p[n]$ cents applied to the considered signal minimizes $C[n]$.

When multiple voices can be adapted simultaneously in a polyphonic multitrack setting, the cost for each individual voice depends on the salient frequencies in the other voices. We denote the cost for each voice v with respect to all other voices by $C_v[n]$ and the current pitch shift for the signal of v by $p_v[n]$. Then the optimal shift can be found by solving

$$\min_{p_v[n]} C_v[n]. \quad (7)$$

As described in [12], gradient descent is an effective method to find the local minimum. Pitch-shifting with $p_v[n]$ affects the salient frequencies in $\mathcal{P}[n]$ equally on a logarithmic scale while the frequencies in $\mathcal{P}_{\text{ref}}[n]$ stay constant. Thus, the shift p in cents can be moved out of the logarithm in the tonal distance $d_t^K(f_1, f_2)$ and the dissonance $d_h(f_1, f_2)$, for which we introduce auxiliary functions $\delta_t^K(f_1, f_2, p)$ and $\delta_h(f_1, f_2, p)$:

$$\begin{aligned} \delta_t^K(f_1, f_2, p) &:= \frac{1}{2} \left(1 - \cos(2\pi K(\log_2(f_1/f_2) + p/1200))\right) \\ \delta_h(f_1, f_2, p) &:= \exp\left(-\ln^2\left(\frac{|\log_2(f_1/f_2) + p/1200|}{w_c}\right)\right) \end{aligned} \quad (8)$$

In the following, we omit the arguments of the distance and dissonance functions for brevity. Analogous to (5),

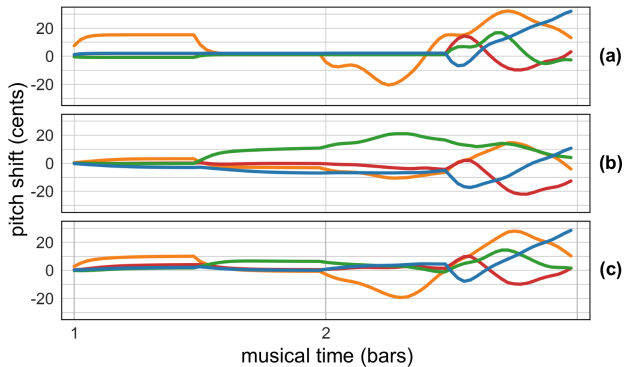


Figure 6: Adaptation curves $p_v[n]$ resulting from joint gradient descent with all four voices (orange: soprano, red: alto, green: tenor, blue: bass, $\mu = 350$). (a) $w = 1.0$ (b) $w = 0.0$ (c) $w = 0.33$

we define $\delta_h := 0$ for $\log_2(f_1/f_2) = -p/1200$ to retain differentiability. Furthermore, we assume for δ_h that w_c stays constant for small shifts p , so that its frequency-dependency does not play a role in a single gradient descent step. Replacing d_t^K with δ_t^K and d_h with δ_h in (4) and (6) allows calculating the derivative of $C_v[n]$ directly with respect to p , so that the update rule becomes

$$p_{v,\text{new}}[n] = p_v[n] - \mu \frac{dC_v[n]}{dp}, \quad (9)$$

where μ is a step size parameter and the derivative of $C_v[n]$ (with the unit “cost change per cent shifted”) is a weighted sum of $\frac{d\delta_t^K}{dp}$ and $\frac{d\delta_h}{dp}$:

$$\frac{d\delta_t^K}{dp} = \frac{\pi K}{1200} \sin\left(2\pi K(\log_2(f_1/f_2) + p/1200)\right) \quad (10)$$

$$\frac{d\delta_h}{dp} = -\frac{\ln\left(\frac{|\log_2(f_1/f_2) + p/1200|}{w_c}\right)}{600(\log_2(f_1/f_2) + p/1200)} \exp\left(-\ln^2\left(\frac{|\log_2(f_1/f_2) + p/1200|}{w_c}\right)\right). \quad (11)$$

with $\frac{d\delta_h}{dp} = 0$ for $\log_2(f_1/f_2) = -\frac{p}{1200}$. By setting $p_v[n]$ to an initial value (e. g. 0) and repeatedly evaluating (9), we can now iteratively find the local minimum of $C_v[n]$ for each time frame. However, for short frame sizes, correlation between salient frequencies in successive time frames can be expected. Therefore, instead of finding the closest local minimum for each frame independently, we can use the pitch shift from the previous frame as the initial value for $p_v[n]$. Furthermore, to retain natural short-term pitch variations in the signal (e. g. vibrato), we require a certain temporal smoothness of $p_v[n]$. This can be achieved by updating $p_v[n]$ with only a single gradient descent step in each time frame, which yields

$$p_v[n] = p_v[n-1] - \mu \frac{dC_v[n]}{dp} \quad (12)$$

for $n > 0$ and $p_v[0] = 0$. Together with the frame size, the step size μ controls the rate at which pitch changes in

the signals influence $p_v[n]$. This can be observed in Figure 6, which shows the resulting $p_v[n]$ from joint gradient descent with (12) for a varying weight w between C_t and C_h in the example cadence. For example, the pitch shift for the soprano voice visibly approaches a minimum in the first few frames of chords I and II. Moreover, it can be seen that the harmonic cost alone (Figure 6b) is not robust in musically dissonant chords like chord III, whereas with $w = 0.33$ (Figure 6c), the obtained pitch shift tends towards II without ending up in a local minimum far away from equal temperament (cf. tenor in chords III and IV).

4. APPLICATION: INTONATION ADAPTATION IN CHORAL MUSIC

In the previous section, we introduced a method to obtain pitch shift curves by minimizing a cost measure that quantifies two aspects of intonation: the distance of a pitch to an equal temperament grid and the perceptual dissonance with regard to a harmonic reference. As a tool, this allows sound engineers to flexibly adapt the intonation in audio recordings between equal temperament and JI using the two parameters w and μ . With w , the relative weight between both aspects can be adjusted depending on artistic intent and musical context. μ controls the temporal behavior of the adaptation, where a larger μ corresponds to a stronger reaction to short-term pitch fluctuations.

A subjective evaluation of preferred intonation in different musical contexts and the resulting suitable choices for the parameters of our cost measure is beyond the scope of this paper. Many additional aspects, including timbre, acoustics, and performative choices (vibrato, portamento, etc.) [25], as well as listener taste and experience [26], influence intonation perception. Instead, we demonstrate the utility of the cost-based adaptation tool with an example from amateur performances of a cappella choral music. In this application with particularly volatile intonation, we show that the approach is robust on real-world signals and can blindly achieve results that are comparable to score-informed intonation adaptation.

For this, we apply the presented method to recordings from the *Dagstuhl ChoirSet* (DCS) [27]. The intonation adaptation of individual voices in a vocal recording requires separate signals for each voice and the dataset contains headset microphone signals for each singer. In this section, we consider the last four bars (45 to 48) from a performance of the motet *Locus Iste* (WAB 23, 1869) by Anton Bruckner (Quartet B, Take 3 in the dataset). The four-part a cappella composition is performed by a quartet of soprano (S), alto (A), tenor (T) and bass (B).

First, we obtain the salient frequencies for each voice from the four individual headset microphone signals, using the method described in Section 3.1. For the STFT, we keep the hop size of 0.1 sec (2205 samples in the DCS audio signals) and use a window size of 4096 samples for an improved frequency resolution. Due to varying levels and timbre of the singing voice and background noise, the number of salient frequencies in $\mathcal{P}[n]$ fluctuates. On average, $|\mathcal{P}[n]|$ is 6.6 (S: 5.6, A: 6.8, T: 4.4, B: 9.6) in

the voiced frames (i.e., where $|\mathcal{P}[n]| > 0$). To assess the robustness of the detected salient frequencies against crosstalk and noise, we calculate the average deviation of each frequency from being an integer multiple of the lowest frequency in this frame (corresponding to the “harmonicity” of the signal). In the excerpt, the detected frequencies deviate from the harmonic overtones by a factor of 1.01 on average (S: 1.007, A: 1.012, T: 1.006, B: 1.016).

We now compute a pitch shift $p_v^{\text{CB}}[n]$ for each voice with the cost-based (CB) method as described in Section 3.4, using a single gradient descent step for each frame (see (12)). For the present example, we set $w = 0.2$ and $\mu = 350$ and used fixed parameters $K = 12$ and $f_{\text{ref}} = 440$ Hz for the tonal cost. The pitch shifts $p_v^{\text{CB}}[n]$ are applied to each signal with a time-variant pitch shift algorithm based on resampling and time-scale modification [28]. In a cappella performances, one often observes (downward) intonation drifts [10], causing $p_v^{\text{CB}}[n]$ to drift in the opposite direction to counteract this effect. For instance, the mean pitch shift across all voices in bar 48 of our example is 21 cents. Note that, to counteract a global drift of all voices in a similar direction, even $|p_v^{\text{CB}}[n]| > 50$ cents may be intended. In this case, additional regularization can be added in the cost minimization to avoid individual voices ending up in local minima that do not reflect the relative intonation in the performance.

For the comparison of our method with a score-informed baseline (BL) approach, we estimate the F0 trajectories for each voice with pYIN [1] and assign the measurements to individual notes from the aligned score annotation provided in DCS. Then, for each time frame of 0.1 sec duration, we choose a pitch shift $p_v^{\text{BL}}[n]$ that shifts the median F0 in the current frame onto the 12-TET pitch of the corresponding note in the score. To counteract larger fluctuations that result from the relatively small frame size for this method, we additionally smooth $p_v^{\text{BL}}[n]$ using a moving average with a window size of 3 frames. The shift is applied to the signals in the same way as $p_v^{\text{CB}}[n]$.

The pitch shift curves for the excerpt, calculated with the blind CB approach (colored) and the score-informed BL (black), are plotted for all voices in Figure 7a, c, e, and g. Furthermore, subfigures b, d, f, and h show the F0 trajectories of the original (black) and adapted (CB: colored, BL: grey) signals. The difference between the two pitch shift curves is small in most frames, particularly when compared to the magnitude of overall pitch fluctuations in the singing voices. Larger differences between the curves can be observed at the onset of some notes. This can be attributed to the strong influence of short-term fluctuations in the measured F0 trajectories on the BL approach.

In addition, the harmonic cost term C_h has a recognizable effect on the local minimum where JI intervals differ from 12-TET. This can be prominently observed in the soprano voice in bar 48 (c.f. the zoomed detail in Figure 7), where the sung note E is the major third of the final C major chord of the piece and therefore has a JI pitch 14 cents lower than 12-TET. This shows that our real-time capable method for cost-based intonation adaptation is able

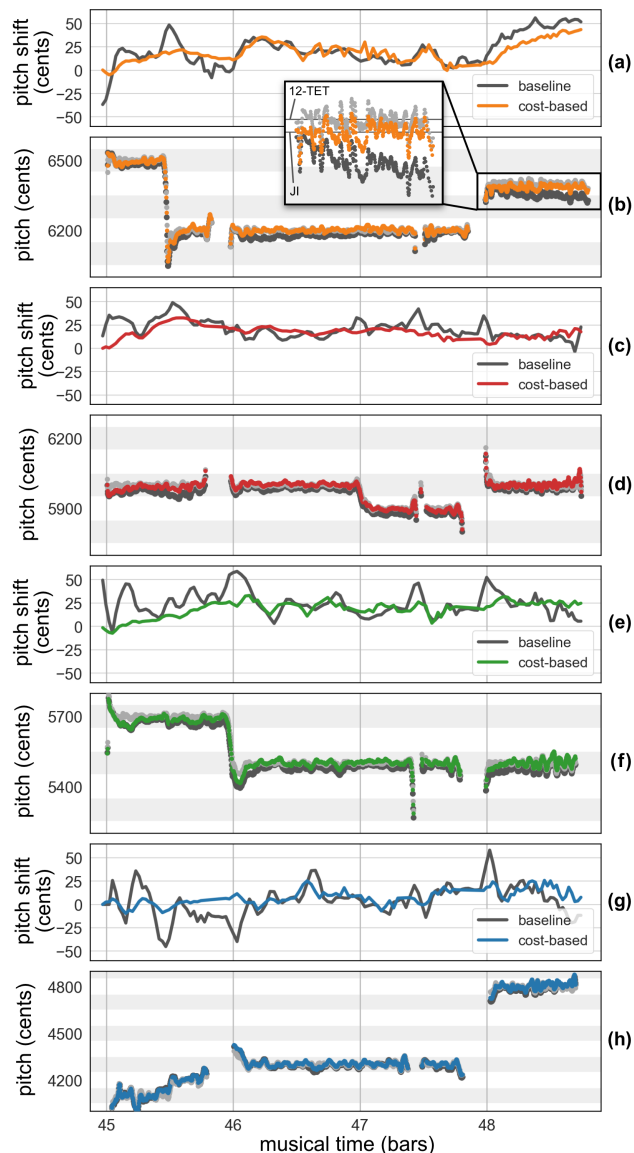


Figure 7: Joint adaptation of bars 45-48 of A. Bruckner “Locus Iste” (DCS, Quartet B, Take 3). The F0 trajectory plots (b,d,f,h) show the F0 of the original signal (black), the baseline (BL, grey) and the cost-based (CB, colored, $w = 0.2$, $\mu = 350$) pitch-shifted signals. (a, b) Soprano (c, d) Alto (e, f) Tenor (g, h) Bass

to approach JI tuning in vocal recordings without explicit knowledge about scales and keys.

5. CONCLUSION

In this paper, we introduced a differentiable cost measure for intonation processing in polyphonic music recordings, which accounts for a tonal and a harmonic aspect in a user-specified proportion. Our method can be used as a flexible tool for intonation adaptation in multitrack choral music recordings. In future work, we will investigate the perceptual implications of our intonation adaptation in real-world signals. Furthermore, we want to apply this principle to more intonation processing tasks such as adaptive tuning of synthesizers and explore ways to incorporate additional aspects of intonation in the cost measure.

Acknowledgements: This project is supported by the German Research Foundation (DFG MU 2686/12-1, MU 2686/13-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS.

6. REFERENCES

- [1] M. Mauch and S. Dixon, “pYIN: A fundamental frequency estimator using probabilistic threshold distributions,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 659–663.
- [2] M. Mauch, K. Frieler, and S. Dixon, “Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory,” *Journal of the Acoustical Society of America*, vol. 136, no. 1, pp. 401–411, 2014.
- [3] C. Weiß, S. J. Schlecht, S. Rosenzweig, and M. Müller, “Towards measuring intonation quality of choir recordings: A case study on Bruckner’s Locus Iste,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 276–283.
- [4] J. Devaney, M. I. Mandel, and I. Fujinaga, “A study of intonation in three-part singing using the automatic music performance analysis and comparison toolkit (AMPACT),” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012, pp. 511–516.
- [5] F. Havrøy, ““You Cannot Just Say: “I am Singing the Right Note””. Discussing intonation issues with Neue Vocalsolisten Stuttgart,” *Music & Practice*, vol. 1, 2013.
- [6] A. Grell, J. Sundberg, S. Ternström, M. Ptok, and E. Altenmüller, “Rapid pitch correction in choir singers,” *The Journal of the Acoustical Society of America*, vol. 126, no. 1, pp. 407–413, 2009.
- [7] J. Dai and S. Dixon, “Singing together: Pitch accuracy and interaction in unaccompanied unison and duet singing,” *The Journal of the Acoustical Society of America*, vol. 145, no. 2, pp. 663–675, 2019.
- [8] —, “Intonation trajectories within tones in unaccompanied soprano, alto, tenor, bass quartet singing,” *The Journal of the Acoustical Society of America*, vol. 146, no. 2, pp. 1005–1014, 2019.
- [9] P.-G. Alldahl, *Choral Intonation*. Gehrmans Musikförlag, 1990.
- [10] D. M. Howard, “Intonation drift in a capella soprano, alto, tenor, bass quartet singing with key modulation,” *Journal of Voice*, vol. 21, no. 3, pp. 300–315, 2007.
- [11] J. Berezovsky, “The structure of musical harmony as an ordered phase of sound: A statistical mechanics approach to music theory,” *Science Advances*, vol. 5, p. eaav8490, May 2019.
- [12] W. Sethares, “Adaptive tunings for musical scales,” *The Journal of the Acoustical Society of America*, vol. 96, 1994.
- [13] Celemony, “Melodyne,” <https://www.celemony.com/en/melodyne/>, accessed: 2021-05-07.
- [14] Antares, “Auto-Tune,” <https://www.antarestech.com/>, accessed: 2021-05-07.
- [15] D. Code, “Groven.Max: An adaptive tuning system for MIDI pianos,” *Computer Music Journal*, vol. 26, pp. 50–61, 2002.
- [16] W. Mohrlock, “Hermod Tuning,” http://www.hermod.com/index_en.html, accessed: 2021-05-07.
- [17] K. Stange, C. Wick, and H. Hinrichsen, “Playing music in just intonation: A dynamically adaptive tuning scheme,” *Computer Music Journal*, vol. 42, no. 3, pp. 47–62, 2018.
- [18] S. Wager, G. Tzanetakis, C. Wang, and M. Kim, “Deep autotuner: A pitch correcting network for singing performances,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Barcelona, Spain: IEEE, 2020, pp. 246–250.
- [19] W. A. Sethares, “Local consonance and the relationship between timbre and scale,” *Journal of the Acoustical Society of America*, vol. 94, no. 3, pp. 1218–1228, 1993.
- [20] R. Plomp and W. J. M. Levelt, “Tonal consonance and critical bandwidth,” *Journal of the Acoustical Society of America*, vol. 38, no. 4, pp. 548–560, 1965.
- [21] J. Villegas and M. Cohen, “Roughness minimization through automatic intonation adjustments,” *Journal of New Music Research*, vol. 39, pp. 75 – 92, 2010.
- [22] D. FitzGerald, “Harmonic/percussive separation using median filtering,” in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Graz, Austria, September 2010, pp. 246–253.
- [23] J. O. Smith and X. Serra, “PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation,” in *Proceedings of the International Computer Music Conference (ICMC)*. Computer Music Association, 1987.
- [24] T. Nakano, M. Goto, and Y. Hiraga, “An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features,” in *Proceedings of the Annual Conference of the International Speech Communication Association (INTER-SPEECH)*, Pittsburgh, PA, USA, 2006, pp. 1706–1709.

- [25] J. M. Geringer, R. B. MacLeod, C. K. Madsen, and J. Napoles, “Perception of melodic intonation in performances with and without vibrato,” *Psychology of Music*, vol. 43, no. 5, pp. 675–685, 2015.
- [26] F. Loosen, “The effect of musical experience on the conception of accurate tuning,” *Music Perception*, vol. 12, no. 3, pp. 291–306, 1995.
- [27] S. Rosenzweig, H. Cuesta, C. Weiß, F. Scherbaum, E. Gómez, and M. Müller, “Dagstuhl ChoirSet: A multitrack dataset for MIR research on choral singing,” *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 98–110, 2020.
- [28] S. Rosenzweig, S. Schwär, J. Driedger, and M. Müller, “Adaptive pitch-shifting with applications to intonation adjustment in a cappella recordings,” to appear in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Vienna, Austria, 2021.

IMPROVING MUSIC PERFORMANCE ASSESSMENT WITH CONTRASTIVE LEARNING

Pavan Seshadri

Center for Music Technology
Georgia Institute of Technology
pseshadri9@gatech.edu

Alexander Lerch

Center for Music Technology
Georgia Institute of Technology
alexander.lerch@gatech.edu

ABSTRACT

Several automatic approaches for objective music performance assessment (MPA) have been proposed in the past, however, existing systems are not yet capable of reliably predicting ratings with the same accuracy as professional judges. This study investigates contrastive learning as a potential method to improve existing MPA systems. Contrastive learning is a widely used technique in representation learning to learn a structured latent space capable of separately clustering multiple classes. It has been shown to produce state of the art results for image-based classification problems. We introduce a weighted contrastive loss suitable for regression tasks applied to a convolutional neural network and show that contrastive loss results in performance gains in regression tasks for MPA. Our results show that contrastive-based methods are able to match and exceed SoTA performance for MPA regression tasks by creating better class clusters within the latent space of the neural networks.

1. INTRODUCTION

Within the context of western classical music, musical performances are a sonic interpretation of a written musical score. Performers are tasked with interpreting the score and translating it to an acoustic rendition. In doing so, they craft a unique performance by controlling and varying performance parameters such as tempo and timing, dynamics, intonation, and tone quality [1]. These performance parameters and their variation impact the way in which listeners perceive the music, and let them distinguish between performances of the same musical score [2].

For music performers, the journey to competence and mastery often spans years of practice and tailored instruction. As music performance is an inherently complex and subjective task, proper feedback on performances is imperative to growth as a performer, and such, regular feedback by professional musicians is necessary. Music teachers are expected to evaluate students on various criteria such as

musicality or tone quality, although rating these criteria is highly subjective, complicating the task of consistent and objective Music Performance Assessment (MPA) [3, 4]. These challenges, however, do not reduce the need of assessing music performances, e.g., in institutions of musical education. Thus, any effort towards either formalizing human assessments or the creation of objective, reproducible, and unbiased systems for automatic assessment contributes to overcoming the above-mentioned challenges.

A system for automatic MPA can be used for software-based music tutoring applications to allow for easier accessibility of music education and individualized instruction. Past this, such objective assessment systems might also serve as tools for the evaluation of performance generation systems.

The approaches in automatic MPA follow the same general historical patterns as other audio analysis systems. Older systems extract hand-crafted features from recorded performances and then use a data-driven approach such as a regression model to map the features to a grade or assessment rating that reflects human ratings [5]. Deep learning methods have since been found to outperform feature extraction-based methods [6]; however, modern systems still fall short of the reliability required for a ready-to-use system [7].

Representation learning aims to accurately encode relevant and useful characteristics into a compressed representation. Representation learning methods such as VG-Gish have been shown to encode powerful audio features into a compressed representation which —when used as input to classification systems— can produce state of the art performance [8]. Contrastive learning is an emerging representation learning method which uses a distance-based loss between pairs of encoded training points in order to create meaningful class separation within the latent space of a neural network [9].

This study aims to investigate the use of contrastive learning to improve the performance of MPA systems. We investigate the use of contrastive-based learning methods in a regression task, where a deep neural network taking an input audio recording of a musical performance is tasked with estimating a numerical rating consistent with that of a professional judge. Our hypothesis is that learning a structured latent space will improve the ability of the regression components of MPA models in predicting an assessment rating. We investigate two methods of incorporating con-



trastive learning into a standard CNN-based architecture to learn a structured latent space, (i) a two-step training method introduced by Khosla et al. [10], and (ii) a joint loss method combining the contrastive loss term with a mean squared error loss term, a standard loss for training regression systems. As contrastive loss within a supervised context is generally designed for classification tasks [10] as opposed to regression tasks, we introduce a weighted contrastive loss suitable for regression tasks.

The remainder of this paper is structured as follows. First, we give an overview of music performance assessment and previous work on contrastive loss. Then, we introduce the proposed method in Sect. 3. Section 4 introduces our experimental setup. In the following Sect. 5, we evaluate the performance of the contrastive-based methods against a baseline architecture to predict a regression rating and perform analysis on the latent space of the baseline and contrastive-based methods to determine the efficacy of this clustering on the overall performance. Overall, we find that contrastive-based learning is able to better cluster the latent representation and produce performance gains within our MPA regression task.

2. RELATED WORK

MPA aims to understand and model the parameters of a musical performance and investigate their impact on a human listener [11]. MPA systems thus have the goal of assessing musical performances based on audio recordings without the input of expert judges. Early research on musical performances centered around analyzing symbolic data extracted from MIDI devices [12, 13], whereas recent research has increasingly focused on analyzing raw audio [11, 14]. In human performance assessment, music instructors must discern the individual subjective qualities and criteria and their importance. Similarly, automatic performance assessment systems extract features representing the audio file and then use a data-driven model to estimate the assessment rating. The features are either hand-crafted for the task [5, 15–18], or learned from the training data [6, 19, 20]. Systems with handcrafted features often use traditional machine learning approaches [21] while feature learning is usually done within a more complex neural model with low-level input representations such as spectrograms [22].

Some studies specifically aim to automatically produce a numerical rating on a predefined scale from audio representations of a musical performance, which involves implicitly learning the aspects of performances that correlate to certain rating criteria [6, 7, 20]. However, since numerical scores do not inherently include specific performance feedback, understanding the impacting factors can be challenging. The methods based on deep neural networks, while generally yielding superior performance, usually lack interpretability. Learning a structured latent space is a first step towards having a more easily understandable representation. Representation learning is an emerging method for performance assessment. For example, Huang et al. proposed a joint-embedding network which learns a shared latent space of a performance and its written score and derives a regression

rating by the cosine similarity between the two embeddings [7]. Representation learning methods thus potentially provide both performance improvements, as well as better interpretability of numerical scoring models, such as the ones in this study.

An emerging method of representation learning is the Contrastive Loss. Contrastive Loss aims to regularize the latent space so that the distances between latent vectors are meaningful. This is achieved by comparing the distances between the latent representations of pairs of training points and pushing them within a set distance in the latent space if they have similar labels, and outside this set distance if they are dissimilar. These distances are compared within the contrastive loss function of a model in order to encode the information within the latent vectors. This ideally creates class clusters within the latent space. Contrastive-based loss functions are often used to specifically learn structured latent representations of data [9, 10, 23], which then can be adapted for downstream tasks by training classifiers on these produced latent vectors [10, 23]. There has been considerable work done on the use of a supervised contrastive loss to cluster latent spaces for classification tasks. Chopra et al. introduce the max margin contrastive loss, and discuss its potential to discriminate classes when the exact number of classes may not be known, such as within recognition or verification tasks [9]. Khosla et al. investigate a supervised contrastive loss to train deep neural networks for classification tasks on the ImageNet dataset, and found that it outperforms general cross entropy based methods [10]. This implies that using the contrastive loss can produce an advantageous latent space layout more suitable for the following tasks. Ferraro et al. investigate the use of contrastive learning for music and audio for three downstream MIR tasks, genre classification, playlist continuation, and automatic tagging and found that contrastive-based learning outperforms the baseline within all three tasks and achieves comparable performance to SoTA [23]. Their findings suggest that contrastive learning is able to cluster similar musical recordings within the latent space of deep neural networks. To our knowledge, the use of contrastive learning has not been investigated within the context of MPA. Since it has been found to be advantageous within classification tasks across several modalities [9, 10, 23], we study the application of contrastive learning to MPA.

3. METHOD

We propose a weighted contrastive loss as a modification of the max margin contrastive loss introduced by Chopra et al. [9]. The loss function is adapted to be suitable for regression tasks. We investigate incorporating the contrastive loss via two different training scenarios for a convolutional neural network architecture.¹

¹ The code is available at: <https://github.com/pseshadri9/contrastive-music-performance-assessment>, last accessed 8/3/2021.

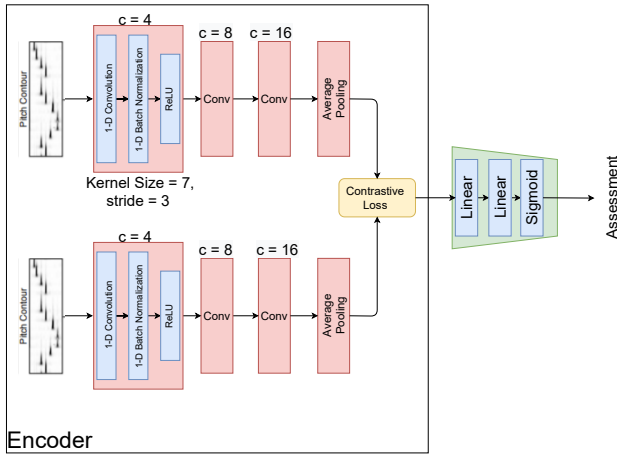


Figure 1: Contrastive-Based network architecture for regression.

3.1 Network architectures

3.1.1 Baseline

The baseline network used is the PCCnnNet architecture introduced by Pati et al. [6]. This architecture takes pitch contours as input and uses three convolutional layers followed by an average pooling layer in order to predict assessment ratings. Each convolutional layer contains a 1-D convolution, 1-D batch normalization [24], and ReLU non-linearity.

3.1.2 ContrastiveCNN

Based on the baseline system, we design the network visualized in Figure 1. Each branch of the network uses the same convolutional layers of the baseline with shared weights. Two linear layers are appended to this to predict the final rating with a sigmoid activation.

A two-step training procedure as detailed by Khosla et al. [10] is used to train this model. First, the encoder is trained using contrastive loss over the output latent vectors. After this, the encoder weights are frozen, and the linear layers are trained to regress this space using a mean squared error loss. Each datapoint within a training pair is fed through one encoder channel.

3.1.3 ContrastiveCNN-JL

The architecture of this network is equivalent to the ContrastiveCNN, but differs in training procedure. Rather than the two-step training procedure outlined above, the loss L is the addition of the contrastive loss L_C over the latent vectors and the mean squared error loss L_{MSE}

$$L = L_{MSE} + L_C. \quad (1)$$

3.1.4 Input representation

The input for each model is a pitch contour of each individual audition. Each pitch contour is an $N \times 1$ vector representing the fundamental frequency of each chunk of a performance of sequence length N chunks. Pitch Contour representations were extracted using the pYIN algorithm [25] at a sample rate of 44.1 kHz with a block size

	Middle School	Symphonic Band
Alto Sax	696	641
Clarinet	925	1156
Flute	989	1196

Table 1: Number of recordings per instrument.

and hop size of 1024 and 256 samples, respectively. The extracted fundamental frequencies are converted to MIDI pitch values and normalized to a range of $[0, 1]$ by dividing by 127, the maximum MIDI value.

3.2 Weighted Contrastive Loss

Contrastive loss is generally used for classification tasks in order to create distinct class boundaries within the latent space [9, 10, 23, 26]. The standard max margin contrastive loss [9] is defined as:

$$L_C = \frac{1}{2}YD^2 + \frac{1}{2}(1 - Y)\max(0, (m - D))^2, \quad (2)$$

where $Y = 1$ if the two datapoints in the pair have the same ground truth label, and $Y = 0$ if they do not. D is the Euclidean distance between the two latent vectors, and m is a set margin distance for which similarly labeled points should be clustered within. This results in points from the same class clustered together, while differently labeled points will be pushed past this pre-defined distance margin.

Since this loss is not suitable for regression tasks like ours, we propose a weighted contrastive loss term. For this new loss, we first split our continuous regression range $[0, 1]$ into C evenly spaced rating bins. For $C = 5$, for example, each rating bin has a range of 0.2, with exact multiples of 0.2 serving as the lower bounds for each bin X (i.e., $[0, 0.2)$, $[0.2, 0.4)$, ...). Each datapoint is assigned its respective bin according to its ground truth rating. These bins are then assigned the class indices $[0, 1, 2, \dots, C - 1]$, which will be used for our weighted contrastive loss. Second, we propose a variable margin to represent the ordered nature of the rating bins. For example, it is expected that the rating bin spanning $[0, 0.2)$ should have a greater distance from the bin covering $[0.8, 1]$ than from the $[0.2, 0.4)$ bin, as the bins themselves express a rating distance. The variable margin can therefore be defined as

$$m = |X_i - X_j| \cdot s, \quad (3)$$

where X_i and X_j represent the ground truth class indices of each datapoint within a pair (X_i, X_j) and s is the set margin distance. This variable margin scales the set distance proportionally to the expected distance between each rating bin. This variable margin then replaces the fixed margin m in Eq. (2).

4. EXPERIMENTS

Our experiments investigate primarily the performance of the proposed contrastive-based methods for MPA. In particular, we are interested in evaluating (i) the raw performance

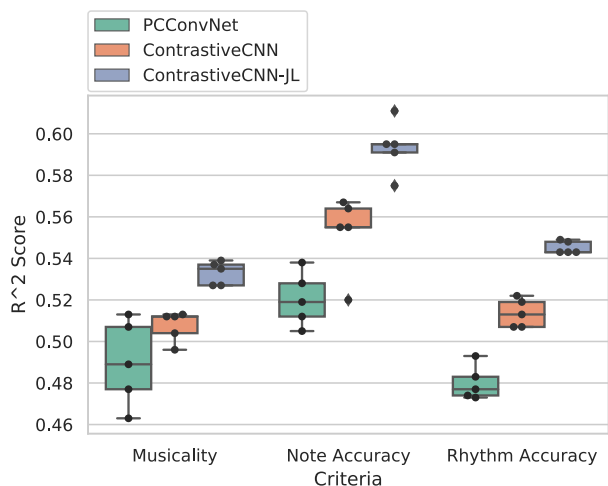


Figure 2: Results over the *Middle School* set.

in predicting ratings, (ii) the quality of the clustering of the latent spaces, and (iii) the effect of this clustering on the performance. We evaluate the learned representation both quantitatively and qualitatively.

4.1 Dataset

The used dataset comprises audio recordings and ratings of auditions from the Florida Bandmaster’s Association (FBA) from 2013 to 2018. This dataset contains raw audio recordings from three different levels of all-state auditions, *Middle School*, *Concert Band*, and *Symphonic Band*. Each student performs a prepared *lyrical exercise*, *technical exercise*, *scales*, and a *sight reading exercise*. This dataset includes several monophonic and percussive instruments. A subset of these data was used in this study, using the technical exercise from the alto saxophone, Bb clarinet, and flute recordings for the *Middle School* and *Symphonic Band* levels. Table 1 shows the number of recordings per instrument for both *Middle School* and *Symphonic Band*. The average duration of a *Middle School* and a *Symphonic Band* recording is approximately 30 s and 50 s, respectively.

Each singular recording represents the complete audition for one student and has assessment ratings by an expert judge for four assessment criteria defined by the FBA: *musicality*, *note accuracy*, *rhythm accuracy*, and *tone quality*. For consistency, we normalized each rating to the range [0, 1] by dividing the maximum rating, with 0 representing the worst possible score, and 1 representing the best possible score. Furthermore, the tone quality rating was ignored for this study as the audition is represented as pitch contours at our network input, a representation that does not carry sufficient information for modeling this criterion.

4.1.1 Pre-processing

Pitch contour representations were computed from raw audio recordings. Data augmentation via random chunking was used while training due to its ability to improve model performance [6]. Each pitch contour is chunked into sections of length 1000 (about 6 s) by randomly selecting the

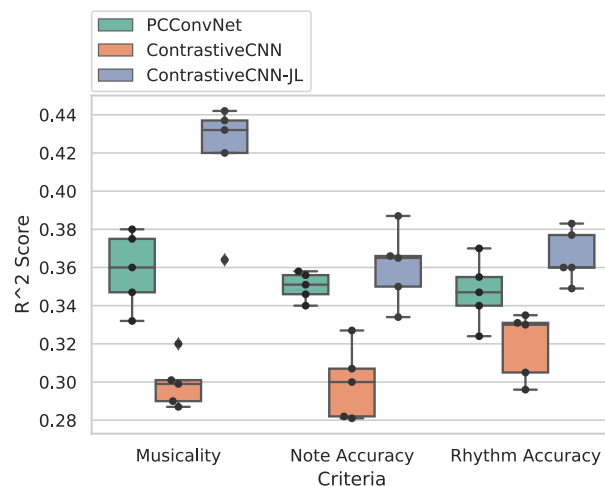


Figure 3: Regression results over the *Symphonic Band* set.

start position. This approach has been shown to improve model performance [6]. We assume the chunked segment would receive the same assessment rating as the entire audition recording.

4.2 Training procedures

Pairs for the contrastive loss were randomly sampled via generated random sequences each batch. Each datapoint within the pair was fed into a separate encoder channel of the model. Each model was trained using a stochastic gradient descent optimizer with a weight decay of $1e-5$ and momentum of 0.9. Early stopping was applied in each training sequence to stop if the validation loss had not decreased in 75 epochs. For training and evaluation, each dataset was split into training, testing, and validation sets in an 8:1:1 ratio. To measure the variance of the models, each model was trained five times using random seeds, as represented by the box plots. Within the two step training method following [10], the encoder channels were trained for 150 epochs at a learning rate of 0.1, while the linear layers were trained for 300 epochs at a learning rate of 0.005. The Joint Loss Network was trained for 300 epochs at a learning rate of 0.005.

4.3 Evaluation

We investigate the performance of the baseline and the contrastive-regularized networks amongst three different rating criteria, *musicality*, *note accuracy*, and *rhythm accuracy*. We predict the ratings for these criteria over both the *Middle School* and *Symphonic Band* dataset to determine performance over different levels of musical complexity, which can provide insights over the performance of MPA systems as musical complexity increases. The *Concert Band* dataset was omitted for consistency, as it was not evaluated in previous MPA studies that used this dataset [6, 7]. Each model (PCCConvNet, ContrastiveCNN, ContrastiveCNN-JL) was trained separately for each assessment criterion on both datasets. The unaltered PCCConvNet [6] served as the baseline model.

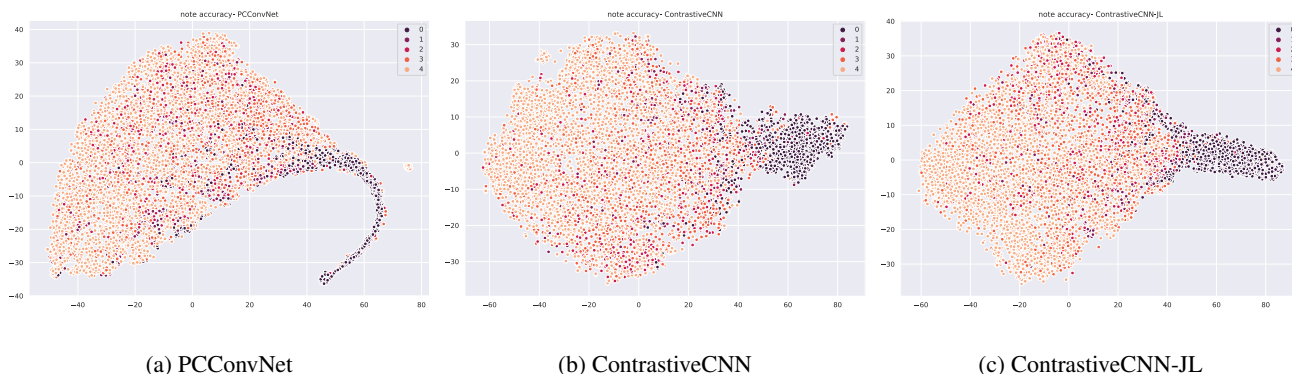


Figure 4: T-SNE visualization of the latent space of the three presented models.

4.3.1 Regression analysis

The coefficient of determination (R^2 score) over the output scores serves as the evaluation metric:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{4}$$

where y_i is the ground truth rating for a given datapoint, \hat{y}_i is the predicted rating, and \bar{y} is the mean ground truth rating over the entire set.

4.3.2 Latent space analysis

Using each trained model, the latent vectors of the testing set were obtained by only passing the input through each model’s encoder channels. A visualization was produced by applying T-SNE dimensionality reduction [27] to the latent vectors and plotting the output. Optimal parameters were found via a parameter search.

For a quantitative evaluation of the clustering quality, the latent space is evaluated by its Davies-Bouldin index [28]. The Davies-Bouldin index describes the average similarity of each cluster to its most similar cluster, which is defined as the ratio of within-cluster distances to between-cluster distances. The minimum index is zero and smaller values indicate better clustering [28].

5. RESULTS AND DISCUSSION

5.1 Regression results

Figure 2 and Figure 3 detail the results of the models on the *Middle School* and *Symphonic Band* datasets, respectively. Each box plot contains the five runs with random seeds 0-4 per each criteria and model. We can make the following observations:

- (i) All models perform better on the *Middle School* set than the *Symphonic Band* set (higher R^2 score). One possible explanation for this is that the *Symphonic Band* auditions tend to be longer and more complex with higher skilled players, potentially increasing the difficulty of extracting meaningful features representing the quality of the performance.
- (ii) All contrastive-based models outperform the baseline on the *Middle School* set; however, only the

ContrastiveCNN-JL meets and outperforms the baseline on *Symphonic Band*. This implies that the contrastive learning is more beneficial at a lower complexity of performance, but possibly has difficulty with data of higher complexity. One possible explanation could be that with a higher level of performance, and thus a higher complexity of information within each latent vector, the contrastive loss is unable to properly semantically relate the distances to the quality of the performance.

- (iii) The ContrastiveCNN-JL outperforms both the baseline and the ContrastiveCNN in every trial. This implies that the information gained by combining the traditional loss term with the contrastive loss helps learning a more meaningful latent space representation.

5.2 Latent space analysis

5.2.1 T-SNE plots

T-SNE visualizations of the latent space are presented for the baseline PCConvNet, ContrastiveCNN, and ContrastiveCNN-JL in Figure 4. As an example, we only present results for *Note Accuracy* on the *Middle School* dataset. The effect of the contrastive loss can be easily noticed, although the embedding spaces are not ordered perfectly in either case. The two models based on contrastive loss display a more defined distinction between low classes (0, 1) and higher classes (3, 4). Small same-class clusters can also be identified.

5.2.2 Class Distance Surface plots

Figure 5 shows the distances between the centroids of each class cluster within the latent space of the models trained on the *Middle School* dataset for *Note Accuracy*. While the contrastive-based models appear to have trouble properly ordering the middle range of ratings between classes 2 and 3, the distances appear to scale more smoothly than the distances within the baseline PCConvNet, indicating better ordering within the latent space.

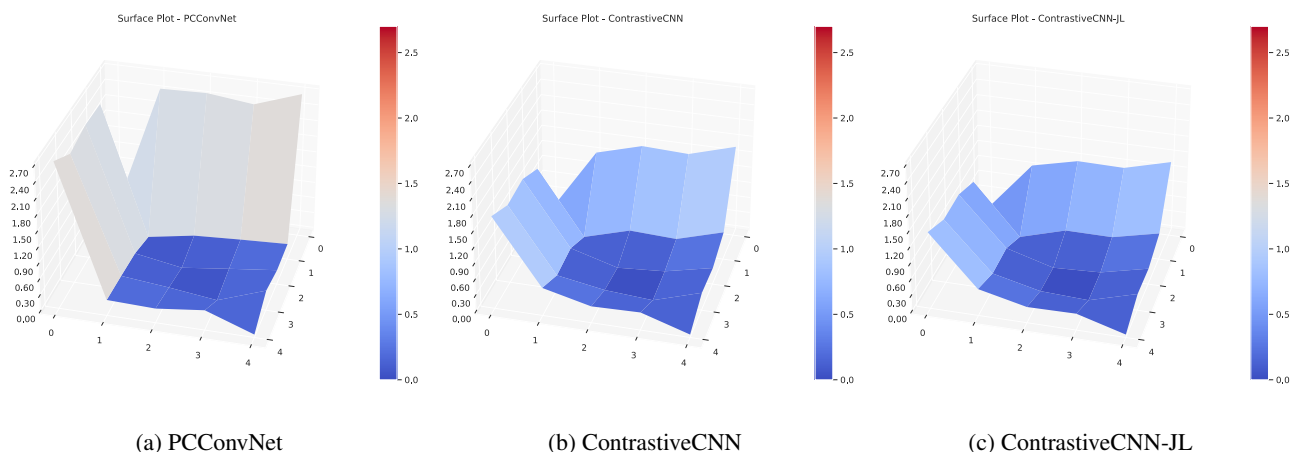


Figure 5: Class distances in the learned latent space of the three models.

5.2.3 Davies-Bouldin Index

Figure 6 shows the Davies-Bouldin indices of each model on the *Middle School* regression set. Each contrastive-based model has a considerably lower index than the baseline, which indicates that the latent space clustering is improved. Within this set, the lower the Davies-Bouldin index, the better the regression performance, implying that better clustered latent spaces do correlate with better regression.

6. CONCLUSION

This paper presented an approach to representation learning to improve the accuracy of a system for music performance assessment. We introduced a weighted contrastive loss suitable for regression tasks and showed how this latent space regularization improves results on a large real-world dataset for music performance assessment.

In future work, we plan to incorporate score information into the models, as this has been shown to improve performance [7]. More analysis should be done within contrastive learning methods to assess the effect of margin size, and number of classes on the performance of the model and the goodness of its clustering. Another approach to ensure that the learned representations contain relevant information is multi-task learning. It is worth investigating what related tasks might help increase the performance of music performance assessment. Moreover, supervised latent space regularization methods such as AR-VAE [29] and I-VAE [30] might be incorporated to force specific dimensions to specific performance characteristics.

7. ACKNOWLEDGMENTS

We would like to thank the Florida Bandmasters Association for providing the dataset used in this study.

We also gratefully acknowledge NVIDIA Corporation (Santa Clara, CA, United States) who supported this research via the NVIDIA GPU Grant program.

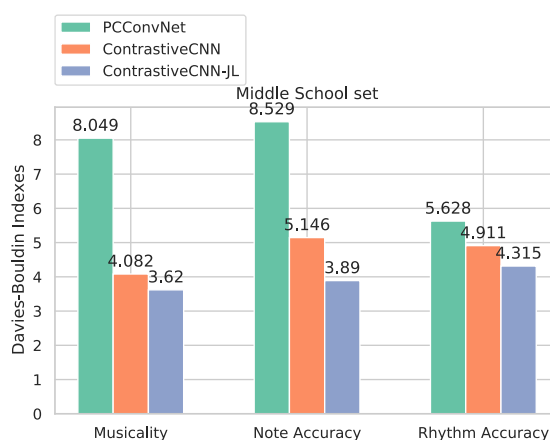


Figure 6: Davies-Bouldin indices of each model over the *Middle School* dataset. Smaller values indicate better clustering.

8. REFERENCES

- [1] E. F. Clarke, “Understanding the Psychology of Performance,” in *Musical Performance – A Guide to Understanding*, J. Rink, Ed. Cambridge: Cambridge University Press, 2002.
- [2] A. Lerch, C. Arthur, K. A. Pati, and S. Gururani, “An Interdisciplinary Review of Music Performance Analysis,” *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, pp. 221–245, 2020.
- [3] B. C. Wesolowski, S. A. Wind, and G. Engelhard, “Examining Rater Precision in Music Performance Assessment: An Analysis of Rating Scale Structure Using the Multifaceted Rasch Partial Credit Model,” *Music Perception: An Interdisciplinary Journal*, vol. 33, no. 5, pp. 662–678, Jun. 2016.
- [4] S. Thompson and A. Williamon, “Evaluating Evaluation: Musical Performance Assessment as a Research Tool,” *Music Perception: An Interdisciplinary Journal*, vol. 21, no. 1, pp. 21–41, Sep. 2003.

- [5] C.-W. Wu, S. Gururani, C. Laguna, A. Pati, A. Vidwans, and A. Lerch, "Towards the Objective Assessment of Music Performances," in *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*, San Francisco, 2016, pp. 99–103.
- [6] K. A. Pati, S. Gururani, and A. Lerch, "Assessment of Student Music Performances Using Deep Neural Networks," *Applied Sciences*, vol. 8, no. 4, p. 507, Mar. 2018.
- [7] J. Huang, Y.-N. Hung, K. A. Pati, S. Gururani, and A. Lerch, "Score-informed Networks for Music Performance Assessment," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*. Montreal: International Society for Music Information Retrieval (ISMIR), 2020.
- [8] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN Architectures for Large-Scale Audio Classification," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135, iSSN: 2379-190X.
- [9] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality Reduction by Learning an Invariant Mapping," in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Jun. 2006, pp. 1735–1742, iSSN: 1063-6919.
- [10] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised Contrastive Learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.
- [11] A. Lerch, *Software-Based Extraction of Objective Parameters from Music Performances*. München: GRIN Verlag, 2009.
- [12] C. Palmer, "Mapping Musical Thought to Musical Performance," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 15, no. 2, pp. 331–346, 1989.
- [13] B. H. Repp, "Patterns of note onset asynchronies in expressive piano performance," *Journal of the Acoustical Society of America (JASA)*, vol. 100, no. 6, pp. 3917–3932, 1996.
- [14] S. Dixon and W. Goebel, "Pinpointing the Beat: Tapping to Expressive Performances," in *Proceedings of the 7th International Conference on Music Perception and Cognition (ICMPC)*, Sydney, 2002.
- [15] T. Nakano, M. Goto, and Y. Hiraga, "An Automatic Singing Skill Evaluation Method for Unknown Melodies Using Pitch Interval Accuracy and Vibrato Features," in *Proceedings of the International Conference on Spoken Language Processing (INTER-SPEECH)*, vol. 12, Pittsburgh, PA, 2006, p. 1.
- [16] T. Knight, F. Upham, and I. Fujinaga, "The Potential for Automatic Assessment of Trumpet Tone Quality," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, FL, 2011, pp. 573–578.
- [17] C. Dittmar, E. Cano, J. Abeßer, and S. Grollmisch, "Music Information Retrieval Meets Music Education," *Multimodal Music Processing*, vol. 3, pp. 95–120, 2012.
- [18] S. Gururani, K. A. Pati, C.-W. Wu, and A. Lerch, "Analysis of Objective Descriptors for Music Performance Assessment," in *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*, Toronto, Ontario, Canada, 2018.
- [19] Y. Han and K. Lee, "Hierarchical Approach to Detect Common Mistakes of Beginner Flute Players," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 77–82.
- [20] C.-W. Wu and A. Lerch, "Learned Features for the Assessment of Percussive Music Performances," in *Proceedings of the International Conference on Semantic Computing (ICSC)*. Laguna Hills: IEEE, 2018.
- [21] A. Vidwans, S. Gururani, C.-W. Wu, V. Subramanian, R. V. Swaminathan, and A. Lerch, "Objective descriptors for the assessment of student music performances," in *Proceedings of the AES Conference on Semantic Audio*. Erlangen: Audio Engineering Society (AES), 2017.
- [22] C. Bhat, B. Vachhani, and S. K. Kopparapu, "Automatic assessment of dysarthria severity level using audio descriptors," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5070–5074.
- [23] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched Music Representations With Multiple Cross-Modal Contrastive Learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, 2021.
- [24] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 448–456, iSSN: 1938-7228.
- [25] M. Mauch and S. Dixon, "PYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 659–663, iSSN: 2379-190X.
- [26] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon, "A Survey on Contrastive Self-Supervised Learning," *Technologies*, vol. 9, no. 1, p. 2, Mar. 2021, number: 1 Publisher: Multidisciplinary Digital Publishing Institute.

- [27] L. van der Maaten and G. E. Hinton, “Visualizing High-Dimensional Data Using t-SNE,” *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
- [28] D. L. Davies and D. W. Bouldin, “A Cluster Separation Measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.
- [29] K. A. Pati and A. Lerch, “Attribute-based Regularization for Latent Spaces of Variational Auto-Encoders,” *Neural Computing and Applications*, 2020.
- [30] I. Khemakhem, D. Kingma, R. Monti, and A. Hyvarinen, “Variational Autoencoders and Nonlinear ICA: A Unifying Framework,” in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, Aug. 2020, pp. 2207–2217.

TRACING AFFORDANCE AND ITEM ADOPTION ON MUSIC STREAMING PLATFORMS

Dougal Shakespeare¹

Camille Roth^{1,2}

¹ Computational Social Science Team, Centre March Bloch, Berlin

² CNRS, France

dougal.shakespeare@cmb.hu-berlin.de, roth@cmb.hu-berlin.de

ABSTRACT

Popular music streaming platforms offer users a diverse network of content exploration through a triad of affordances: *organic*, *algorithmic* and *editorial* access modes. Whilst offering great potential for discovery, such platform developments also pose the modern user with daily adoption decisions on two fronts: platform affordance adoption and the adoption of recommendations therein. Following a carefully constrained set of Deezer users over a 2-year observation period, our work explores factors driving user behaviour in the broad sense, by differentiating users on the basis of their temporal daily usage, adoption of the main platform affordances and the ways in which they react to them, especially in terms of recommendation adoption. Diverging from a perspective common in studies on the effects of recommendation, we assume and confirm that users exhibit very diverse behaviours in using and adopting the platform affordances. The resulting complex and quite heterogeneous picture demonstrates that there is no blanket answer for adoption practices of both recommendation features and recommendations.

1. INTRODUCTION

The contemporary music streaming platform is a far cry from the digital repository which it once was. Increased diversification at a platform level has resulted in a range of affordances that is, modes of content access projected by the platform to the user, which allow one to explore a platform's ever expanding musical catalogs through novel paths. No longer is it necessary for the avid music listener to spend hours on end trawling through digital repositories to find their 'niche', thus performing a direct, interest-driven exploration of the whole catalog that is typically denoted as *organic* (*O*) use. Rather, they are free to draw upon further affordances commonly encapsulated within state-of-the-art platforms which provide some level of assistance or guidance: either purely based on *algorithmic*

(*A*) devices or on some level of *editorial* (*E*) curation. While at one level, it is true that affordance diversification yields increased potentials for exploration, this also comes at the price of greater emphasis being placed on user platform proficiency – questions of what affordance to employ and what items therein to adopt quickly become frequent decisions the modern user must deal with.

In our work we trace the aforementioned user tendencies through the notion of adoption on two fronts: *affordance adoption* (among the triad *O, A, E*) and *item adoptions* therein (i.e., item transfers across affordances). Through a comprehensive quantitative analysis of user listening practices on the popular music streaming platform, Deezer, our work sheds light on the varied and often heterogeneous nature of adoption and behavioural differences amongst users. The contribution of this work both sits within and helps to re-frame the growing body of literature which appraises the interconnected effects of human behaviour and algorithmic influence via an organic comparison.

2. LITERATURE REVIEW

2.1 Appraising Algorithmic Influence via an Organic Comparison

Whilst historically the primary role of a music Recommender System (mRS) on streaming platforms was to facilitate the efficient personalised exploration of a platform's often vast musical catalog thereby minimising the risk of *choice overload* [1], a substantial body of multi-disciplinary literature [2–5] points towards the conclusion that user exploration may still in practice, remain confined to a minute fraction of homogeneous musical content – a phenomenon famously denoted as the '*filter bubble*' [6]. Similarly in recent years a growing body of simulation based RS literature has also shed light on the tendency for feedback loops to emerge as a product of algorithmic recommendation and (simulated) human consumption [7–10]. Nonetheless, while the tendency for such practices to emerge is clearly outlined in literature, a less trivial second order question still remains ambiguous:

To what degree is user platform behaviour primarily a product of algorithmic influence or rather, an autonomous organic process imposed by the user themselves?



© Dougal Shakespeare, Camille Roth. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Dougal Shakespeare, Camille Roth, "Tracing Affordance and Item Adoption on Music Streaming Platforms", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

In light of such questions, a novel branch of the state of the art has sought to measure algorithmic influence by drawing parallels to a user’s organic platform behaviour. Roth conceptualises this debate through what he coins the ‘ROM-COM dichotomy’ [11] – the tendency for filtering algorithms to either Read Our Minds (ROM) acting as cognitive aids to facilitate organic exploration or rather, Change Our Minds (COM) algorithmically distorting a prior organic preference. Literature appraising algorithmic influence through an organic reference has been applied to a range of multi-disciplinary contexts. Bakshy et al. [12] study the role of Facebook’s NewsFeed algorithm in exposing users to cross-partisan content through an organic reference. Their work finds filtering algorithms to have minimal effect in reducing cross-partisan information in comparison to the organic selection processes suggesting a user’s diversity limitation may be principally due to human pre- and post-selection. In the music domain, Epps-Darling et al. [13] study the role of Spotify’s algorithmic influence on gender representation through an organic reference point. Their findings show user’s organic preference towards male artists to be marginally higher than that recommended by Spotify’s mRS, again suggesting the human organic bias to be stronger than that generated algorithmically. Anderson et al. [14] also analyse diversity with respect to organic vs programmed (algorithmic/editorial) listening events on Spotify. Through the generation of a usage-based embedding space, they find algorithmically-driven exploration to be less diverse than organic, providing evidence for a COM effect.

Furthermore, recent literature on music streaming uses paints a picture of divergent practices dependent upon the user’s degree of organic-algorithmic usage and actualised with respect to context and user mindsets [15, 16]. Thus, we commence our work with the prior hypothesis that user platform behaviour is largely varied - there is no *average user* for which a blanket answer to algorithmic influence may be applied.

2.2 Item Adoption in Recommender Systems

Whilst item adoption in terms of consumption confined to a given affordance has been covered extensively and critically both in terms of user studies [17, 18] and user modelling [19–21], literature concerning item adoption in relation to the dynamic transfer of items across affordances remains to date, sparse. Nonetheless, the adoption of music streaming platforms independent of their affordances has notably been studied in the field of Cultural Studies. In the works of Datta et al. [22], streaming adoption is shown to lead to substantial increases in both the quantity and diversity of music consumed by a user. Similarly, Rushan et al. [23] also study factors of the platform interface which in itself, determine a consumer’s decision to adopt music streaming platforms as a result of increased platform familiarisation. Still, adoption with respect to transitions across platform affordances remains a literature void which this

work seeks to fill.

2.3 Temporal Dimensions of mRS Usage

Time of day information has been evidenced to be an important signal in disentangling platform behaviour [24] and has thus in recent years, become a commonly utilised signal in context dependent RS literature [25–27] to increase personalisation and accuracy of recommendations. What is more, user studies have also revealed that both the time of day and week can play a substantial role in mediating user platform experience and downstream projected user behaviour [28, 29]. Utilising such rich temporal signals encapsulated within listening logs, our work seeks to explore the degree to which adoption on both fronts may differ across temporal daily usages.

3. METHODOLOGY

3.1 Listening Events Data Set

We work with about 2 years of listening histories from about 13K Deezer users who registered in the month of September 2017. We constrain our field of observation to users who remained active over the entire observation period. Formally, we impose a maximum inter-event time threshold of 10 days thus ensuring that users rely on the platform for a regular source of music. This eventually yields 2701 users which forms the ultimate user base for our analysis. We discard listening events <30s as these are deemed as so-called ‘skips’. We further merge unique song identifiers which share identical audio embeddings in a pre-build latent space supplied by Deezer (see [30] for use case / generation details). This prevents double counting of identical songs which may have been mis-labeled as distinct. For each listening event we characterise the affordance used to retrieve content of which on Deezer there exists a triad: *organic* (*O*), *algorithmic* (*A*) and *editorial* (*E*). Our research commences from a strict ternary classification of affordance taxonomies drawing influence from the existing body of literature of which this work closely relates [13, 14, 31]. We classify *A* affordances as those that refer to the platform’s plethora of Recommender System (RS) architectures (e.g., the popular Flow playlist on Deezer) whilst *E* affordances correspond to curated playlists (such as recommended playlists variously called “10s electronic”, “Rock & Chill”, etc.) of which the majority are mostly human constructed. All remaining modes of access are classified as *O* including for instance, the search bar, user-constructed playlists and more broadly, modes of content access which do not utilise any degree of recommendation.

3.2 Defining User Classes

Affordance Adoption. We capture user adoption of platform affordances through the proportion of content accessed via each of the platform’s three main affordances after aggregating listening histories. The listening history of a given user is represented by the temporally-ordered list

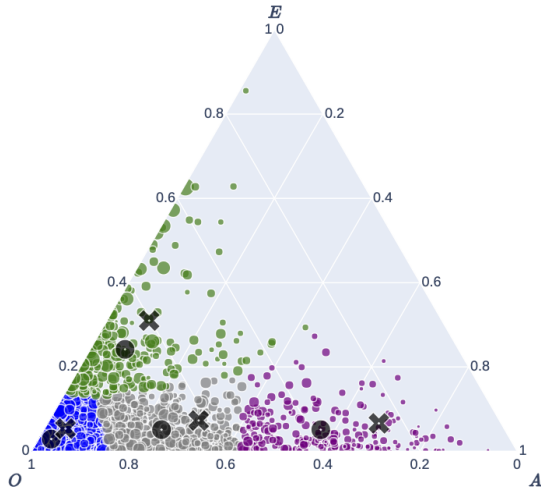


Figure 1. Ternary plot of affordance adoption classes $o+$ (blue), o (grey), α (green), a (purple), disks represent centroid positions. Crosses represent corrected centroid positions after taking into account the pre-adoption origin of plays (see Sec. 5).

of listening events over our observation period, formally defined as:

$$P = ((s_i, t_i, f_i))_{i \leq n}$$

where s_i is the song ID, t_i the timestamp, and $f_i \in \{A, E, O\}$ the affordance used for the i -th listening event. We accordingly denote the sublist of P restricted to a given affordance F as $P_F = ((s_i, t_i, f_i))_{i \leq n \wedge f_i = F}$. Considering the proportion of plays accessed *organically*, *algorithmically* and *editorially* respectively, the affordance profile of a given user is defined by the triplet $(|P_O|/|P|, |P_A|/|P|, |P_E|/|P|)$ which sums to 1 and can be represented as a barycentric coordinate in ternary space (see Figure 1). Performing a k-means clustering ($k = 4$) across affordance profiles yields 4 distinct classes of users which we label as follows: very organic ‘ $o+$ ’ (1786 / 65.98%), organic ‘ o ’ (429 / 15.85%), algorithmic ‘ a ’ (224 / 8.27%), organic/editorial ‘ α ’ (268 / 9.90%). We note that we rather deal with bins, areas or classes than with well-separated clusters per se. Thus, from herein we refer to affordance adoption clusters as classes. Already, a highly varied picture of affordance adoption on the platform emerges along with a shared preliminary benchmark: for all classes, users on average display some degree of O adoption whilst the same cannot be said for A and E . Indeed, at first sight it appears Deezer users do not typically adopt affordances on all fronts but rather, use the platform predominately as an organic catalog much the way one would search through a traditional song library, albeit a much larger one here. However, as we shall later detail, a user’s tendency to consume mostly O content may be misleading: if a significant proportion of a user’s O catalog is a product of A or E adoption the very definition of what it means to adopt O affordances is brought into question.

Exploration Behaviour. Beyond sheer user activity over the entire observation period denoted by play counts $|P|$,

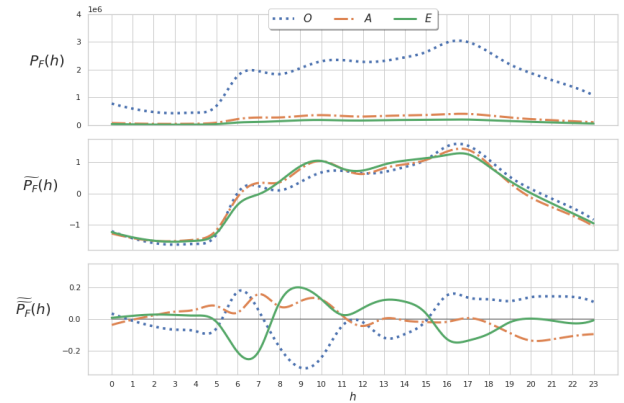


Figure 2. Normalised platform affordance time-of-day activity. Aggregate levels are shown (top) followed by z-normalised activity (middle) and residual de-trended activity levels (bottom).

we consider a notion of *redundancy* [29] quantified as a measure of how much a given user saturates their listening catalog (i.e., plays the same songs repeatedly), formally defined as $R = 1 - |S|/|P|$ where $S = \{s | (s, t, f) \in P\}$ is the set of unique songs in P .

We additionally characterise the diversity of a user’s exploration using a pre-built 32-dimensional latent space \mathcal{E} constructed from low-level audio features via metric learning [30]. The audio embeddings were primarily used by Deezer for the task of artist disambiguation (i.e., where artists had the same name but were stylistically unique). Thus, the construction of \mathcal{E} maximises distance for acoustically dissimilar artists while acoustically similar artists remain close in this space. For each user, we compute their average pairwise *cosine distances* between audio embeddings \mathcal{E}_s for each $s \in S$ and ultimately, report average values.

Temporal Time of Day Analysis. Platform usage can also be seen to vary significantly over the elapsed day at both an aggregate platform and user-centric level. Considering the former first, we compute and plot aggregate activity levels across all affordances at each hour of the day (Fig. 2, top row). Activity is found to largely consist of O followed by A and E at much lower magnitudes, consistent with what has been previously detailed. Without loss of generality, we denote \mathcal{P} as the platform-level history for all users (i.e., the whole dataset). To analyse temporal trends independent of magnitude, we consider hourly play counts for each affordance:

$$\mathcal{P}_F(h) = \left| ((s_i, t_i, f_i))_{i \leq n \wedge (t_i \text{'s hour} = h) \wedge (f_i = F)} \right|$$

to which we subsequently apply a z-normalisation relative to daily play count averages defined as:

$$\widetilde{\mathcal{P}}_F(h) = \frac{\mathcal{P}_F(h) - \langle \mathcal{P}_F(h) \rangle_h}{\sigma_{\mathcal{P}_F(h)}}$$

These normalised activity levels (Fig. 2, second row) reveal three peaks of gradually increasing magnitude, respectively in the early morning, morning, and afternoon

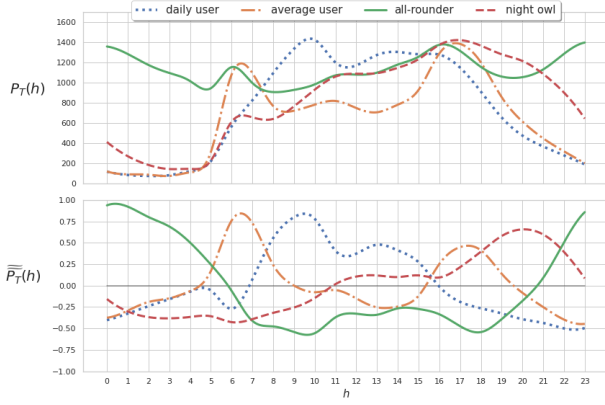


Figure 3. Normalised time-of-day activity levels for each time-of-day class. Aggregate levels are shown (top) followed by residual de-trended activity levels (bottom).

(16-17:00), from which a gradual decay in activity is experienced. To capture temporal adoption variations of affordances with respect to one another, we finally apply so-called ‘*detrending*’ as per [32] by comparing the above z-normalisations $P_F(h)$ relative to other affordances, formally defined as:

$$\widetilde{\widetilde{P}}_F(h) = \widetilde{P}_F(h) - \langle \widetilde{P}_F(h) \rangle_F$$

Affordance adoption at a platform level clearly varies across hours of the day (Fig. 2, bottom row). In both the early morning and between the afternoon hours and evening we observe a tendency to favour O respective to A and E affordance life cycles in the same time blocks (i.e. independent of magnitude). Otherwise, either A or E seem to be favoured (again, in relative trend) and essentially appear to exhibit bi-modal relative adoption peaks across the day, albeit at different moments (rather in the morning for A and in the early morning and early afternoon for E).

We complement the temporal platform-level analysis by examining daily patterns at the user level, as users will commonly have varied activity levels on music streaming platforms [33] which may be missed by a standalone platform level analysis. To this end, we consider user-level hourly activity aggregated over all affordances $P(h) = \left| \left((s_i, t_i, f_i) \right)_{i \leq n \wedge (t_i \text{'s hour} = h)} \right|$. As in [29], we subsequently cluster users using k-means ($k = 4$ again) applied on $P(h)$ as a 24-dimensional unit vector. We observe 4 distinct user-centric behavioural dynamics which we label as follows: the *average user* (726 / 26.82%), the *night owl* (928 / 34.28%), the *all rounder* (245 / 09.05%) and the *daily user* (808 / 29.85%). We represent the variations of $P(h)$ on Fig. 3 by applying the same type of normalization as used above for platform-level quantities $\mathcal{P}(h)$, except that we consider temporal clusters T instead of affordances F i.e., $\mathcal{P}_T(h)$ in place of $\mathcal{P}_F(h)$.

3.3 Item Adoption

In this work we focus exclusively on item adoptions which are imposed directly by the user themselves and thus, we

only characterise the tendency for users to adopt A or E songs into their O catalog. To capture this behaviour, we introduce the novel measure of *adoption* α . At a high level, an item can be said to be adopted the first time it has been played organically by a user given that the song was first recommended through some affordance F and not played organically as a prior. Since we are interested in item adoption and thus unique songs, we now focus on song sets rather than lists of plays i.e., S . We outline two possible mutually exclusive song sets, denoted ϕ and ρ , to differentiate songs which could have been adopted yet were not, from those which were actually adopted:

– **Adoption feasible**, ϕ , denoting the set of songs which were played through F but not via O and thus, had the potential to be adopted yet were not:

$$\phi_F = \left\{ s \in S \mid \exists (s, t, F) \in P, \nexists (s, t', O) \in P \right\}$$

– **Adoption realised**, ρ , denoting the set of songs which were played via F as a prior and were consumed through O at least once subsequently:

$$\rho_F = \left\{ s \in S \mid \exists (s, t, F) \in P, t < \min_{(s, t', O) \in P} t' \right\}$$

Furthermore, if a user is exposed to more recommendations before ultimately making the decision to adopt, this may indicate a weaker influence of the platform’s affordance or that the act of adoption is less likely to be a direct product of it (for instance the user might have heard the song from an external source such as the radio). To capture this intuition, we introduce $r_F(s)$, the number of recommendations of song s which appeared through F before organic adoption:

$$r_F(s) = \left| \left\{ (s, t, F) \in P \mid t < \min_{(s, t', O) \in P} t' \right\} \right|$$

to which we apply a polynomial scaling function which decays to give more weight to lower numbers of recommendations - a similar practice to how listening counts are often scaled logarithmically in mRS literature [34, 35].

We assess the relative impact of item adoption at two levels of abstraction. Foremost, with respect to the number of items which both could have been and were adopted by the user through F i.e., $|\phi_F| + |\rho_F|$. Formally let this be defined by:

$$\alpha_F = \frac{\sum_{s \in \rho} r_F(s)^{-\lambda}}{|\phi_F| + |\rho_F|}$$

where $\lambda \in (0, 1]$ is a hyperparameter which affects the degree of polynomial decay with respect to algorithmic impact. In our experiments we set the value of $\lambda = 0.5$. We note our choice of λ is cautious and should in future work be more refined with statistical and qualitative user studies exploring the role of repeated affordance recommendation prior to adoption.

Secondly, we normalise adoption with respect to the number of unique items consumed via O , thereby capturing the relative impact of algorithmic adoption in a user’s

	o+					o				
	All rounder	Average user	Night owl	Daily user	All	All rounder	Average user	Night owl	Daily user	All
$ P $	*26.62K	15.59K	*20.87K	15.94K	18.58K	26.09K	15.63K	19.68K	16.88K	18.01K
R	*0.90	0.85	0.86	*0.83	0.85	*0.82	0.77	0.77	0.77	0.77
α_A	0.28	0.25	0.28	0.25	0.25	0.15	0.14	0.15	0.14	0.14
α_E	0.26	0.27	0.24	*0.21	0.25	0.18	0.19	0.16	0.15	0.17
α'_A	0.01	0.02	0.02	0.02	0.01	0.06	0.06	0.06	0.05	0.06
α'_E	0.02	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.03	0.02
\mathcal{E} dist.	0.28	0.29	0.29	0.29	0.28	0.28	0.29	0.30	0.29	0.29

	a					æ				
	All rounder	Average user	Night owl	Daily user	All	All rounder	Average user	Night owl	Daily user	All
$ P $	41.20K	15.06K	16.18K	19.75K	19.01K	23.80K	15.40K	19.68K	23.09K	20.14K
R	*0.85	0.77	0.74	0.77	0.77	0.77	0.74	0.77	0.76	0.76
α_A	0.10	0.09	0.09	0.09	0.09	0.17	0.16	0.20	0.15	0.17
α_E	0.20	0.14	0.15	0.13	0.14	0.14	0.14	0.16	0.13	0.14
α'_A	0.08	0.08	0.08	0.08	0.08	0.03	0.02	0.02	0.02	0.02
α'_E	0.03	0.03	0.02	0.03	0.03	0.05	0.05	0.05	0.05	0.05
\mathcal{E} dist.	0.29	0.30	0.29	0.30	0.30	0.28	0.29	0.28	0.29	0.29

Table 1. Experimental results across two static affordance and temporal *time of day* user classes. Values in bold represent the top value, while marked with * are results where the difference is statistically significant (two tailed t-test, $\alpha = 0.05/n$ after Bonferroni correction).

overall organic listening catalog. Formally,

$$\alpha'_F = \rho_F / |\{s \in S \mid \exists(s, t, O) \in P\}|$$

We note that this value is bounded by the number of organic streams in a user’s listening history but nonetheless, we deem this to be a useful measure to capture the influence of item adoption in bringing into question the very meaning of what is deemed organic.

4. RESULTS

Temporal Affordance Adoption Variations. We first examine the distribution of affordance classes across time-of-day classes. As shown in Figure 4 we observe two fundamental preliminary findings: (1) daily users are more heavily composed of both *a* and α users respective to other time-of-day classes; (2) *o+* users are more proportionally likely to reside within the all rounder and, to a lesser extent, night owl class. Framed differently, users who adopt almost solely *O* are more likely to favour platform activity in the evening hours of the day whilst users who more heavily *A/E*-adopt are more likely to favour activity in the day time hours. Once again, our findings reiterate what was observed from our temporal platform evaluation – the use of recommendation affordances corresponds to different categories of temporal use as well as, we contend, different types of users.

Characterising platform behaviour. To further disentangle the respective use cases we now characterise behavioural dynamics for each time-of-day and affordance class. Focusing first on affordances, we attain results that go against the grain of a diversity-constraining narrative (see Table 1). Users who *A*-adopt more frequently are found to have more diverse exploration in \mathcal{E} whilst maintaining relatively low redundancy levels as measured by *R*.

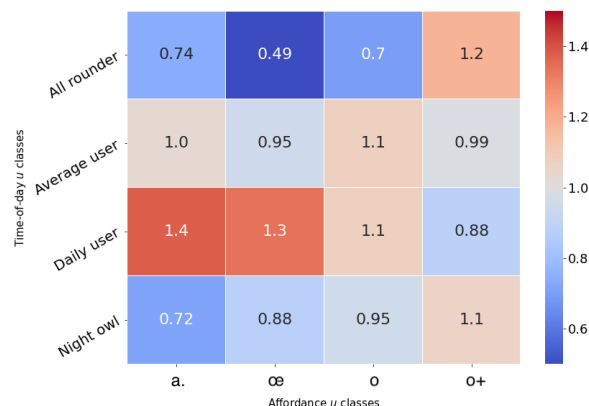


Figure 4. Affordance vs. time-of-day distributions. Values are normalised such that above or below 1 indicate respectively similar, over- or under- representation of affordance classes respective to those found globally.

It appears the consumption of *A* content in fact diversifies a user’s *P* at both a behavioural and deeper content-based level whilst on the contrary, *o+* users are found to saturate their listening catalog reflected in the high *R* levels attained.

Regarding item adoption within affordance classes we observe both *a* and α users to have low levels of α_A and α_E respectively. This can be interpreted as a passivity to recommendations - such users are more likely to use *A* and *E* affordances regularly but on a so-called *auto-pilot* akin to radio consumption. Nonetheless, when *a* and α users do take the decision to adopt this makes a substantial impact to their *O* catalog and thus, the dispersion of users in the *A, E, O* ternary space as we shall later detail. Drawing parallels to a more pure organic behaviour through the *o+*, we observe polar opposite dynamics in

comparison to the a and α users. Whilst $o+$ users A/E -adopt sparsely, their ultimate downstream platform use is less recommendation-skeptic reflected in the much higher levels of item adoption rates for A and E affordances attained ($\alpha_A = 0.25$, $\alpha_E = 0.25$) but with minimal impact to the constitution of their overall O catalog. From the detailed findings it is clear that our preliminary assumption that users display varied behavioural dynamics holds true: users diverge dramatically regarding their affordance adoption, adoption of items therein and perhaps most importantly, subsequent downstream impact experienced to their overall O catalog.

We next study the effect of temporal preference upon behavioural practices. For a given affordance class and behavioural measure (e.g. $o+$, R) we perform two-tailed Welch’s unequal-variance paired t-tests [36] over all 6 pairs of temporal classes which form the affordance class i.e., (*all rounder, average user*), (*all rounder, daily user*), (*all rounder, night owl*), (*average user, daily user*), (*average user, night owl*), (*daily user, night owl*). For a given temporal class, Table 1 marks a value as significant if $p < \alpha/6$ (i.e., after adjusting for errors via Bonferroni correction [37]) is attained for all 3 pairwise t-tests performed containing the temporal class under consideration (e.g. (*all rounder, average user*), (*all rounder, daily user*), (*all rounder, night owl*)). The results show that users vary significantly with respect to activity and R levels. Central to this finding is the *all rounder* who displays significantly higher activity levels for $o+$ users but at the price of saturating their listening catalogue reflected in the significantly higher R levels across the majority of affordance classes. Such findings suggest that for a given affordance class, significant deviations across temporal classes only occur for low level behavioural features. There is no clear downstream propagation to a user’s deeper musical preferences, reflected by diverse musical content in an audio embedding space and preference for item adoption – a theory which we shall now test empirically.

Disentangling heterogeneous platform behaviour. To disentangle the influence of time-of-day preference and affordance adoption on user item adoption and reactions to recommendations we next perform a factorial ANOVA, shifting each behavioural attribute to be the dependent variable whose variance we seek to explain. We primarily fit our data to an OLS model $Y = \beta_0 + \beta_1F + \beta_2T + \beta_3FT + \varepsilon$ (where F and T represent affordance and time-of-day labels respectively) before subsequently applying a factorial ANOVA. Due to space constraints, the full ANOVA results table is not included however we now detail the results most relevant to this work. As hypothesised, we observe the only effect size (η_p^2) for which temporal classes may have both a moderate and significant effect is with regard to a user’s activity $|P|$. On the contrary, affordance classes offer a moderate-to-high explanatory factor for the variance of the remaining behavioural attributes, foremostly adoption. Perhaps most interestingly, the effect of affordance classes on α_A is particularly strong (0.11) implying that a user’s decision to adopt items into their or-

ganic catalog may, as previously hypothesised, be principally a product of adopting recommendation affordances.

For completeness we ultimately examine the effect of sequential time-of-day and affordance adoption influence on the notion of what is meant by an organic stream. Contrary to our preliminary belief that organic access acted as a benchmark for Deezer platform exploration, we observe users to actually be more algorithmic and editorial than first thought, albeit indirectly. Considering a stream to belong to the affordance in which it was adopted as opposed to organic we recompute centroids for each affordance class. In cases where a stream was both adopted via A and E we deem that the item was adopted via the affordance which had the most streams prior to adoption. As shown in Fig. 1, we observe all affordance adoption classes centroids to experience a marked shift towards A and E poles – even more so for users who are already closer to these poles i.e., particularly for a and α users.

5. CONCLUDING REMARKS

In a time where the modern music streaming platform encapsulates a myriad of modes of accessing content, users may and do personalise their platform use in highly varied ways. By acknowledging, assuming and confirming the diversity of user platform behaviour, our work traces the interconnected yet surprisingly sequential factors which drive affordance and item adoption. Our results paint a highly complex picture of user platform behaviour whereby time-of-day preference mediates low-level platform behaviour (activity levels) while affordance adoption preference mediates the ultimate higher-level decision to adopt content into one’s O catalog, a factor which is indeed more reflective of musical taste.

Coming full circle, the heterogeneity of item adoption and its subsequent impact brings into question the nature of what constitutes an O stream - after taking into consideration the role of adoption, users are indeed found to be markedly less organic than was initially thought. This, in turn, may redefine what affordance adoption really is and perhaps most importantly, challenge the emerging literature which seeks to appraise A influence via an O comparison. Our work hints at the non-binary nature of O which should be carefully considered and analysed. Although beyond the scope of this work, we suggest a fruitful future direction could be to explore non-human item adoptions (i.e., item transfers from $O \rightarrow A|E$). We also advice for future work to explore the role of temporal preference upon adoption practices at varied degrees of abstraction be it weekly, monthly or longitudinal. On the whole, this work aims to hint at a direction that currently remains relatively unexplored in outstanding scholarship concerning the impact of RS on the diversity of user consumption: that user behaviour determines how recommendation affordances are being adopted. In practice, this type of work and approach could be utilised at the platform level to further the development of context-dependent RS, providing musical recommendations which are far more suited to the high variety of user’s driving use cases.

6. REFERENCES

- [1] D. Bollen, B. P. Knijnenburg, M. C. Willemsen, and M. Graus, “Understanding choice overload in recommender systems,” in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys ’10. New York, NY, USA: Association for Computing Machinery, 2010, p. 63–70. [Online]. Available: <https://doi.org/10.1145/1864708.1864724>
- [2] M. Airoidi, “The techno-social reproduction of taste boundaries on digital platforms: The case of music on youtube,” *Poetics*, p. 101563, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0304422X21000462>
- [3] N. Seaver, “Captivating algorithms: Recommender systems as traps,” *Journal of Material Culture*, vol. 24, no. 4, pp. 421–436, 2019.
- [4] T. Bonini and A. Gandini, ““first week is editorial, second week is algorithmic”: Platform gatekeepers and the platformization of music curation,” *Social Media + Society*, vol. 5, no. 4, pp. 1–11, 2019.
- [5] T. T. Nguyen, P.-M. Hui, F. M. Harper, L. Terveen, and J. A. Konstan, “Exploring the filter bubble: The effect of using recommender systems on content diversity,” in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 677–686. [Online]. Available: <https://doi.org/10.1145/2566486.2568012>
- [6] E. Pariser, *The Filter Bubble: What the Internet Is Hiding from You*. The Penguin Press, 2011.
- [7] A. Ferraro, D. Jannach, and X. Serra, “Exploring longitudinal effects of session-based recommendations,” *Fourteenth ACM Conference on Recommender Systems*, Sep 2020. [Online]. Available: <http://dx.doi.org/10.1145/3383313.3412213>
- [8] A. Ferraro, X. Serra, and C. Bauer, “Break the loop: Gender imbalance in music recommenders,” in *Proceedings of the 2021 Conference on Human Information Interaction and Retrieval*, ser. CHIIR ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 249–254. [Online]. Available: <https://doi.org/10.1145/3406522.3446033>
- [9] D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac, “What recommenders recommend: an analysis of recommendation biases and possible countermeasures,” *User Modeling and User-Adapted Interaction*, vol. 25, pp. 427–491, 2015.
- [10] J. Zhang, G. Adomavicius, A. Gupta, and W. Ketter, “Consumption and performance: Understanding longitudinal dynamics of recommender systems via an agent-based simulation framework,” *Info. Sys. Research*, vol. 31, no. 1, p. 76–101, Mar. 2020. [Online]. Available: <https://doi.org/10.1287/isre.2019.0876>
- [11] C. Roth, “Algorithmic distortion of informational landscapes,” *Intellectica*, vol. 70, no. 1, pp. 97–118, 2019.
- [12] E. Bakshy, S. Messing, and L. A. Adamic, “Exposure to ideologically diverse news and opinion on facebook,” *Science*, vol. 348, no. 6239, pp. 1130–1132, 2015. [Online]. Available: <https://science.sciencemag.org/content/348/6239/1130>
- [13] A. C. Epps-Darling, H. Cramer, and R. T. Bouyer, “Artist gender representation in music streaming,” in *Proceedings of the 21st International Society for Music Information Retrieval Conference*. Montreal, Canada: ISMIR, Oct. 2020, pp. 248–254. [Online]. Available: <https://doi.org/10.5281/zenodo.4245416>
- [14] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, “Algorithmic effects on the diversity of consumption on spotify,” in *Proceedings of The Web Conference 2020*, ser. WWW ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2155–2165. [Online]. Available: <https://doi.org/10.1145/3366423.3380281>
- [15] C. Hosey, L. Vujović, B. St. Thomas, J. Garcia-Gathright, and J. Thom, *Just Give Me What I Want: How People Use and Evaluate Music Search*. New York, NY, USA: Association for Computing Machinery, 2019, p. 1–12. [Online]. Available: <https://doi.org/10.1145/3290605.3300529>
- [16] N. Karakayali, B. Kostem, and I. Galip, “Recommendation systems as technologies of the self: Algorithmic control and the formation of music taste,” *Theory, Culture & Society*, vol. 35, no. 2, pp. 3–24, 2018.
- [17] M. D. Ekstrand, M. Tian, I. M. Azpiazu, J. D. Ekstrand, O. Anuyah, D. McNeill, and M. S. Pera, “All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness,” in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, ser. Proceedings of Machine Learning Research, S. A. Friedler and C. Wilson, Eds., vol. 81. New York, NY, USA: PMLR, 23–24 Feb 2018, pp. 172–186. [Online]. Available: <http://proceedings.mlr.press/v81/ekstrand18b.html>
- [18] Y. Jin, N. Tintarev, and K. Verbert, “Effects of individual traits on diversity-aware music recommender user interfaces,” in *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, ser. UMAP ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 291–299. [Online]. Available: <https://doi.org/10.1145/3209219.3209225>
- [19] M. Pichl and E. Zangerle, “User models for multi-context-aware music recommendation,” *Multimedia Tools and Applications*, pp. 1–23, 10 2020.
- [20] G. Aridor, D. Goncalves, and S. Sikdar, “Deconstructing the filter bubble: User decision-making and

- recommender systems,” in *Fourteenth ACM Conference on Recommender Systems*, ser. RecSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 82–91. [Online]. Available: <https://doi.org/10.1145/3383313.3412246>
- [21] A. B. Melchiorre and M. Schedl, “Personality correlates of music audio preferences for modelling music listeners,” in *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 313–317. [Online]. Available: <https://doi.org/10.1145/3340631.3394874>
- [22] H. Datta, G. Knox, and B. J. Bronnenberg, “Changing their tune: How consumers’ adoption of online streaming affects music consumption and discovery,” *Marketing Science*, vol. 37, no. 1, pp. 5–21, 2018.
- [23] R. I. Rushan, “The way you listen to music: effect of swiping direction and album arts on adoption of music streaming application,” *Behaviour & Information Technology*, pp. 1–22, 2020.
- [24] P. Herrera, Z. Resa, and M. Sordo, “Rocking around the clock eight days a week: an exploration of temporal patterns of music listening,” in *1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010, Barcelona, Spain*, Barcelona, 26/09/2010 2010. [Online]. Available: files/publications/womrad2010_submission_16.pdf
- [25] L. Baltrunas and X. Amatriain, “Towards time-dependant recommendation based on implicit feedback,” in *In Workshop on context-aware recommender systems (CARS&A'Z09)*, 2009.
- [26] S. Volokhin and E. Agichtein, “Understanding music listening intents during daily activities with implications for contextual music recommendation,” in *Proceedings of the 2018 Conference on Human Information Interaction Retrieval*, ser. CHIIR '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 313–316. [Online]. Available: <https://doi.org/10.1145/3176349.3176885>
- [27] D. Sánchez-Moreno, Y. Zheng, and M. N. Moreno-García, “Incorporating time dynamics and implicit feedback into music recommender systems,” in *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, 2018, pp. 580–585.
- [28] A. Krause, A. North, and L. Hewitt, “The role of location in everyday experiences of music.” *Psychology of Popular Media Culture*, vol. 5, pp. 232–257, 07 2016.
- [29] T. Louail and M. Barthelemy, “Headphones on the wire,” *ArXiv*, vol. abs/1704.05815, 2017.
- [30] J. Royo-Letelier, R. Hennequin, V. Tran, and M. Mousallam, “Disambiguating music artists at scale with audio metric learning,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018, pp. 622–629.
- [31] Q. Villermet, J. Poiroux, M. Moussallam, T. Louail, and C. Roth, “Follow the guides: disentangling human and algorithmic curation in online music consumption,” in *Fifteenth ACM Conference on Recommender Systems*, ser. RecSys '21. New York, NY, USA: Association for Computing Machinery, 2021.
- [32] J. L. Toole, M. Ulm, M. C. González, and D. Bauer, “Inferring land use from mobile phone activity,” in *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, ser. UrbComp '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 1–8. [Online]. Available: <https://doi.org/10.1145/2346496.2346498>
- [33] R. Mehrotra, C. Shah, and B. Carterette, “Investigating listeners’ responses to divergent recommendations,” in *Fourteenth ACM Conference on Recommender Systems*, ser. RecSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 692–696. [Online]. Available: <https://doi.org/10.1145/3383313.3418482>
- [34] S. Dean, S. Rich, and B. Recht, “Recommendations and user agency,” *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, Jan 2020. [Online]. Available: <http://dx.doi.org/10.1145/3351095.3372866>
- [35] G. Jawaheer, M. Szomszor, and P. Kostkova, “Comparison of implicit and explicit feedback from an online music recommendation service,” in *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, ser. HetRec '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 47–51. [Online]. Available: <https://doi.org/10.1145/1869446.1869453>
- [36] G. Ruxton, “The unequal variance t-test is an underused alternative to student’s t-test and the mann-whitney u test,” *Behavioral Ecology*, vol. 17, 04 2006.
- [37] C. Bonferroni, *Teoria statistica delle classi e calcolo delle probabilità*, ser. Pubblicazioni del R. Istituto superiore di scienze economiche e commerciali di Firenze. Libreria internazionale Seeber, 1936. [Online]. Available: <https://books.google.de/books?id=3CY-HQAACAAJ>

COMPUTATIONAL ANALYSIS AND MODELING OF EXPRESSIVE TIMING IN CHOPIN MAZURKAS

Zhengshan Shi

Center for Computer Research in Music and Acoustics, Stanford University

kittyshi@ccrma.stanford.edu

ABSTRACT

Performers' distortion of notated rhythms in a musical score is a significant factor in the production of convincingly expressive music interpretations. Sometimes exaggerated, and sometimes subtle, these distortions are driven by a variety of factors, including *schematic* features (both structural such as phrase boundaries and surface events such as recurrent rhythmic patterns), as well as relatively rare *veridical* events that characterize the individuality and uniqueness of a particular piece. Performers tend to adopt similar pervasive approaches to interpreting schemas, resulting in common performance practices, while often formulating less common approaches to the interpretation of veridical events. Furthermore, some performers choose anomalous interpretations of schemas. We present a machine learning model of expressive performance of Chopin Mazurkas and a critical analysis of the output based upon statistical analyses of the musical scores and of recorded performances. We compare the timings of recorded human performances of selected Mazurkas by Frédéric Chopin with performances of the same works generated by a neural network trained with recorded human performances of the entire corpus. This paper demonstrates that while machine learning succeeds, to some degree, in expressive interpretation of schemata, convincingly capturing performance characteristics remains very much a work in progress.

1. INTRODUCTION

Performers of classical music typically interpret a score's symbolic music notation as a basis of performance. This interpretive transformation from symbols to musical sound demands decisions regarding inherently imprecise or vague symbols such as dynamic and tempo markings. Furthermore, performers often divert from strict interpretations of precise symbols such as quantized rhythms in order to provide a sense of musical shape and direction. Expressive timing is a particularly important aspect of performance, with temporal deviations of tempi and distorted rhythms [1] to indicate structural demarcations, express implied affective [2] and articulate stylistic conventions.

These interpretive performance decisions are often made with little conscious thought reflecting internalized notions of traditional performance practices and schemas.

The complexity and multidimensionality complicit in the creation of an expressive musical performance has made the task a rich domain for theoretical analysis and computational modeling. Prior studies include analysis-derived rule-based methods such as the KTH model [3], as well as machine learning approaches dating back to Widmer's inference of note-level performance principles based on Sonatas by Mozart [4]. Statistically derived rules include historically rooted schematic tendencies such as the note inégales, arching tempo curves, and cadential ritard were encapsulated in the KTH model. Some generalized schematic rules, such as the tendency to perform a note staccato if the note is repeated immediately, were observed both in KTH and in Widmer's machine learning model.

More recent novel data-driven approaches including both linear [5] and nonlinear [6, 7] methods have been developed to model expressive performance by extracting basis functions (i.e. features) of each note. These features include note, metrical position, dynamic, and tempo markings. Recent efforts apply hierarchical attention networks [8] and conditional variational RNNs [9] to generate expressive piano music performances.

Our goal here is to examine computational models of expressive timing. As noted, performers rarely play metronomically but rather introduce more or less subtle nuances to vary performed durations. For example, most performers tend to slow their tempo in response to major structural breaks [10]. Repp [1] studies patterns of expressive timing over 115 performances of a same piece and suggests independent timing strategies that can describe each pianist's timing pattern. Chew [11] reveals extreme pulse elasticity as musical *tipping points*. Peperkamp et al. [12] propose ways to formally represent relative local tempo variations in a vector space.

We aim to understand how a neural-network-based system generalizes performance practices and compositional style given multiple performances of each of the Mazurkas in our corpus. We train a neural network to predict the tempo curve of each Mazurka. We then analyze expressive timing by comparing human performed Mazurkas to computer generated performances. We observe that while machine learning generalizes key schematic performance practices, it is less successful in capturing veridical performance characteristics.



2. SCHEMATA AND STATISTICAL PERFORMANCE ANALYSIS

Schemata are prototypical melodic, harmonic, or rhythmic/metric characteristics that constitute defining attributes of a particular style or genre [13]. Some schemata, such as the harmonic progression at the approaches to cadential phrase endings, have evolved as pervasive attributes of a common musical language. Chopin’s Mazurkas share schemata. They are all composed in triple meter, with regular phrase lengths typically comprised of short motivic units of one or two measures. Along with signature stylistic attributes they also share particular performance practice traditions. In this section we focus on the evolution of schematic features in performance.

2.1 Data

CHARM’s Mazurka Project¹ comprises a collection of approximately 3000 individual recorded performances of Mazurkas composed by Frédéric Chopin. Kosta et al. [14] augmented the recordings from the project with automatically aligned score-beat positions, loudness values, as well as positions of expressive markings. The resulting dataset, named as “MazurkaBL”, contains 44 Mazurkas with 2000 performances. Sapp [15] has manually annotated 5 Mazurkas with around 300 performances. For each performance, beat times were recorded. These annotations, as well as “MazurkaBL” were used as data for the study.

2.2 Statistical Analysis of Rhythmic Schemata in the Mazurkas

Particular rhythmic patterns characterize the Mazurka, as evident in the frequency of pattern occurrences across the corpus. The ten most recurrent rhythmic patterns are summarized and illustrated in Figure 1.

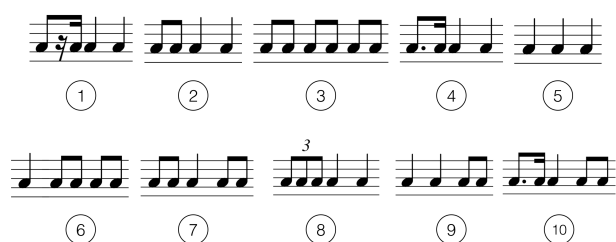



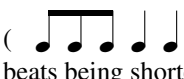
Figure 1: The ten most frequently occurring rhythmic patterns in Chopin’s Mazurkas in order of prominence.

We define the *Mazurka Quality* as a beat that is accentuated by temporal elongation repeatedly across most instances of a given Mazurka. Mazurka performances are often characterized as having a “stretched” second or third beat of the measure, at the durational expense of the downbeat [16]. We observed this *short-long* pattern in some mazurkas. However, we also observed that the elongation of the Mazurka Quality was not always compensated

Pattern #	% of 1st beat shortened
1	45.30%
2	67.74%
3	37.11%
4	50.29%
5	56.28%
6	22.00%
7	58.08%
8	82.08%
9	39.23%
10	33.36%

Table 1: Percentage of the first beat shortened over all recorded performances of our corpus for the 10 most recurring Mazurka rhythmic patterns. Note that in most performances, the first beat is shortened in rhythmic patterns #2 and #8, indicating a schemata interpretive performance practice.

for by shortening the downbeat. To validate this, we observed how each pianist executes a pattern on each measure, comparing the duration of the downbeat to the other beats in that measure. Alas, across all performances in the dataset, we observed that only 47.93% of the downbeats were shortened (as compared with the second beat).

This suggests that the *short-long* pattern appears only in specific rhythmic patterns. We then examined the tempo curve where human pianists played the above 10 rhythmic patterns respectively. Table 1 summarizes a comparison of how the duration of the first beat of each rhythm is altered in the recorded performances compared with the second beat. These 10 rhythmic patterns comprise over 70% of all rhythmic patterns in 44 Mazurkas in MazurkaBL. We see that in column 2 of Table 1, for rhythmic pattern #2 () and rhythmic pattern #8 (), there is a great percentage of downbeats being shortened (67.74% and 82.08%).

2.3 High Correlation Sections

We examined musical phrases where performers have the highest agreement. We calculated the Pearson’s correlation coefficient (PCC), a statistic that measures linear correlation between two variables (or two sets of numbers). This method was previously used by Sapp [15] to represent similarities between performers of Mazurkas. Pearson’s correlation coefficient is defined as:

$$\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_i - \bar{y})^2}} \quad (1)$$

The equation means to divide the covariance of the two variables by the product of their standard deviations. This results in a number ranging from -1 to 1 where -1 indicates negative correlation, 0 indicates no linear correlation, and 1 indicates positive correlation.

¹ <http://www.mazurka.org.uk>



Figure 2: Phrases with the highest correlation in tempo (in order) in Op.68 No.3 performed by 53 pianists.

We looked at Op.68 No.3. As in Figure 2, we found that phrase ends have the highest correlation. Figure 2(a) is the end of C section, 2(b) is the end of the A section, 2(c) is the very end, 2(d) is the end of phrase 1, and 2(e) is the end of phrase 1 after the C section. The Pearson’s correlation coefficient of phrase (a) through (e) are: 0.84, 0.84, 0.79, 0.76 and 0.72. When we plotted the tempo of how 53 different pianists playing the highest correlation phrase in Figure 3, we found that all have similar trends: the tempo of the phrase reached climax at the beginning, and then gradually slowed down towards the end. A similar situation was found in Op.24 No.2. The tempo curves of the phrase reached highest at the beginning, then dropped tremendously at the second half of the phrase.

3. VERIDICAL EVENTS IN MAZURKA PERFORMANCES

We borrow Bharucha’s [17] distinction between *schematic* events and *veridical* events. A *veridical* event is a musical occurrence characterized by something unexpected within the context of the work. This salient characteristic—whether rhythmic, melodic, harmonic, textural, articulatory or a combination thereof—is typically relatively unique and rare in the specific work, and often is noticeable and attention grabbing. As opposed to a schema, veridical

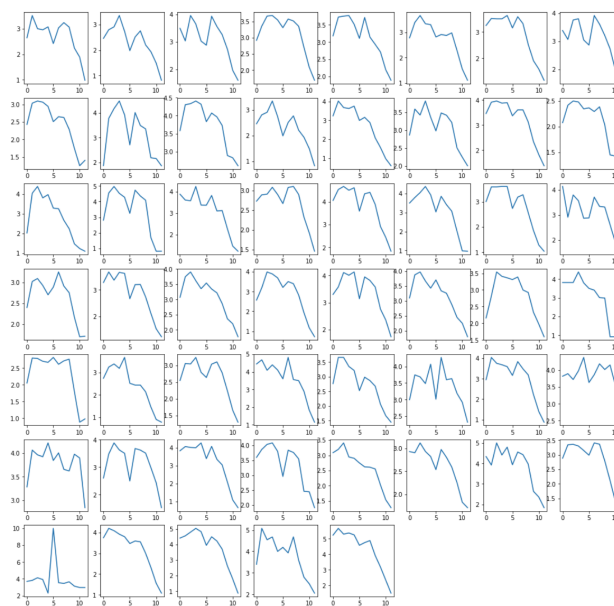
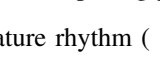


Figure 3: Tempo curves of 53 pianists playing the phrase with the highest correlation in Op.68 No.3. The Y axis represents beat-per-second.

events are less likely to have broadly shared prevalent performance practices.

3.1 Unexpected Change in Harmony

Unexpected changes in harmony often cause veridical events. In the opening passage of Op.24 No.1 (Figure 4), the signature rhythm () appears for 6 times (beats 1–3, 7–9, 13–15, 19–21, 31–33, and 37–39). They are rhythmic pattern #1. According to the schema in section 2, they should be performed as a lengthening of the first beat and a shortening of the second or third beat. But performers did not all follow this schema. For the repeating motif in beats 1–3, 13–15, as well as 37–39, most pianists performed the first beat long and the second beat short. However, in beats 7–9, 19–21, 31–33, most pianists changed their Mazurka Quality to lengthen the third beat and shorten the first beat. This is due to a change of harmony in these beats. For example, in beat 9 and beat 33 the piece goes to a vii-th chord that makes the F# in the top voice lead to the G on the next measure. In beat 21, there is an accidental of C# that leads to D. These leading actions cause the elongation of the third beat, rather than the downbeat.

Another example is in Op.63 No.3. Figure 5 and Figure 6 show two phrases that appear at the end of the A section, and at the end of the A’ section of Op.63 No.3. According to the trends we summarized in section 2, the phrases that are located at the end of a section are usually the highest correlated phrases among all pianists, as the tempo curves are usually gradually going down. However, these two phrases are the least correlated phrases among different pianists in the whole piece. When we performed a harmonic analysis, we found that there is a secondary dominant chord in the middle of both phrases. The secondary

Figure 4: The first theme of Op.24 No.1. Veridical events happen on highlighted notes where there are unexpected changes in harmony. Yellow rectangles mark a lengthening of the beat and red rectangles mark the shortening of the beat.

Figure 5: The low correlation phrase in Op.63 No.3. There is a secondary dominant chord in measure 7.

dominant chords change the harmonic color. In addition, the last chord of each phrase is a common tone diminished chord. Thus the phrase becomes a veridical event.

3.2 Rubato

In Op.24 No.2, the least correlated phrase among 68 pianists is the one with a “rubato” mark on it, as in Figure 7. This phrase appears after the B phrase, serving as a transitional phrase. When Chopin marks rubato, it typically departs from the Mazurka schema and is thus a veridical event, as there is no strict rule about how to perform this excerpt. Pianists usually play the phrase with their own interpretations.

4. COMPUTATIONAL MODEL OF EXPRESSIVE PERFORMANCE

Can machine play music as expressively as humans do? If so, to what extent? This section describes a computational model to synthesize expressive piano music performances. The motivation is mainly to model the complex

Figure 6: The low correlation phrase 2 in Op.63 No.3. There is a secondary dominant chord in measure 55.

Figure 7: The least correlated phrase in Op.24 No.2.

relationship between score properties and tempo in the performance. The goal is to understand how a neural-network-based system generalizes performance practices and compositional style given multiple performances of each of the Mazurkas in the corpus.

4.1 Input Features

We used MusicXML encoding of the Mazurka scores. Most computational systems of expressive performances take a sequence of note features extracted from MusicXML as input. However, for practical reasons MusicXML format does not readily identify simultaneities. For example, a chord is represented as a sequence of note tokens. Since our goal is to study expressive timing in Mazurkas, especially on how beats are grouped and emphasized, it is important to capture such metrical relationships in the encoding. As a result, we used beat-based features (i.e., features for each beat, rather than for each note).

We first extracted note information on a MusicXML file using *partitura* [18]. For each metrical position, we extracted the following features: highest and lowest notes within the beat, number of simultaneous notes within the beat, the rhythmic pattern of the beat (i.e., triplet, two eighth notes, one quarter note, etc.), articulation (accent and staccato) markings in the beat, metrical phase (i.e., first, second, or third beat in the measure), indicator of the start beat of a phrase, indicator of the final beat of a phrase. The maximum and minimum pitch are represented numerically between 0 and 1, while the rest of the features are represented by one-hot vectors.

4.2 Output Features

Piano allows for expressive variation in timing, dynamics, and articulation [1]. The output features are velocity, tempo, and articulation.

The velocity is a numeric value between 0 and 1, corresponding to the same values in the dataset. The beat tempo is first calculated as the reciprocal of the beat interval such that

$$\text{tempo}_i = \frac{\text{IBI}_i^{\text{score}}}{\text{IBI}_i^{\text{perf}}} = \frac{1}{\text{onset}_{i+1} - \text{onset}_i} \quad (2)$$

, where IBI_i represents the inter-beat-interval for the i -th beat. The unit of the tempo is beat-per-second.

Then, we translated the absolute value of the tempo into relative tempo ratio, such that

$$\text{tempo}'_i = \frac{\text{tempo}_i - \overline{\text{tempo}}}{\overline{\text{tempo}}} \quad (3)$$

For example, -0.2 means 20% slower than the average tempo, 0 means the same tempo as the average tempo, and

1 means doubling the average tempo. We limited this value to within -1 to 1.

During generation time, given the features of each beat, our system predicts the velocity and the relative tempo ratio of the beat. We then scaled the velocity to 1–127 and we calculated the onset time for each beat as

$$\text{onset}_i = \text{onset}_{i-1} + \frac{1}{\text{tempo}'_i \times T + T} \quad (4)$$

, where T is a constant that the user specifies the mean tempo to be. The unit of T is beat-per-second.

We encoded the generated performance in MIDI files capturing onset time, offset time, and velocity of each note. Due to the limitation of the dataset representation, the offset time of each note is unknown. Thus the prediction of the articulation (duration of the note in the performance over duration of the note in the score) is replaced with a fixed length. This does not affect our research about expressive timing since tempo is affected only by the onsets.

4.3 Training

We used 3-layer bi-directional LSTMs with 128 units to model beat-wise parameters. For velocity, the final layer is a sigmoid activation. For tempo prediction, the final layer is a tanh activation. The models were trained in a supervised fashion to minimize the mean-square-error loss. The sequence length was 64, the dropout rate was 0.5, and the learning rate was 0.001. We split the data into 80% training data and 20% validation data. The validation loss was 0.0412 for velocity and 0.0837 for tempo.

5. WHAT DOES THE NEURAL NETWORK LEARN?

5.1 Schemata

In section 2, we demonstrated that the characteristic signature rhythms are associated with the Mazurka Quality. To validate that the neural network can learn the Mazurka Quality on different signature rhythms, we encoded 16 measures of each signature rhythm as testing cases (the input scores) to feed into the beat-based neural network.

We then plotted the absolute tempo curve for each 16-bar signature rhythm input score. As in Figure 8, different input scores output different tempo curves. We see that in signature rhythms #2, #7, and #8, the output of the neural networks shows the “short-long” pattern, i.e., the tempo value of the first beat of each measure is higher than that of the second beat. Such pattern is especially strong on signature rhythm #8—there is on average a 26.6 beat-per-minute tempo difference between the first beat and the second beat. While in other rhythm as input, we see lengthening (slower tempo) of the first beat. This result aligns with Table 1, showing that the model learns about this general schema about Mazurkas.

5.2 Tempo Correlation

To evaluate the correlation of model-generated performance and human average performance, we calculated the

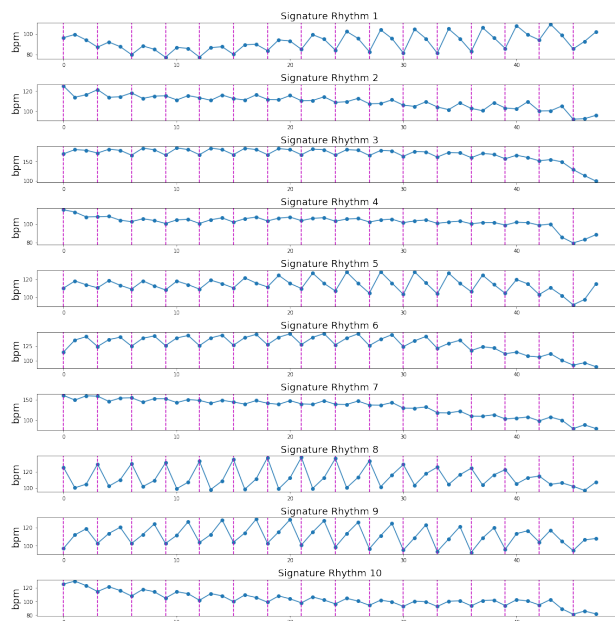


Figure 8: Tempo curves of different signature rhythm inputs. The “short-long” pattern is evident rhythms #2, #7, and #8.

Pearson’s correlation coefficient (PCC) between computer-generated tempo and average human tempo for five pieces in the test set. From Table 2, we see that PCC-BH (PCC between the tempo generated by beat-based model and human average tempo) is higher than PCC-VH (PCC between the tempo generated by VirtuosoNet [8] and human average tempo), indicating that the beat-based model learned generalized schematic performance practices more successfully than VirtuosoNet in tempo estimation. As a reference, PCC between one human performance and other human performances is between 0.29 and 0.97.

Mazurka Op. #	PCC-VH	PCC-BH	PCC-HH
Op.17 No.4	0.065	0.151	0.794
Op.24 No.2	N/A	0.497	0.778
Op.30 No.2	0.048	0.44	0.786
Op.63 No.3	0.167	0.59	0.714
Op.68 No.3	0.489	0.59	0.889

Table 2: Pearson’s correlation coefficient of performance tempo generated by VirtuosoNet and human average tempo (PCC-VH), the beat-based model and human average tempo (PCC-BH), and a random human performance tempo and average human performance tempo (PCC-HH).

5.3 Veridical Events

When plotting the tempo curve of Op.63 No.3 (Figure 9), we see that the beat-based model (line 3) learns about the schema that when pattern #2 occurs, the downbeat gets shortened. This aligned mostly with human performances. For performances generated by VirtuosoNet (line 4), since it is trained on 16 composers’ pieces, it is understandable that it does not favor Mazurka’s rhythmic tempo. Thus we

see an almost opposite direction.

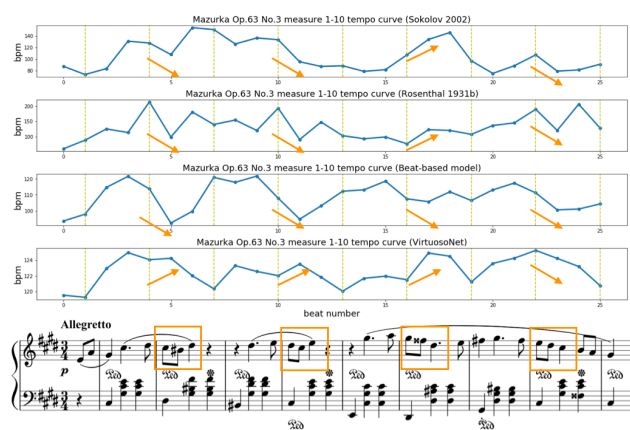


Figure 9: Tempo curves of the first 10 measures of Op.63 No.3 from two human performances (lines 1 & 2), a beat-based model (line 3) and VirtuosoNet (line 4), with the score (line 5). Tempo trends of pattern #2 (boxed) are noted with arrows.

In an analysis of 95 distinct human performances of the same work, in beats 5 and 6 (the first rectangle in Figure 9), 89 performers shortened the downbeat, in beats 11 and 12 (the second rectangle), 82 performers shortened the downbeat, and in beats 23 and 24 (the fourth rectangle), 88 performers shortened the downbeat. However, in beats 17 and 18 (the third rectangle), there were more performers (53 out of 95) who lengthened the downbeat rather than shortening it. While this seems to be an “outlier”, we were interested to further investigate what’s happening musically on these two beats.

Harmonic analysis was performed on the first ten measures of the score as in Figure 10. Note that there is a secondary dominant in measure 7 (beats 17-19), whose veridical change of color and direction prompts performers to emphasize the moment. The beat-based network captured the schematic “short-long” accent across many Mazurka performances, however, performance of this salient veridical event was not compellingly captured.

Another example is the A phrase of Op.24 No.2 (as in Figure 11). This phrase consists of four sub-phrases. Each sub-phrase is two-bar long. When the machine played this passage, we saw a very consistent trend. As in Figure 12, for each sub-phrase of 2 bars (6 beats), the machine lengthened the second to last beat. In addition, for all four sub-phrases, the tempo curves were similar: during the first three beats the tempo surged, and for the next two beats the



Figure 10: Harmonic analysis of the first 10 measures of Op.63 No.3. The secondary dominant in measure 7 is a veridical characteristic not captured by the beat-based network.



Figure 11: Phrase A of Op.24 No.2 . This phrase consists of four sub-phrases, each two-bar long.

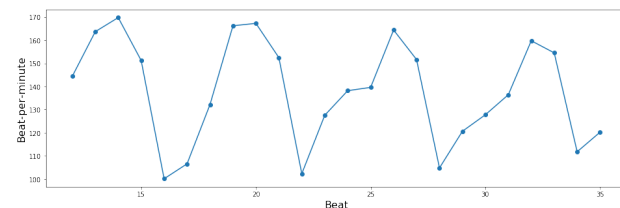


Figure 12: The tempo curve of machine playing the A Phrase of Op.24 No.2. For each sub-phrase, the machine emphasized (lengthened) the second to last note. This trend is consistent among all four sub-phrases.

tempo decreased, and finally the tempo slightly increased for beat 6. However, when we examined human performances, we found different results. As in Table 3, in contrast to the computer performance, human performers tend to vary the emphasized beat for each sub-phrase, whereas the computer performs the same one for each sub-phrase.

Sub-phrase	B1	B2	B3	B4	B5	B6
1	24	11	2	3	8	2
2	0	0	0	11	27	12
3	7	7	0	26	7	3
4	4	2	4	25	10	5

Table 3: Beats that human performers lengthened most for each sub-phrase in Op.24 No.2.

6. SUMMARY

In this paper we described our implementation of a beat-based model to learn expressive timing parameters in Chopin Mazurkas. Comparing human performances with performances generated by our model, we note that neural network succeeds at modeling schemas (such as distortion of the characteristic “short-long” Mazurka rhythm, and temporal augmentation at the approach to phrase-ends). However piece-specific veridical events (such as performed variations of repeated rhythmic units) are difficult to learn. One reason for this is that insufficient instances of examples of such veridical moments in the training set make it difficult for a deep learning-based system to acquire. Capturing the performance nuances of veridical events is a critical next step for the success of future computational models of expressive music performance.

7. REFERENCES

- [1] B. Repp, “A microcosm of musical expression. i. quantitative analysis of pianists’ timing in the initial measures of chopin’s etude in e major,” *The Journal of the Acoustical Society of America*, vol. 104, no. 2, pp. 1085–1100, 1998.
- [2] C. Palmer, “Music performance,” *Annual review of psychology*, vol. 48, no. 1, pp. 115–138, 1997.
- [3] A. Friberg, R. Bresin, and J. Sundberg, “Overview of the kth rule system for musical performance,” *Advances in Cognitive Psychology*, vol. 2, no. 2-3, pp. 145–161, 2006.
- [4] G. Widmer, “Machine discoveries: A few simple, robust local expression principles,” *Journal of New Music Research*, vol. 31, no. 1, pp. 37–50, 2002.
- [5] M. Grachten and G. Widmer, “Linear basis models for prediction and analysis of musical expression,” *Journal of New Music Research*, vol. 41, no. 4, pp. 311–322, 2012.
- [6] C. E. Cancino-Chacón, T. Gadermaier, G. Widmer, and M. Grachten, “An evaluation of linear and non-linear models of expressive dynamics in classical piano and symphonic music,” *Machine Learning*, vol. 106, no. 6, pp. 887–909, 2017.
- [7] C. E. C. Chacón, “Computational modeling of expressive music performance with linear and non-linear basis function models,” Ph.D. dissertation, Johannes Kepler University Linz, Austria, 2018.
- [8] D. Jeong, T. Kwon, Y. Kim, K. Lee, and J. Nam, “Virtuosonet: A hierarchical rnn-based system for modeling expressive piano performance,” in *Proceedings of the 20th International Society for Music Information Retrieval*, Delft, The Netherlands, 2019, pp. 908–915.
- [9] A. Maezawa, K. Yamamoto, and T. Fujishima, “Rendering music performance with interpretation variations using conditional variational rnn,” in *Proceedings of the 20th International Society for Music Information Retrieval*, Delft, The Netherlands, 2019, pp. 855–861.
- [10] N. Todd, “A model of expressive timing in tonal music,” *Music Perception: An Interdisciplinary Journal*, vol. 3, no. 1, pp. 33–57, 1985.
- [11] E. Chew, “Playing with the edge: Tipping points and the role of tonality,” *Music Perception: An Interdisciplinary Journal*, vol. 33, no. 3, pp. 344–366, 2016.
- [12] J. Peperkamp, K. Hildebrandt, and C. C. S. Liem, “A formalization of relative local tempo variations in collections of performances,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 158–164.
- [13] R. Gjerdingen, *Music in the Galant Style*. Oxford University Press, 2007.
- [14] K. Kosta, O. Bandtlow, and E. Chew, “MazurkaBL: score-aligned loudness, beat, expressive markings data for 2000 chopin mazurka recordings,” in *Proceedings of the fourth International Conference on Technologies for Music Notation and Representation (TENOR)(Montreal, QC)*, 2018, pp. 85–94.
- [15] C. S. Sapp, “Comparative analysis of multiple musical performances,” in *ISMIR*, 2007, pp. 497–500.
- [16] A. Swartz, “The polish folk mazurka,” *Studia musicologica Academiae Scientiarum Hungaricae*, vol. 17, no. Fasc. 1/4, pp. 249–255, 1975.
- [17] J. J. Bharucha, “Tonality and expectation,” *Musical perceptions*, pp. 213–239, 1994.
- [18] M. Grachten, C. E. Chacón, and T. Gadermaier, “partitura: A python package for handling symbolic musical data,” in *Late-Breaking Demo Session of the 20th International Society for Music Information Retrieval Conference. Delft, The Netherlands*, 2019.

COMPUTATIONAL ANALYSIS OF MELODIC MODE SWITCHING IN RAGA PERFORMANCE

Nithya Shikarpur¹ Asawari Keskar² Preeti Rao¹

¹Department of Electrical Engineering, ²Department of Physics

Indian Institute of Technology Bombay, India

{nithyas, prao}@ee.iitb.ac.in

ABSTRACT

Melodic mode shifting is a construct used occasionally by skilled artists in a raga performance to enhance it by bringing in temporarily shades of a different raga. In this work, we study a specific North Indian Khyal concert structure known as the Jasrangi jugalbandi where a male and female singer co-perform different ragas in an interactive fashion. The mode-shifted ragas with their relatively displaced assumed tonics comprise the identical set of scale intervals and therefore can be easily confused when performed together. With an annotated dataset based on available concerts by well-known artists, we present an analysis of the performance in terms of the raga characteristics as they are manifested through the interactive engagement. We analyse both the aspects of modal music forms, viz. the pitch distribution, representing tonal hierarchy, and the melodic phrases, across the sequence of singing turns by the two artists with reference to representative individual performances of the corresponding ragas.

1. INTRODUCTION

Ragas are melodic modes that underpin all performances of Indian art music across both the North Indian (Hindustani) and South Indian (Carnatic) traditions. There are dozens of ragas in common practice, distinguished by their salient melodic properties which include the choice of tonal material, the hierarchy of notes (*svara*), their intonation and typical phrasal contexts and, finally, the association with a particular mood. A drone sounds the tonic throughout the concert making the relative intervals of all the notes clearly apparent to the listener providing, thus, strong cues to the raga identity in terms of the corresponding tonal material and hierarchy, and the melodic phrases. Automatic raga identification from computed pitch class histograms, normalised by the concert tonic, have worked well, especially when ragas with different tonal material appear in the dataset [1, 2]. Ragas with the identical scale notes relative to the tonic such as allied ragas have

also been successfully differentiated with pitch distributions that exploit the differences in intonation and emphasis across the common set of *svara* [3]. A particular raga performance construct that has not received much attention computationally is melodic mode switching. As in other modal music, it is possible to develop a large number of scales by means of the modal shift by using any note of a raga as the tonic and building a new set of scales by maintaining the intervals or ratios between the notes and the tonic [4]. Termed '*murchana*', this is a subtle form of modulation (similar to key modulation from a major to its relative minor) and is used by skilled musicians to temporarily bring in shades of another raga during a performance.

In this work, we specifically consider a recent but widely acclaimed development in the North Indian Khyal classical music scenario of mode-shifted ragas performed together in concert by a pair of singers. Motivated by a desire to create a space where a male and a female singer can perform together in spite of the $\frac{1}{2}$ octave difference typical of their vocal pitch ranges, the two ragas are chosen such that the M (the lower octave fourth) of the higher-pitched voice serves as the S (tonic) for the lower-pitched voice with all actual note values (i.e. in terms of fundamental frequency or MIDI number) being identical in the two [5]. Figure 1 shows the scales of the pentatonic raga pairs considered in this paper. With the concert drone typically tuned to the Sa of the female voice, the singers strive to maintain the character of their respective ragas in the '*jugalbandi*' performance that interleaves the two singers' voices in equally weighted roles, as in what may be considered a 'call and response' musical format. What makes it particularly interesting is that the call and the response, both improvised, are drawn from different ragas. With a common set of notes, the challenge lies in meaningfully linking the phrases during the interaction while also carefully preserving the individual raga-specific characteristics. There are occasional episodes of singing together.

The above discussed form, known as the "*Jasrangi jugalbandi*" after proponent Pt. Jasraj, has been performed over the past decade by a handful of well-known Hindustani vocalists drawing from a limited number of raga pairs [6]. The chosen raga pairs presumably satisfy the music theory and aesthetic requirements for the simultaneous presentation in the jugalbandi concert format. In the present work, we carry out the computational analy-



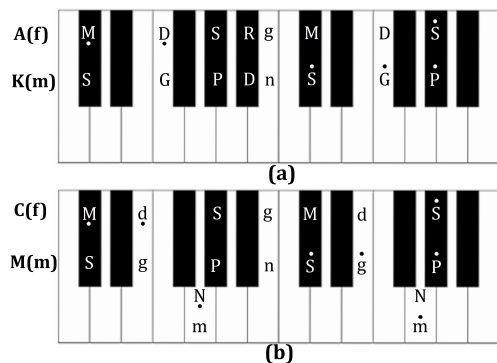


Figure 1. The solfege of each raga superposed on the standard keyboard. ‘S’ denotes the tonic note of the raga. We see that the same set of keyboard notes is shared across ragas in each pair. (a) Abhogi (A) and Kalavati (K); (b) Chandrakauns (C) and Madhukauns (M). Ragas A and C are sung by the female (f) artist of the corresponding pair, ragas K and M by the male (m).

ses of the melodic content of available Jasarangi jugalbandi (shortened to "JJ" in this paper) concerts in two different raga pairs. Specifically, we seek to answer questions around the JJ concerts with reference to the corresponding normally performed individual ragas that can possibly be addressed with computed melodic representations. An important question is the extent to which raga characteristics are preserved when performed in the context of a JJ song. Considering that the interaction between singers is the main highlight of a JJ performance, we are also interested in analysing the melodic relationships of the call and response format. The larger goal for this empirical study is to derive the structure or schema of the JJ concert in terms of the individual raga presentations and the dynamics of the interaction between the two artists. Apart from its value to music appreciation and pedagogy, the insights obtained could serve to identify new raga pairs that potentially fit the format for future JJ performance. In the next section, we present our dataset with descriptions of the represented raga pairs. The manual annotation and the observations directly derivable from this are provided. The melodic representations considered in this work are discussed next. Our experimental analyses are presented with a discussion of the observations and the implications of the outcomes.

2. DATASET & AUDIO PROCESSING

There have been about 7 distinct raga pairs in publicly performed JJ concerts so far. We select, for our work, the two raga pairs that are best represented in the publicly accessible JJ concert recordings. These are the pentatonic raga pairs, *Abhogi-Kalavati* (A-K henceforth) and *Chandrakauns-Madhukauns* (C-M). We also identify a number of the corresponding raga-specific (i.e. individual) concert recordings.

A concert may comprise more than one song (*bandish*), complete with improvisation and chosen composition, and we segment the concert audio accordingly. Our

Feature	Abhogi	Kalavati	Chandrakauns	Madhukauns
Aaroh	SRgMD \dot{S}	SGPDnD \dot{S}	SgMdN \dot{S}	SgmPn \dot{S}
Avaroh	\dot{S} DMgMgRS	\dot{S} nDPGPGS	\dot{S} NdMgMgS \dot{N} S	\dot{S} nPmgS
Vadi, Samvadi	S, M	P, S	M, S	P, S
Nyas	S, R, M, D	S, P, D	S, M, N	S, m, P
Char. Phrases	DSRg, MRS, R \dot{D} SRg, MgRS, MD \dot{S}	SGPD, PDnD, GPD \dot{S} , GPPD, GPGS	SgMgS, NS, gMdN \dot{S} , Nd \dot{S} , NdMgMgS	Sgm, PmgmP, mPn \dot{S} , \dot{S} nPmg S

Table 1. Raga grammar details; dots over and under indicate upper and lower octave pitches respectively

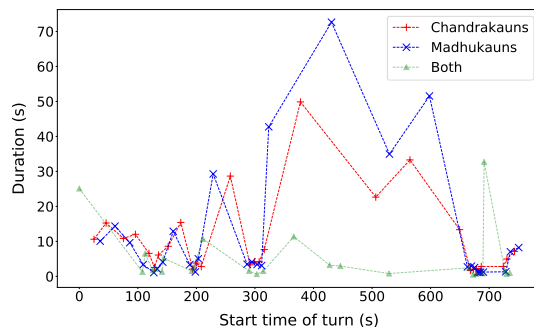


Figure 2. Turn duration versus start time plotted separately for each singer and for the simultaneous (both) singing for a C-M JJ song from our dataset

dataset, summarised in Table 2, comprises recordings from the Hindustani music corpus *Dunya* compiled as part of the *CompMusic* project [7–9] supplemented by available YouTube audio content by well-known Khyal artists, especially for the relatively more scarce JJ concerts. The chosen recordings are vocal performances accompanied by the tanpura (drone), tabla and harmonium. The JJ concerts are manually segmented into male, female and simultaneous-singing episodes or turns Figure 2 illustrates the temporal sequence of singing turn durations across the three labels for one of our C-M JJ songs. We note the dominance of solo singing with roughly equal durations across the two artists, an important characteristic of the JJ format. The simultaneous singing episodes are relatively short and are observed to correspond to the refrain (*mukhda*) of the song and sometimes the long held notes.

Our melodic analysis is based on vocal pitch contours automatically extracted from the audio recordings. The pre-trained 4-stems Spleeter model [10] is used to obtain the vocals component from the stereo mixtures. While not trained on Indian art music, the model produces vocals with either absent, or low enough, levels of the accompanying melodic instruments for the ensuing vocal pitch detection step to work reliably. The regions of the separated vocals, corresponding to the previously labeled solo singing regions, are processed for F0 detection at intervals of 10 ms using an autocorrelation function based method followed by temporal smoothing with search range restricted to the anticipated two octave range of the singer [11, 12]. The resulting pitch time series were checked for accuracy via listening to the resynthesis for any extended duration errors that could be corrected with suitable adjustments of

Raga Pair	Number of songs (minutes)		
	Raga 1	Raga 2	JJ
A-K	12 (185)	12 (227)	5 (89)
C-M	13 (214)	14 (171)	4 (69)

Table 2. Number of songs and total audio duration for each raga and JJ raga-pair in our dataset

the pitch analysis settings. The concert tonic is obtained for each song using an automatic tonic detection method that exploits the presence of the drone in a multipitch detection framework and further manually verified [13, 14].

3. MELODIC REPRESENTATIONS

In Western music, the psychological perception of “key” has been linked to distributional and structural cues present in the music [15]. Likewise, the bare essential theoretical description of a raga lists the allowed pitch classes, the dominant (*vadi*) and sub-dominant (*samvadi*) notes, the resting notes (*nyas*) and the raga motifs or characteristic phrases. Table 1 presents the same for the ragas in our study. The distribution of pitch classes in terms of either duration or frequency of occurrence in scores of Western music compositions has been found to correspond with the perceived tonal hierarchies in different keys [16–19]. That is, the emphasis given to the different notes in a composition is indicative of the underlying musical scale.

First-order, octave-folded pitch distributions (or pitch profiles) computed from the concert recording, and normalised by the concert tonic F_0 , have been widely used in raga identification. Given the pitch continuous nature of the Indian traditions, different kinds of representations and distance measures have been experimentally evaluated in different contexts [1]. Derived from the continuous pitch contour, the instantaneous pitch samples can be binned as such to obtain ‘continuous pitch histograms’. The bin width is an important analysis parameter choice in this case with finer bin widths more fully capturing precise note intonation and note transitions that are possibly distinctive of the raga. Bin widths of up to 25 cents (i.e. 48 intervals per octave) were found to obtain perfect separation in the clustering of allied raga concerts, with performance degrading at larger bin widths [20].

Alternately, a stage of segmentation and quantization can be applied to the continuous pitch time series to obtain ‘stable note’ regions. To account for non-standard note intonations, the underlying scale interval locations or *svara* are estimated from the most prominent peaks of the finely-binned long-term tonic-normalized continuous pitch histogram across the concert. Following [3], segments of the melodic contour of duration greater than 250 ms that display a deviation of within ± 35 cents from a *svara* location, with gaps upto 100 ms discounted, are labeled as stable notes corresponding to the particular *svara*.

Octave-folded histograms are computed for each raga-specific song by accumulating the pitch values across the song audio. The continuous pitch (CP) distribution with

25 cent bin width has a vector dimension of 48. The stable notes histogram has a dimension equal to the number of raga notes or pitch classes. We consider two distinct interpretations for the strength of a pitch class in the song audio, viz. its total duration and the number of occurrences (or count) to obtain two types of stable-note (SN) histogram representations. The previous studies on raga recognition using first-order distributions have considered a variety of distance measures to quantify the similarity between two histograms. For this work, we implement the two measures that have been found to perform best, viz. correlation and Bhattacharya distance [3,9]. We further extend the study to unfolded pitch distributions to investigate how the octave dependence of the realised notes contributes to the discrimination of the mode-shifted ragas in performance.

Apart from pitch profiles, the melodic character of a raga lies in sequential representations including motifs, as is true for other forms of modal music [21, 22]. Similarity of phrase shapes has been exploited in raga recognition using sequence matching techniques [8, 23, 24]. Different ways to deal with the challenges from melodic variation inherent to oral traditions and the imperfect correspondence of any automatically quantized pitch contour to the underlying note sequence give rise to a variety of melodic representations and distance computation methods [25]. For the current task, we employ the continuous pitch contour as well as the extracted stable note (SN) sequence to derive various features that potentially capture the salient characteristics of the JJ song call and response phrases.

4. EXPERIMENTAL RESULTS AND DISCUSSION

Our experiments are targeted towards (i) examining how closely the singers adhere to the raga-specific characteristics of their respective parts in the JJ song while drawing from the identical sets of notes in terms of the standard keyboard as in Figure 1, and (ii) describing the interaction between the two singers to model the phenomenon of ‘call and response’. Raga-specific characteristics are modeled from the individual raga concerts in our dataset. We implement the distinct melodic representations, presented in the previous section, for each raga-specific song as also for the separated raga components of each JJ song. In order to simulate the JJ concert scenario of a single overall concert tonic, we pitch transpose all the concert audios so that the female-sung raga songs (i.e. Abhogi and Chandrakauns) assume a tonic (i.e. the *svara* S) of 207 Hz (G#3 on the standard keyboard) and their complementary raga songs assume C#3 for the tonic, corresponding to the depiction in Figure 1. This ensures that all the represented note pitches in terms of the standard keyboard notes are drawn from the same set for all concerts within a raga pair, making the raga discrimination ambiguous to that extent (just as it would be to the listener of the JJ performance). The experiments, presented next, are organised based on the specific melodic representation employed in the comparisons.

4.1 Pitch distributions

We consider three distinct first-order pitch distributions: the 25-cent bin width continuous pitch (CP) histogram and the stable-note histograms based on duration, SN-d, and count, SN-c. We evaluate both octave-folded and unfolded histogram representations of each type. The fixed tonic normalization gives rise to histograms that are aligned across the two ragas of a pair as in Figure 1. The similarity between two songs is then computed by the distance between their corresponding distributions. Based on the assumption that the individually performed ragas adhere closely to the raga grammar, we use the estimated ‘goodness of clustering’ for the raga-specific songs (i.e. the individual raga concerts only) to obtain a model for the subsequent investigation of the JJ songs. A popular measure of cluster quality is the silhouette coefficient [26]. For the 2-class problem, it is computed as a normalised difference between the mean distance of a point to points in the opposite cluster and the mean distance to other points in its own cluster [27]. It can take on values in $[-1.0, 1.0]$ with higher positive values indicating superior clustering, i.e. the points are better matched to their own cluster members. The silhouette coefficient is computed for each raga-specific song with respect to the two clusters, viz. other songs in the same raga and all songs of the complementary raga.. Finally, an average silhouette coefficient is obtained for each raga pair across all the individually performed songs in either raga, as presented in Table 3.

4.1.1 Discriminating raga-specific songs

All the considered combinations of pitch distribution and histogram distance measure appear in Table 3. A number of observations are apparent. We obtain positive valued coefficients in all cases implying that the tonal hierarchy is sufficiently differentiated within a pair even with identical keyboard notes and fixed tonic. For both raga pairs, the unfolded representations yield superior clusters, with the distinctions between other variations being somewhat less marked. The unfolded distribution is, of course, helped by the pitch transposition effected between the ragas, as reviewed earlier in this section. The distance measures, on the other hand, exhibit differences. As for the octave-folded representations, only the A-K ragas demonstrate good separation with the stable-note duration (SN-d) representation taking on the highest values for both the distance measures. A possible explanation for reduced separation in C-M is the overlapping nayas (rest notes) across the 2 ragas. We can conclude that of the considered methods of comparing pitch distributions, the Bhattacharyya distance between SN-d representations overall best separates the individual songs across the two ragas of the complementary raga pairs with the octave unfolded representation doing better. This therefore forms our computational model of melodic similarity to be applied to the investigation of raga components of the JJ songs, as described next.

4.1.2 Discriminating ragas performed in JJ songs

The silhouette coefficient computed using the above model on a JJ song raga component (i.e. the song part rendered

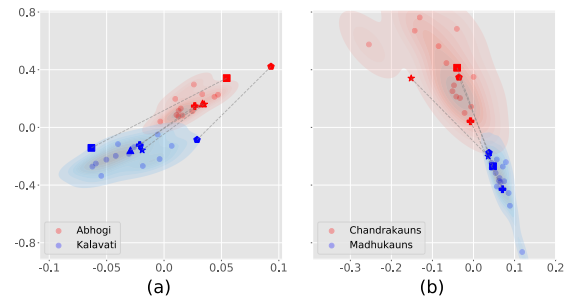


Figure 3. MDS scatter plots for A-K and C-M raga pairs with unfolded SN-d representations and Bhattacharyya distance. All raga-specific songs appear as uniform filled circles with the corresponding kernel density estimates superposed. The JJ songs are indicated with distinct symbols of shared shape but different raga-specific colours for the two raga components of the same song, also connected with a dashed line.

by a specific singer) with respect to the corresponding two raga-specific song clusters can tell us about its “faithfulness” to its own raga characteristics in the JJ context. We obtained positive valued silhouette coefficients across the set of JJ song components for all the representations in Table 3, but report here only the values computed with the unfolded SN-d and Bhattacharyya distance measure, viz. 0.75 (A-K) and 0.77 (C-M).

Further, a more visual rendition of the similarities between JJ song components and their own class of raga-specific songs is possible with multi-dimensional scaling (MDS) [27, 28]. The chosen representation (unfolded SN-d) and distance measure (Bhattacharyya) are used to obtain all inter-song distances in the corresponding multi-dimensional space of the representation which is then projected to a two-dimensional space where the similarity between items is preserved in the visual distances using MDS. Figure 3 presents the MDS plots for each raga pair separately. The different colours in each plot separate the two ragas of the corresponding pair. The JJ song components are depicted with special symbols that distinguish them from the raga-specific songs but also serve to identify the same-song components. We note that the raga-specific songs as well as the JJ component songs of the A-K are well separated in the 2 dimensional space. The two raga components of the same JJ song tend to be at least as well separated as the most closely spaced raga-specific songs drawn from opposite ragas. The similar observations hold for the C-M dataset indicating the preservation of the distributional characteristics of the raga in the JJ songs.

4.2 Melodic phrases

Our goal is to study the interaction between the two JJ singers in terms of the nature of ‘call’ and ‘response’ phrases. We isolate the individual singing turns from the JJ song pitch time-series and create pairs out of consecutive turns with one each from the female and male singers. The pairing is achieved from the manual annotation of the singers’ parts as described in Section 2 and starts from the first singing turn in the song. The interaction, if any, is

Raga Pair	Metric	Folded			Unfolded		
		CP (48)	SN-d (5)	SN-c (5)	CP (128)	SN-d (14)	SN-c (14)
A-K	Correlation	0.47	0.54	0.35	0.55	0.55	0.61
	Bhattacharyya	0.36	0.53	0.31	0.82	0.78	0.79
C-M	Correlation	0.16	0.19	0.03	0.71	0.68	0.68
	Bhattacharyya	0.11	0.18	0.08	0.87	0.84	0.83

Table 3. Average silhouette coefficient for raga-specific songs in each raga pair for different representations and distance measures. (CP: continuous pitch; SN-d and SN-c: stable notes weighted with duration and count respectively. The corresponding vector dimension appears in parentheses.)

expected to be more evident with shorter duration turns. While any JJ song has singer turns with a wide range of durations, short turns (10 s or lower) are relatively more frequent in the faster tempo (*drut laya*) songs. We therefore restrict this part of the study to the drut songs in our dataset, i.e. 4 A-K songs (83 turn pairs in total) and 3 C-M songs (80 turn pairs in total), omitting altogether only one JJ song from each of the raga pairs in our dataset. Figure 2 shows the variation of the duration of a turn with start time of the turn in a drut song. The presence of rapid pitch modulations (taans) in the drut songs leads to fewer detected stable notes. We therefore relax the duration threshold used in their extraction from the continuous pitch contour to 150 ms.

We begin by examining the local pitch distributions at the turn level. This allows us to compare the tonal content across the two turns in a call and response pair. The two singing turns in a pair are assigned the same ‘index’. Figure 4 presents 2 JJ song examples, one from each of our raga pairs. We see the time-varying distribution of the stable notes in terms of their durations in a given turn. An immediate observation is the fairly consistent vertical displacement in the range of notes covered in going from one raga turn to the same-index turn in the complementary raga. Further, it appears that the shape of the distribution of notes per turn, as it changes with turn index, is roughly matched across the 2 singers with a visually somewhat closer match in the case of the C-M song.

4.2.1 Melodic shape features

In order to obtain a more quantitative analysis comparing call and response across turn pairs, we define a few meaningful scalar features that can be reliably computed from the melodic pitch contours. The previously paired turns are assumed to comprise a call and its response. We compute the linear correlation between the corresponding features of the two turns in a pair across all the pairs in the A-K drut songs; similarly, the C-M drut songs. The selected features are the turn duration (in seconds), the number of notes in the SN sequence representation of the turn and the pitch range spanned by the turn as computed from the corresponding continuous pitch contour. The estimated correlations obtained for each raga pair appear in Table 4 as contrasted with the corresponding correlations between randomly paired turns averaged over 50 shuffles of the set. We note moderately high positive correlations for all the features, suggestive of the similarity between the melodic contours of the paired turns.

Raga pair	No. of turn pairs	Duration (s)	No. of notes	Pitch range (cents)
A-K	83	0.52 (0.02)	0.61 (0.02)	0.47 (0.04)
C-M	80	0.81 (0.02)	0.64 (0.03)	0.55 (0)

Table 4. Correlations between matched-index turns from the 2 ragas for turn duration, number of notes and pitch range, averaged over the turns in each raga pair. Values in parentheses are the corresponding correlations across randomly paired turns, serving as a baseline.

4.2.2 Transposition interval

With the turn-level pitch distributions of Figure 4 suggestive of a fixed pitch interval shift between turns in a call and response pair, we attempt to estimate the transposition interval. In order to determine a suitable computational model for this, we manually examined the continuous pitch contours of a few call and response pairs where the singers used the raga solfege as lyrics, providing us with a ready transcription of the phrase in terms of the raga notes.

Figure 5 shows instances of two distinct kinds of call and response interactions observed in a C-M JJ song. We have (a), where the turns correspond to the identical keyboard pitch classes, i.e. the singers are actually in unison (or one octave apart). In such a case, the solfege transcriptions are quite different across the phrases, as expected from different ragas and different assumed tonics. While the notes uttered are valid for each raga given its assumed tonic, the realised phrase is not necessarily a raga characteristic phrase. The more common pattern, however, is (b), where the turns have similar solfege notation, i.e. the singers utter (mimic) the same solfege notes (svara) as far as possible. When the svara is not available to the responding singer, they draw from the closest available svara of their raga (verifiable from the raga grammar of Table 1). The case (b) is expected to correspond to an exact transposition by a fifth (700 cents), a desirable state of harmony (*samvaad*), in the case of svara common to the two ragas but only approximately so otherwise. We also occasionally observe geometric transformations such as melodic contour inversion or reversal across call and response.

Given the known challenges in transcribing the continuous melodic contours to solfege notation (evident also in Figure 5), we use the simplified, even if crude, measure of the mean pitch (in Hz) of a turn to estimate the pitch offset between the corresponding phrases. Figure 6 presents a histogram computed for each raga-pair showing the distri-

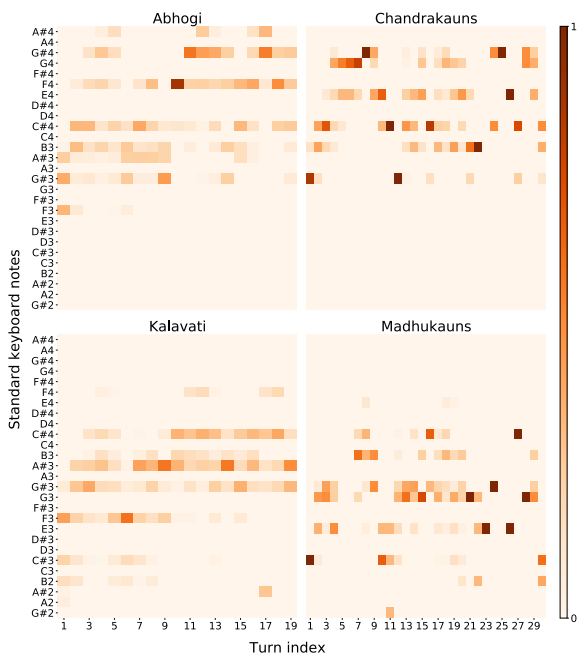


Figure 4. Distribution of stable notes in a turn versus turn index, separately (top and bottom) for each singer, in an A-K JJ song (left) and a C-M JJ song (right). The song notes are provided in terms of standard keyboard notes with normalizing F0 selected so that the assumed tonic of the female component corresponds to G#3, and therefore that of the male voice to C#3.

bution of the estimated intervals between the mean pitches of turns in a call and response pair. We note that both raga pairs show distributions concentrated around the interval of a fifth. While the C-M distribution is centered around 700 cents, A-K is more skewed to lower intervals. To explain the relatively high A-K histogram peak at (a relatively dissonant) 600 cents, we examined turn pairs that exhibited this specific mean pitch difference. A prominent example of this turned out to be phrases that matched ‘g’ in Abhogi with ‘G’ in Kalavati. Another case was the upward going ‘D-Ś’ transition in Abhogi being matched with a ‘D-n-Ś’ transition in Kalavati. In any case, given that we do not have precise within turn alignments of the svara rendered in call and response, we can only infer that the phrases in a pair are largely centered at matching pitch intervals relative to their respective assumed tonics. With the two tonics offset by a fifth, we see the transposition by 700 cents. The occurrence of mean pitch differences of values below and above 700 cents in Figure 6 may be explained by our observation that the automatic pairing of call and response turns is occasionally incorrect with singers switching roles at times.

5. CONCLUSION

We have presented here what is probably the first empirical analysis of mode-shifting (murchana) in Hindustani

Supplementary material available at this [link](#).

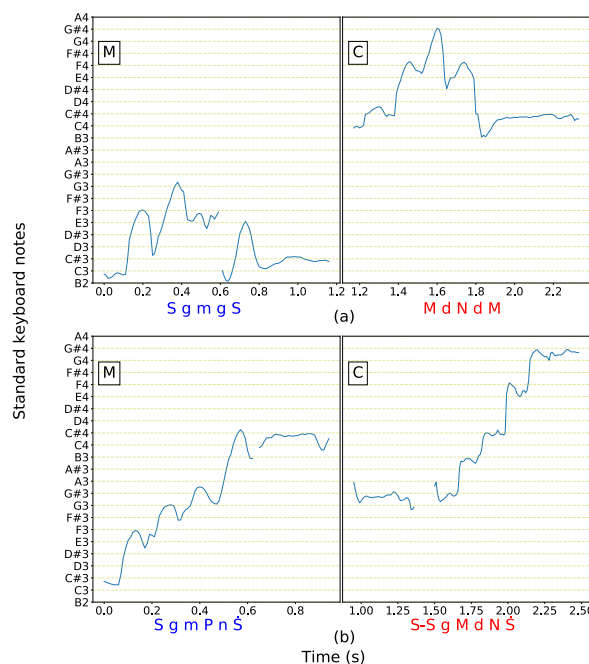


Figure 5. Two examples of call and response patterns from a CM JJ song. (a) replicating the keyboard notes of the call phrase, but an octave apart; (b) approximating the solfege of the call phrase relative to own assumed tonic.

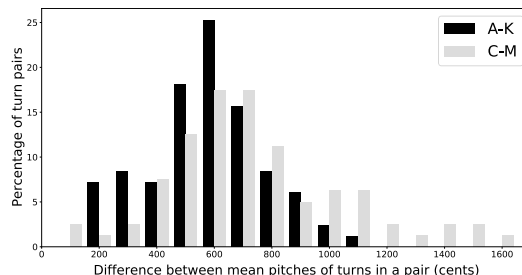


Figure 6. Histograms of the differences between the mean pitches between turns in a pair, computed for all turn pairs corresponding to A-K (83 turn pairs) and C-M (80 turn pairs) drut songs.

raga performance using MIR tools. The specific form considered was the Jasrangi jugalbandi where two artists collaborate, each singing a different raga with the same set of pitches shared across the two. Computed song-level pitch distribution similarity revealed that the JJ singers adhere to raga characteristic tonal hierarchy to the same extent as in the corresponding individually performed ragas. Simple melodic features of the singing turns during the artists’ interaction revealed interesting insights about the melodic shape transformations and transposition interval in the course of the call and response. Overcoming the limitations posed by data scarcity would facilitate the investigation of more complex models of melodic similarity that also take note sequences into account.

6. ACKNOWLEDGEMENTS

We thank Ankita Joshi, disciple of Pandit Jasraj, for her guidance and useful discussions.

7. REFERENCES

- [1] G. K. Koduri, S. Gulati, P. Rao, and X. Serra, "Rāga recognition based on pitch distribution methods," *Journal of New Music Research*, vol. 41 (4), pp. 337–350, 2012.
- [2] P. Chordia and A. Rae, "Raag recognition using pitch-class and pitch-class dyad distributions," in *Proc. of the 8th Int. Soc. for Music Information Retrieval Conference*, Vienna, Austria, 2007.
- [3] K. K. Ganguli and P. Rao, "On the distributional representation of ragas: Experiments with allied raga pairs," *Transactions of the Int. Soc. for Music Information Retrieval*, vol. 1 (1), pp. 79–95, 2018.
- [4] S. Bagchee, *Nad: Understanding Raga Music*. Business Publications Inc., 1998.
- [5] H. Hirlekar, *Nuances of Hindustani Classical Music*. Unicorn Books, 2010.
- [6] M. Desai, "Jasrangi jugalbandi: A wondrous show of musical acumen," <https://creativeyatra.com/reviews/jasrangi-jugalbandi-a-wondrous-show-of-musical-acumen/>, 2018, accessed: May-2021.
- [7] X. Serra, "A multicultural approach in music information research," in *Proc. of the 12th Int. Soc. for Music Information Retrieval Conference*, Miami, USA, 2011.
- [8] S. Gulati, J. Serrà, V. Ishwar, S. Şentürk, and X. Serra, "Phrase-based rāga recognition using vector space modeling," in *Proc. of the 41st IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Shanghai, China, 2016.
- [9] S. Gulati, J. Serrà, K. K. Ganguli, S. Şentürk, and X. Serra, "Time-delayed melody surfaces for rāga recognition," in *Proc. of the 17th Int. Soc. for Music Information Retrieval Conference*, New York City, USA, 2016.
- [10] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, p. 2154, 2020.
- [11] Y. Jadoul, B. Thompson, and B. de Boer, "Introducing parselmouth: A python interface to praat," *Journal of Phonetics*, vol. 71, pp. 1–15, 2018.
- [12] P. Boersma and D. Weenink, "Praat: doing phonetics by computer [Computer program]," <http://www.praat.org/>, 2021, version 6.1.38, retrieved May 2021.
- [13] J. Salamon, S. Gulati, and X. Serra, "A multipitch approach to tonic identification in indian classical music," in *Proc. of the 13th Int. Soc. for Music Information Retrieval Conference*, Porto, Portugal, 2012.
- [14] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor *et al.*, "Essentia: an audio analysis library for music information retrieval," in *Proc. of the 14th Int. Soc. for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.
- [15] D. Huron, *Sweet Anticipation: Music and the Psychology of Expectation*. MIT Press, 2006.
- [16] C. L. Krumhansl and L. L. Cuddy, *Music Perception*. New York: Springer, 2010, ch. A Theory of Tonal Hierarchies in Music, pp. 51–87.
- [17] N. A. Smith and M. A. Schmuckler, "Pitchdistributional effects on the perception of tonality," in *Proc. of the 6th Int. Conf. on Music Perception and Cognition*, Staffordshire, England, 2000.
- [18] R. Raman and W. J. Dowling, "Real-time probing of modulations in south indian classical (carnatic) music by indian and western musicians," *Music Perception: An Interdisciplinary Journal*, vol. 33 (3), pp. 367–393, 2016.
- [19] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17 (1), pp. 65–100, 1999.
- [20] K. K. Ganguli and P. Rao, "Towards computational modeling of the ungrammatical in a raga performance," in *Proc. of the 17th Int. Soc. for Music Information Retrieval Conference*, Suzhou, China, 2017.
- [21] B. Cornelissen, W. Zuidema, and J. A. Burgoyne, "Mode classification and natural units in plainchant," in *Proc. of the 21st Int. Soc. for Music Information Retrieval Conference*, Montréal, Canada, 2020.
- [22] S. Sertan and P. Chordia, "Modeling melodic improvisation in turkish folk music using variable-length markov models," in *Proc. of the 12th Int. Soc. for Music Information Retrieval Conference*, Miami, USA, 2011.
- [23] S. Dutta, K. S. PV, and H. A. Murthy, "Raga verification in carnatic music using longest common segment set," in *Proc. of the 16th Int. Soc. for Music Information Retrieval Conference*, Malaga, Spain, 2015.
- [24] S. T. Madhusudhan and G. Chowdhary, "Deepsrgm - sequence classification and ranking in indian classical music with deep learning," in *Proc. of the 20th Int. Soc. for Music Information Retrieval Conference*, Delft, The Netherlands, 2019.
- [25] N. Kroher, A. Pikrakis, and J.-M. Díaz-Báñez, "Discovery of repeated melodic phrases in folk singing recordings," *IEEE Transactions on Multimedia*, vol. 20 (6), pp. 1291–1304, 2018.
- [26] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

- [27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [28] I. Borg and G. P., *Modern Multidimensional Scaling - Theory and Applications*. Springer, 1997.

SINTRa: LEARNING AN INSPIRATION MODEL FROM A SINGLE MULTI-TRACK MUSIC SEGMENT

Qingwei Song¹ Qiwei Sun² Dongsheng Guo¹ Haiyong Zheng^{1*}

¹ College of Electronic Engineering, Ocean University of China, China

² School of Microelectronics, Xi'an Jiaotong University, China

* Corresponding author: zhenghaiyong@ouc.edu.cn

ABSTRACT

In this paper, we propose SinTra, an auto-regressive sequential generative model that can learn from a single multi-track music segment, to generate coherent, aesthetic, and variable polyphonic music of multi-instruments with an arbitrary length of bar. For this task, to ensure the relevance of generated samples and training music, we present a novel pitch-group representation. SinTra, consisting of a pyramid of Transformer-XL with a multi-scale training strategy, can learn both the musical structure and the relative positional relationship between notes of the single training music segment. Additionally, for maintaining the inter-track correlation, we use the convolution operation to process multi-track music, and when decoding, the tracks are independent to each other to prevent interference. We evaluate SinTra with both subjective study and objective metrics. The comparison results show that our framework can learn information from a single music segment more sufficiently than Music Transformer. Also the comparison between SinTra and its variant, i.e., the single-stage SinTra with the first stage only, shows that the pyramid structure can effectively suppress overly-fragmented notes.

1. INTRODUCTION

The current development trend of music generation is to generate harmonious multi-track music with longer-term dependency. However, in the composition of real life, the inspiration is the beginning of a song, and the composer usually creates a music based on a single music segment that comes to mind. Thus, it's more important to find ideas that are relevant to the inspiration.

As for composers, the composition process can be divided into two stages. The first stage is to generate a large number of ideas that provide inspiration for subsequent creation. The second stage is about idea convergence. Composers need to find ideas that can express their feelings and emotions to the audience from a large number of ideas, then expand, repeat and arrange them, and

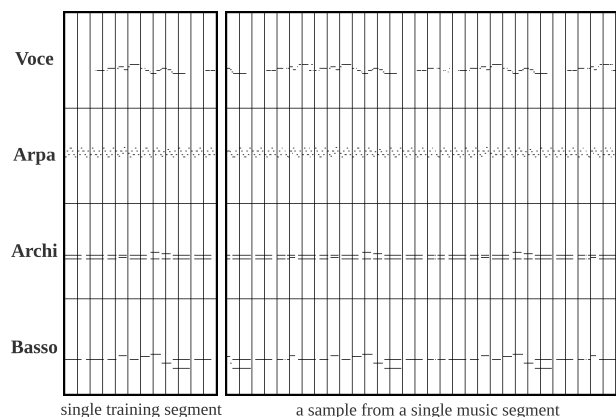


Figure 1. One sample of four-track piano-roll (right) with 32-bar length (each block represents a bar), generated by our SinTra model trained from a single music segment (left). The y-axis and x-axis represent note pitch (range from 0 to 127) and time step T , respectively.

finally end up with a song. At the first stage, generated ideas constitute small segments of music, from which to compose more similar segments can provide more inspiration for composers. Therefore, as to music generation, it's significant to learn an inspiration model, that is, a generative model to generate music, inspiring the composers for creating a song according to a single music segment.

Recently, music generation has witnessed great progress due to the development of deep learning technologies. The mainstream methods are modeling the note sequences by drawing lessons from language models in natural language processing (NLP), which require a large number of MIDIs as training set to learn the distribution of notes. Generally, the training period is long, and the randomness of generated music is relatively high. However, if only a single music segment is available for training, it's hard for previous models to acquire enough information for learning reasonable musical structure and relation position relationship between notes, resulting in chaotic and aesthetically unpleasant music (refer to Section 6).

At present, some one-shot generation works [1, 2] followed the multi-scale training mechanism and achieved compelling results, which allow the network to learn the information from single training data in different scales more sufficiently. Besides, recent works in music gener-

ation [3–5] adopted the Transformer-XL [6], an improved variant of the Transformer [7], to introduce recurrence to the architecture, as the backbone sequence model.

In this paper, we leverage the multi-scale training scheme and the Transformer-XL architecture, to tackle the more challenging and meaningful one-shot music generation, that is, generating music from a single multi-track music segment. To ensure the relevance of generated music and training segment, we present a novel pitch-group representation to contain all pitch group types of the single training music segment. Besides, in order to deal with more complex multi-track music, we design three modules in each stage (scale) of our multi-scale training.

To summarize, SinTra can actually be regarded as an inspiration model for music composition. When the composer is lacking in inspiration, or in creation, it would be very repetitive to make detailed adjustments at the structure level, and SinTra can help. Certainly, the subsequent fine-tuning still needs to be done by humans. We make four contributions: (1) A novel pitch-group representation is presented to model polyphonic music of single instrument into a sequence; (2) A novel inspiration model, namely SinTra, is devised to generate meaningful music from only a single music segment; (3) Three modules based on Transformer-XL are designed for each stage of multi-scale training to process multi-track music. (4) The source code¹ and music data are made publicly available.

2. RELATED WORK

2.1 Music Generation

Music generation, as a niche research task of music information retrieval (MIR), has a long history and has attracted great attention in both industrial and art communities recently.

As a traditional method, Markov models are often used in the field of MIR, such as the work from Simon et al. [8] and Tsushima et al. [9]. Chuan et al. [10] have also used support vector machine (SVM) to select chord tones from given melodies. Then, recurrent neural network (RNN) [11] with long short-term memory (LSTM) [12] and gated recurrent unit (GRU) [13], variational auto-encoder (VAE) [14], and generative adversarial network (GAN) [15], are common deep learning frameworks used for modeling music sequence. MuseGAN [16] generated music as an image (converting MIDI into piano-roll) with GANs, and used an inter-track latent vector to make the generated multi-track music coherent. To overcome the binarization issue, the upgraded version of MuseGAN, Binary MuseGAN [17] proposed an additional refiner network, which enables generator to directly generate binary-valued piano-rolls at test time. Moreover, two types of binary neurons (BNs) considered features fewer overly-fragmented notes as compared to MuseGAN. MIDI-Sandwich2 [18], which also used piano-roll, applied a hierarchical multi-modal fusion generative VAE network

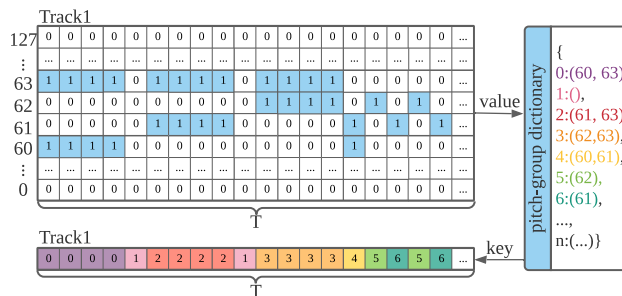


Figure 2. Illustration of our pitch-group representation. The pitch-group dictionary is built and regarded as the database of music segment containing $n + 1$ key-value pairs, where the key (e.g., 0) means the index of pitch group type and the value (e.g., (60, 63)) means the pitch group information. Piano-roll ($T \times 128$) can be mapped to token sequence ($T \times 1$) through pitch-group dictionary (T means time step).

based on RNN to collaboratively generate multi-track symbolic music. XiaoIce Band [19], a melody and arrangement generation framework for pop music, introduced cooperate GRUs between each generation track to generate melody and multi-track music arrangement. DeepJ [20], based on Bi-LSTM [21], was trained using piano-roll for style-specific music generating (baroque, classical, and romantic). Different from previous music generation models, our work devotes to learning a model to generate music from only a single segment.

2.2 Transformer and Multi-scale Training

Compared to LSTM or GRU, Transformer, a sequence model based on multi-head self-attention mechanism, is more parallelizable for both training and inferring, and more interpretable [7]. Transformer has achieved compelling results in tasks that require maintaining long-range dependencies, such as neural machine translation [7], pre-training language models [22], text-to-speech synthesis [23], and speech recognition [24].

For music generation, Music Transformer [25] was the first work that applies the Transformer to symbolic music generation, Huang et al. used relative positional encoding [26] within the original Transformer architecture to capture relative timing information. MuseNet [27] used sparse kernels [28] to remember the long-term structure in the composition. More recent works [3–5] adopted Transformer-XL [6] that uses recurrent memory to enable the model to attend beyond a fixed context. In this work, we also leverage the powerful long-term dependency modeling of Transformer-XL for one-shot music generation.

Notably, SinGAN [1] has achieved compelling results on the task of unconditional generation from a single natural image, via a pyramid of fully convolutional light-weight GANs in a coarse-to-fine fashion. Then TOAD-GAN [2] was proposed for coherent style level generation following the one-shot training approach of SinGAN. Our work also involves the multi-scale training scheme into our SinTra model for better training from a single music segment.

¹ <https://github.com/qingweisong/SinTra>

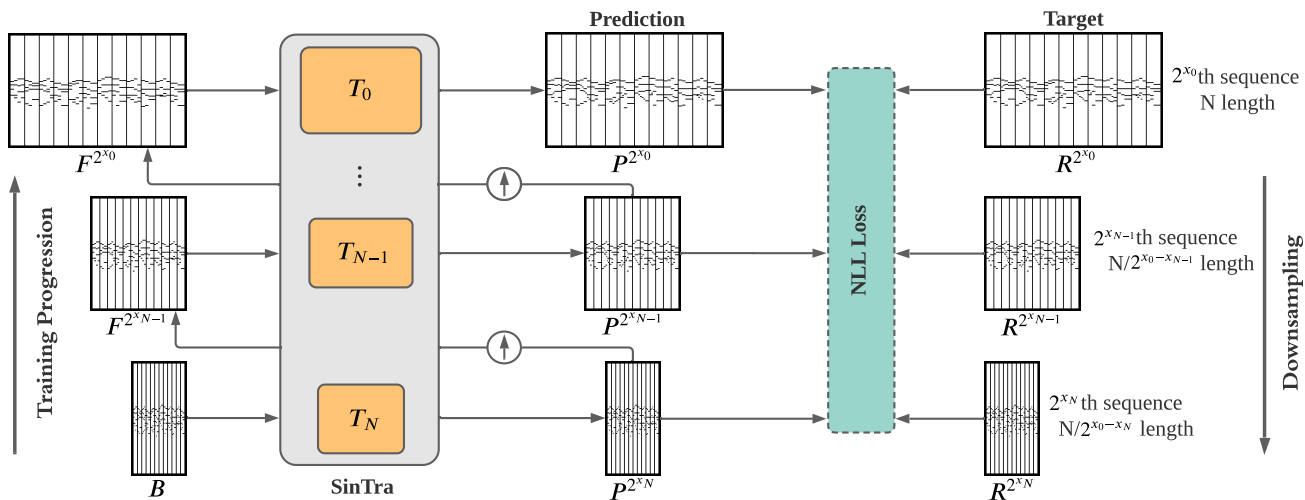


Figure 3. SinTra’s multi-scale pipeline. Our model consists of a pyramid of Transformer-XLs, where both training and inference are done in a coarse-to-fine fashion. At each scale, T_n learns the information of target $R^{2^{x_n}}$ by NLL loss, 2^{x_n} th sequence means the sequence sampled by the 2^{x_n} th note (visualize the sequence into piano-roll, $2^{x_0} > \dots > 2^{x_N} \geq 4$). The input $F^{2^{x_n}}$ to T_n from the previous output $P^{2^{x_{n+1}}}$, upsampled to the current temporal resolution (except for the coarsest stage which comes from B, a downsampled version of the real music). The generation process at stage n involves all Transformer-XLs $\{T_N, \dots, T_n\}$ up to this level.

3. DATA REPRESENTATION

We use the method of language modeling to train the generation models for symbolic music. Therefore, by serializing polyphonic music into a single sequence, we express music as a series of discrete symbols determined by the data in music. For music generation learning from a single music segment, we present a novel pitch-group representation, which uses the index to represent various pitch group types by flexibly building a pitch-group dictionary of key-value pairs. The principle of pitch-group representation and the construction of pitch-group dictionary are shown in Figure 2. For a segment of music, the dictionary of pitch-group representation will not be very complex, so this representation method is feasible.

We treat all types of pitch group in each time step as elements, such as *harmony*, *single tone* and *interval*, which can use 1-dim sequence to represent polyphonic music. And the pitch-group representation can be regarded as an upgraded representation of pitch-based representation (support only monophonic music). However, this method is a double-edged sword, which will limit the output space and affect the diversity of generated music.

Piano-roll and event-based are the two most commonly data representations. Piano-roll representation used in DeepJ [20], MuseGAN [16], Binary MuseGAN [17], MIDI-Sandwich2 [18], and Music Transformer [25], is a 5-dim matrix representation of music where the vertical and horizontal axes respectively represent note pitch and time step. However, the piano-roll matrix is sparse since there are many zeros, only a few notes are attacked during each time step. Gale et al. [29] proved that sparse matrix has great computational potential. Treating piano-roll directly as dense matrix processing will waste computing resources.

Event-based representation used in Music Transformer [25] and LakhNES [3], means that the MIDI note events are converted into a sequence of tokens by a vocabulary containing 388 events. A one-minute song may need about 900 tokens in event-based representation. When the temporal resolution is 16th note and the tempo is 120 bpm, the piano-roll is a matrix whose shape is (480, 128) and the pitch-group is a sequence whose length is 480. It means that the length of the event-based is twice that of the other two methods. Besides, although events are generated in probability order, when there is not enough training data, it’s easy to generate unreasonable note (e.g., *Note_on* event of the same note is generated before the *Note_off* event or super long note). In addition, *TIME_SHIFT* can possibly cause the confusion of time value information as proposed by Wu et al. [5].

Since traditional representation considers the universality of music representation, the coding space utilization is low when representing the information of a specific segment of music. For example, a segment of music only uses 20 pitches but still needs to use 128 pitch coding spaces. Or only 80 events appeared, but the encoding space of 388 events still needs to be used.

4. MUSIC GENERATION LEARNED FROM A SINGLE MUSIC SEGMENT

4.1 Pyramid of Transformer-XL Model

For learning reasonable musical structure and relative position relationship between pitch-group indexes of a single music segment, we adopt the multi-scale training mechanism to design a pyramid of Transformer-XLs $\{T_0, \dots, T_N\}$. Figure 3 shows the pipeline of SinTra for the generation of music samples. SinTra is trained

with a sequence pyramid of each stage’s real music R : $\{R^{2^{x_0}}, \dots, R^{2^{x_N}}\}$ by the Negative Log Likelihood (NLL) loss, where $R^{2^{x_n}}$ is a downsampled version of $R^{2^{x_0}}$. Each Transformer-XL T_n is responsible of producing music samples P^{2^n} with the corresponding scale of $R^{2^{x_n}}$. The generation of a music sample starts at the coarsest scale and sequentially passes through all models up to the finest scale. The Transformer-XLs have the different processing length and thus capture more details as we go up the generation process.

In order to get the sequence pyramid of R for each stage, we use different note-values to sample the original sequence. This kind of down-sampling method preserves the coarse-grained music structure information of the training music. We artificially define the note-value of each stage as 2^{x_n} th, and the scale of the corresponding stage is $\frac{N}{2^{x_0-x_n}}$. For our training, the 16th note sequence of N length is sampled down to $\frac{N}{2}$ and $\frac{N}{4}$ by the 8th note and the 4th note respectively.

4.2 Processing of Multi-track Sequences

All the models of each scale have a similar architecture, as depicted in Figure 4. For maintaining the inter-track correlation, we use convolution operation to process multi-track music, and when decoding, the tracks are independent of each other to prevent interference. The `Track_multi2one` module is used to map multi-track sequence into a single sequence, and the `Track_one2multi` is the inverse process of `Track_multi2one`. The `Track-wise Decoder` module is used to decode each track independently.

4.3 Training and Inference

The first transformer generates bar by bar sequentially and the others refine each bar in a coarse-to-fine manner. In the 1st scale of training, the sequence sampled by the 4th note of the t^{th} bar R_t^4 is fed into the model, and the $(t+1)^{th}$ bar R_{t+1}^4 is taken as training target with NLL loss. In the 2nd scale of training, to get the 8th sequence F_{t+1}^8 , the output P_{t+1}^4 to the previous scale needs to be upsampled, and the model will be trained with R_{t+1}^8 as output. In the same way, the 3rd scale needs F_{t+1}^{16} as input and to be trained with output R_{t+1}^{16} . And the final output is P_{t+1}^{16} from the 3rd scale. The overall training process is shown in Eqn (1).

$$\begin{aligned}
 1st : P_{t+1}^4 &= Model_{1st}(R_t^4) \\
 loss_{1st} &= NLL(P_{t+1}^4, R_{t+1}^4) \\
 2nd : F_{t+1}^8 &= Upsample(P_{t+1}^4) \\
 P_{t+1}^8 &= Model_{2nd}(F_{t+1}^8) \\
 loss_{2nd} &= NLL(P_{t+1}^8, R_{t+1}^8) \\
 3rd : F_{t+1}^{16} &= Upsample(P_{t+1}^8) \\
 P_{t+1}^{16} &= Model_{3rd}(F_{t+1}^{16}) \\
 loss_{3rd} &= NLL(P_{t+1}^{16}, R_{t+1}^{16})
 \end{aligned} \tag{1}$$

In the inference of the NLP model, if the highest probability result is used as the prediction result every time, the

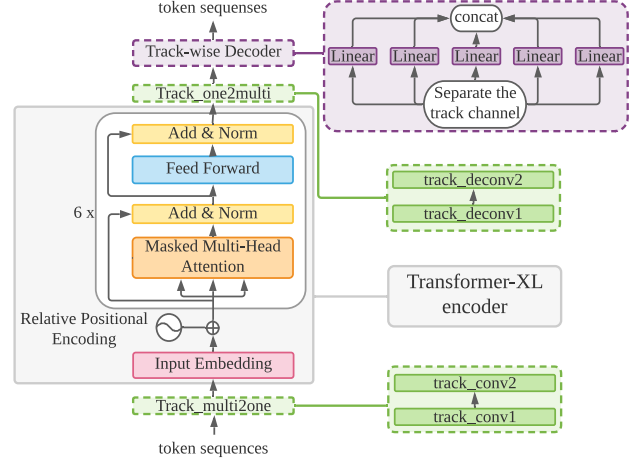


Figure 4. Network structure of each scale T_n (left) and three modules related to multi-track sequence processing (right). After the process of `Track_multi2one`, the shape of input sequence $(1, track, T)$ becomes to $(1, T)$. The shape of the Transformer-XL encoder output $(1, T, Feature)$ becomes to $(1, Track, T, Feature)$ after `Track_one2multi`. Via the `Track-wise Decoder`, the final output token sequence shape is $(1, Track, T)$.

generated content will repeat easily, this method is called `Top1`. Hence, we adopt the `Top p` method proposed by Holtzman et al. [30], that is, the model will sample the predicted results from several of the most likely results. In our inference process, the 1st scale is used to predict the next bar, so we can set a larger $p = 0.9$ to increase the diversity of the generated samples. However, the latter two scales are used to enhance the details, if the variety is robust, the generated content will become overly-fragmented, so we set a smaller $p = 0.3$.

5. EXPERIMENT SETUPS

5.1 Data

We test our method both qualitatively and quantitatively on a variety of music files in MIDI format, containing well-known works of multiple styles of music. The MIDIs that we used are taken from the JSB Chorale dataset [31], classical music used in C-RNN-GAN [32] from the website². We only use the JSB Chorale dataset for objective evaluation, because Music Transformer only supports single-track music. And we use all the MIDIs for subjective study.

5.2 Model Configurations & Training Setup

We performed our experiments under an NVIDIA GeForce GTX TITAN X graphics card, with PyTorch 1.4.0 running under CUDA 11.2. We implemented our multi-scale training framework (Figure 3) based on the Transformer-XL encoder. The encoder has 6 layers and the number of heads is 8 with a dimension of 32. The dimension of the embedding layer is 256 and the hidden layer dimension is 1024.

²<https://www.classicalarchives.com/>

The dropout ratio is set at 0.09. The length of training input tokens (processing length) and the memory length of 3 stages are 4, 8, 16, respectively.

Considering downsampling, to make SinTra learn the single training segment sufficiently, it is necessary to choose the note-value of each scale reasonably. The shorter notes appear in the training music segment, the smaller note-value used for sampling needs to be set in the finest scale, and the number of stages required for training is larger. Besides, the note-values of the adjacent scale are preferably a 2-fold relationship. Formally, the note-value of the finest scale is set to the shortest note that appears, the note-value of the 1st stage is set to the 4th note value (standard time unit in music).

We choose Music Transformer³ and our variant, the model with only the first stage named single-stage SinTra, whose temporal resolution is set to the 16th note value, for comparison. We use Adam optimizer with $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\epsilon = e^{-8}$ and follow the same learning rate schedule in Transformer-XL [6]. We set the number of input bars as 12 and the number of generated bars as 32.

5.3 Subjective Study

We set up a blind listening test for human evaluation in which test-takers listen to 7 segments of music, one from the real music, three from SinTra and three from Music Transformer. In the test, test-takers will be asked the same set of questions after listening to each of the two test groups, namely, to rate them on a five-point scale about the following aspects:

- **Quality (Q):** Does the generated music sound pleasing overall?
- **Relevance (R):** Whether the generated music give you the same feeling as the real music?
- **Diversity (D):** Does the generated music have new arrangement that impresses you?

Finally, we collect responses from 50 subjects, of which 20 are classified as professional composers for their musical background. The 20 professions were asked to rate each music segment they heard from the music composition theory aspect, while 30 non-composers were asked to rate their subjective feelings.

5.4 Objective Evaluation

Objective evaluation in music generation is still an open question, though various metrics have been proposed, the feeling of music varies from person to person, it's hard to measure the quality of generated music. To quantitatively compare the performance differences between the three models, we use the following metrics to measure the similarity and diversity between generated music samples and the realistic single music segment.

³ Since the official code is highly coupled with Magenta, data processing and training scripts are not shown explicitly. We thus used a third-party implementation (<https://github.com/jason9693/MusicTransformer-pytorch>) instead.

	Quality	Relevance	Diversity
SinTra	3.20/5.00	3.66/5.00	2.86/5.00
Music Trans.	2.34/5.00	2.38/5.00	2.54/5.00

Table 1. Results of subjective study in a five-point scale.

5.4.1 KL Divergence

KL divergence measures the distance between two distributions. In this paper, pitch group indexes are used as the essential element for calculating the music distribution. The way we measure similarity is by calculating KL divergence between distributions of pitch group:

$$D_{kl}(P||Q) = \frac{1}{N_{sample}} \sum_{j=0}^{N_{sample}} \sum_{i=0}^{N_{type}} P(i) \log_2 \left(\frac{P(i)}{Q(i)} \right), \tag{2}$$

where N_{sample} means the number of generated samples, N_{type} means the number of pitch group types, $P(i)$ means the i^{th} pitch group type of the generated samples, $Q(i)$ means the i^{th} pitch group type of the original music. The smaller the KL divergence, the closer the two distributions.

5.4.2 Pitch Group Overlap

Referring to the idea of IoU (Intersection over Union), we design a metric named pitch group overlap:

$$Overlap = \sum_{j=0}^N \frac{len(set(P) \cap set(Q))}{N * len(set(P) \cup set(Q))}, \tag{3}$$

where $set(P)$ means a set contains all pitch group types appearing in the sample, $set(Q)$ means a set contains all types appearing in the real music segment. Overlap means the length of intersection between $set(P)$ and $set(Q)$ divided by the length of the union. Compared to KL divergence, the overlap can measure the difference between P and Q distributions at a coarser granularity. The larger the overlap, the more similar the pitch group types of two songs.

It should be noted that both KL divergence and overlap can only roughly measure the similarity between the two pieces of music. In the music generation task, if the similarity is too high, the diversity will be low. Conversely, if the similarity is too low, the correlation with real music will not be ideal. Therefore, to balance diversity and relevance, it is necessary to find a suitable similarity interval.

6. RESULTS AND ANALYSIS

6.1 Results of Subjective Study

The results shown in Table 1 indicate that our SinTra receives commendable scores, especially in relevance (R). After communicating with the subjects, they reflected that there were several fluent segments in samples, but the occasional messy notes led to the decline of the overall auditory perception. The output space dictionary constructed in the pitch-group representation ensures the correlation

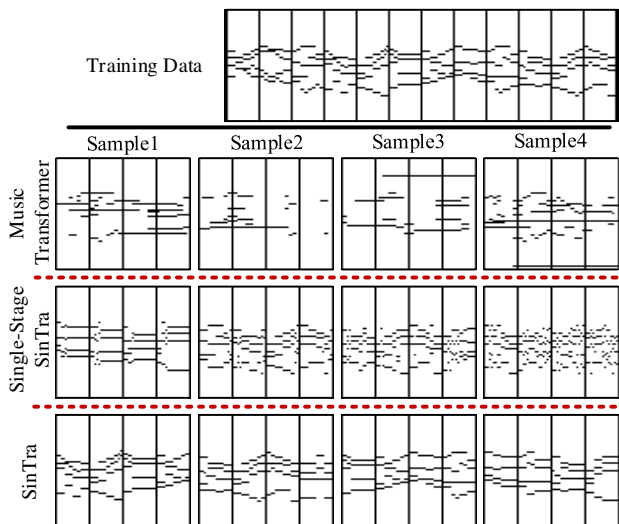


Figure 5. Samples generated by the three models. Each model lists 4 samples. All models are trained under the same training data.

	Music Trans.	Single-stage	SinTra
KL-Div	40.53	27.84	21.67
Overlap	5%	55%	79%

Table 2. Results of objective metrics by Music Transformer, single-stage SinTra and SinTra on JSB Chorale dataset. The metrics of SinTra are marked in bold.

between generated music and real music. Besides, the coarse-to-fine fashion introduced by the pyramid structure enables the model to fully learn the features of training data. Although we introduce randomness at each stage in terms of diversity (D), the results show that the generated music is still close under a single training music.

6.2 Comparison with Previous Work

We compare music generation quality of SinTra with Music Transformer by: 1) we conduct experiments in the same one-shot learning condition, and since Music Transformer only supports single-track music, we use JSB Chorale dataset to evaluate the performance; 2) both models are asked to generate 10 segments in about 1 minute; 3) we set the velocity of all notes in musical pieces generated by models to a reasonable value (100).

The comparison results are shown in Table 2 and the local detail of the generate music is demonstrated in Figure 5. The convergence values of NLL loss shown in Table 3 also validate that Music Transformer does not fit the training music well. The music generated by SinTra is more realistic and fits more closely with real music.

6.3 Method Analysis

6.3.1 Analysis on Pyramid Structure

To verify the effectiveness of the pyramid structure, we build the single-stage SinTra. As depicted in Figure 5,

	Music Trans.	Single-stage	SinTra
NLL	~ 1.0	$10^{-3} \sim 10^{-4}$	$10^{-3} \sim 10^{-4}$

Table 3. Results of NLL loss by Music Transformer, single-stage SinTra and SinTra on JSB Chorale dataset when model converges. Both variants of SinTra can converge in $10^{-3} \sim 10^{-4}$ while Music Transformer can't.

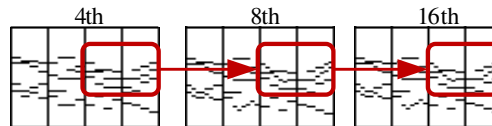


Figure 6. The effect of detail enhancement at each stage of SinTra. In the stage of the 4th note-value, simple notes will be generated but lack details, while in the 8th and 16th stages, the model refines some details.

we can see that the embryonic form of melody appears in single-stage SinTra, but is overly-fragmented and noisy. The KL divergence (27.84) and overlap (55%) shown in Table 2 are not ideal because of overly-fragmented notes. The comparison results show that SinTra can effectively suppress messy notes.

6.3.2 Analysis on Multi-stage Output

Figure 6 shows the output effect of SinTra at each stage. In the 1st stage (the 4th note-value), the model generates simple notes, ignores local details, and sketches roughly the outline of songs. Then the 2nd and 3rd stages enrich the contour in turn, bringing local detail changes. This model structure can effectively solve the problem of music generated by single-stage SinTra, while ensuring the generation quality, which can introduce randomness in each stage.

7. CONCLUSION AND FUTURE WORK

In this work, we propose SinTra, an inspiration generation framework to complete the task of one-shot learning in music generation. SinTra, consisting of a pyramid of Transformer-XL with a multi-scale training strategy, can learn both the musical structure and the relative positional relationship between notes of the single training music segment. Moreover, we present a novel pitch-group representation to ensure the relevance of generated samples and training music. The results of subjective study and objective evaluation show the effectiveness of SinTra for learning from single training data, generating music samples with a strong correlation with the training music. However, there is still room for improvement in the quality and diversity of generated music.

In the future, we will study controllable music generation that can integrate emotion- and style-controlled generations into SinTra. We will also consider large-scale generative pre-training to improve generation quality. We hope SinTra can be leveraged to enhance musicians' productivity and inspire them to compose higher quality music.

8. ACKNOWLEDGEMENTS

This paper was supported by the National Natural Science Foundation of China under Grant Number 61771440.

9. REFERENCES

- [1] T. R. Shaham, T. Dekel, and T. Michaeli, “SinGAN: Learning a generative model from a single natural image,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, Korea, 2019, pp. 4570–4580.
- [2] M. Awiszus, F. Schubert, and B. Rosenhahn, “TOADGAN: Coherent style level generation from a single example,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Online, 2020, pp. 10–16.
- [3] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, “LakhNES: Improving multi-instrumental music generation with cross-domain pre-training,” in *Proceedings of the 20th Conference of the International Society for Music Information Retrieval*, Delft, Netherlands, 2019.
- [4] Y.-S. Huang and Y.-H. Yang, “Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions,” in *Proceedings of the 28th ACM International Conference on Multimedia*, Seattle, United States, 2020, pp. 1180–1188.
- [5] X. Wu, C. Wang, and Q. Lei, “Transformer-XL based music generation with multiple sequences of time-valued notes,” *arXiv preprint arXiv:2007.07244*, 2020.
- [6] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, “Transformer-XL: Attentive language models beyond a fixed-length context,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 2019, pp. 2978–2988.
- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, Long Beach, United States, 2017, pp. 5998–6008.
- [8] I. Simon, D. Morris, and S. Basu, “MySong: Automatic accompaniment generation for vocal melodies,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008, pp. 725–734.
- [9] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii, “Function-and rhythm-aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences,” in *Proceedings of the 18th Conference of the International Society for Music Information Retrieval*, Suzhou, China, 2017, pp. 502–508.
- [10] C.-H. Chuan and E. Chew, “A hybrid system for automatic generation of style-specific accompaniment,” in *Proceedings of the 4th International Joint Workshop on Computational Creativity*, London, United Kingdom, 2007, pp. 57–64.
- [11] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent neural network based language model,” in *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, Chiba, Japan, 2010.
- [12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [13] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN Encoder–Decoder for statistical machine translation,” in *Proceedings of the 19th Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014, pp. 1724–1734.
- [14] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *Proceedings of the 2nd International Conference on Learning Representations*, Banff, Canada, 2014.
- [15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” in *Proceedings of the 28th Annual Conference on Neural Information Processing Systems*, Montreal, Canada, 2014.
- [16] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, New Orleans, United States, 2018.
- [17] H.-W. Dong and Y.-H. Yang, “Convolutional generative adversarial networks with binary neurons for polyphonic music generation,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018.
- [18] X. Liang, J. Wu, and J. Cao, “MIDI-Sandwich2: RNN-based hierarchical multi-modal fusion generation VAE networks for multi-track symbolic music generation,” *arXiv preprint arXiv:1909.03522*, 2019.
- [19] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, “XiaoIce band: A melody and arrangement generation framework for pop music,” in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, London, United Kingdom, 2018, pp. 2837–2846.

- [20] H. H. Mao, T. Shin, and G. Cottrell, "DeepJ: Style-specific music generation," in *Proceedings of the 12th IEEE International Conference on Semantic Computing*, California, United States, 2018, pp. 377–382.
- [21] D. D. Johnson, "Generating polyphonic music using tied parallel networks," in *Proceedings of the 6th International conference on Evolutionary and Biologically Inspired Music and Art*, Amsterdam, Netherlands, 2017, pp. 128–143.
- [22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [23] N. Li, S. Liu, Y. Liu, S. Zhao, M. Liu, and M. Zhou, "Neural speech synthesis with Transformer network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Hawaii, United States, 2019, pp. 6706–6713.
- [24] A. Mohamed, D. Okhonko, and L. Zettlemoyer, "Transformers with convolutional context for ASR," *arXiv preprint arXiv:1904.11660*, 2019.
- [25] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating music with long-term structure," in *Proceedings of the 19th Conference of International Society for Music Information Retrieval*, Paris, France, 2018.
- [26] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the Human Language Technology Conference of the NAACL*, New Orleans, United States, 2018.
- [27] C. Payne, "MuseNet," *OpenAI Blog*, 2019. [Online]. Available: <https://openai.com/blog/musenet>.
- [28] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019.
- [29] T. Gale, M. Zaharia, C. Young, and E. Elsen, "Sparse GPU kernels for deep learning," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Atlanta, United States, 2020, pp. 1–14.
- [30] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proceedings of the International Conference on Learning Representations*, New Orleans, United States, 2019.
- [31] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: application to polyphonic music generation and transcription," in *Proceedings of the 29th International Conference on International Conference on Machine Learning*, Wisconsin, United States, 2012, pp. 1881–1888.
- [32] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," *arXiv preprint arXiv:1611.09904*, 2016.

CONTRASTIVE LEARNING OF MUSICAL REPRESENTATIONS

Janne Spijkervet

John Ashley Burgoyne

Institute for Logic, Language, and Computation, University of Amsterdam

janne.spijkervet@gmail.com, j.a.burgoyne@uva.nl

ABSTRACT

While deep learning has enabled great advances in many areas of music, labeled music datasets remain especially hard, expensive, and time-consuming to create. In this work, we introduce SimCLR to the music domain and contribute a large chain of audio data augmentations to form a simple framework for self-supervised, contrastive learning of musical representations: CLMR. This approach works on raw time-domain music data and requires no labels to learn useful representations. We evaluate CLMR in the downstream task of music classification on the MagnaTagATune and Million Song datasets and present an ablation study to test which of our music-related innovations over SimCLR are most effective. A linear classifier trained on the proposed representations achieves a higher average precision than supervised models on the MagnaTagATune dataset, and performs comparably on the Million Song dataset. Moreover, we show that CLMR’s representations are transferable using out-of-domain datasets, indicating that our method has strong generalisability in music classification. Lastly, we show that the proposed method allows data-efficient learning on smaller labeled datasets: we achieve an average precision of 33.1% despite using only 259 labeled songs in the MagnaTagATune dataset (1% of the full dataset) during linear evaluation. To foster reproducibility and future research on self-supervised learning in music, we publicly release the pre-trained models and the source code of all experiments of this paper.

1. INTRODUCTION

Supervised learning methods have been widely used in musical tasks like chord recognition [1, 2], key detection [3], beat tracking [4], music audio tagging [5] and music recommendation [6]. These methods require labeled corpora, which are difficult, expensive and time-consuming to create for music in particular [7], while raw unlabeled music data is available in vast quantities. Unsupervised alternatives to end-to-end deep learning for music are compelling, especially if they can generalise to smaller datasets.

Despite the importance of unsupervised learning for raw audio signals, unsupervised learning for musical tasks

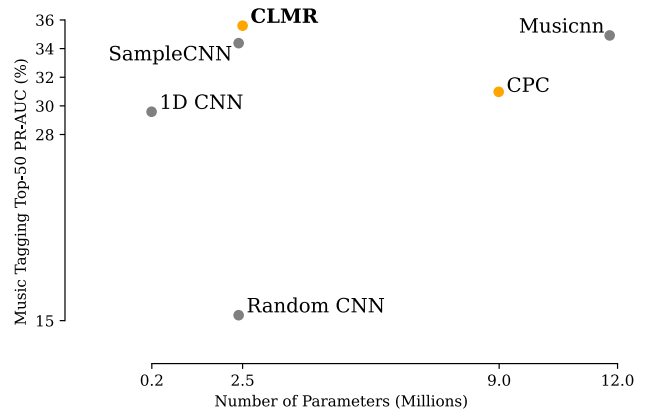


Figure 1: Performance and model complexity comparison of supervised models (grey) and self-supervised models (ours) in music classification of raw audio waveforms on the MagnaTagATune dataset to evaluate musical representations. Supervised models were trained end-to-end, while CLMR and CPC are pre-trained without ground truth: their scores are obtained by training a *linear* classifier on their learned representations but nonetheless perform competitively to the supervised models.

has yet to see breakthroughs comparable to those in supervised learning. There have been successes with methods like PCA, PMSC’s and spherical k -means that rely on a transformation pipeline [8, 9], and very recently with self-supervised methods in the time-frequency domain for general audio classification tasks [10–13], but learning effective representations of raw audio in an unsupervised manner has remained elusive for musical tasks.

Self-supervised representation learning is an unsupervised learning paradigm that has demonstrated advances across many tasks and research domains [14–18]. This includes the ability to use substantially less labeled data when fine-tuning on a specific task [17, 19, 20]. Without ground truth, there can be no ordinary loss function for training; self-supervised learning trains by way of a proxy loss function instead. One way to preserve the amount of useful information during self-supervised learning is to define the proxy loss function with respect to a relatively simple pretext task, with the idea that a representation that is good for the pretext task will also be useful for downstream tasks. Many approaches rely on heuristics to design pretext tasks [21, 22], e.g., by withholding a pitch transformation [23]. Alternatively, *contrastive representation learning* formulates the proxy loss directly on the learned



representations and relies on contrasting multiple, slightly differing versions of any one example by often using negative sampling strategies [17,24,25] or by bootstrapping the representations [18].

In this paper, we combine the insights of a simple contrastive learning framework for images, SimCLR [17], with recent advances in representation learning for audio in the time domain [26]. We also contribute a pipeline of data augmentations on musical audio, to form a simple framework for self-supervised, contrastive learning of representations of raw waveforms of music. To compare the effectiveness of this simple framework compared to a more complex self-supervised learning objective, we also evaluate representations learned by contrastive predictive coding (CPC) [15]. The self-supervised models are evaluated on the downstream music tagging task, enabling us to evaluate their versatility: music tags describe many characteristics of music, e.g., genre, instrumentation and dynamics. Our key contributions are the following.

- CLMR achieves strong performance on the music classification task compared to supervised models, despite self-supervised pre-training and training a linear classifier on the downstream task with raw signals of musical audio (see Figure 1).
- CLMR enables efficient classification: we achieve comparable performance using as few as 1% of the labeled data.
- We show the out-of-domain transferability of representations learned from pre-training CLMR on entirely different corpora of musical audio.
- CLMR can learn from *any* dataset of raw music audio, requiring neither transformations nor fine-tuning on the input data; nor do the models require manually annotated labels for pre-training.
- We provide an ablation study on the effectiveness of individual audio data augmentations.

2. RELATED WORK

The goal of representation learning is to identify features that make prediction tasks easier and more robust to the complex variations of natural data [27]. In unsupervised representation learning, generative modeling and likelihood-based models typically find useful representations of the data by attempting to reconstruct the observations on the basis of their learned representations [28, 29]. *Self-supervised* representation learning aims to identify the explanatory factors of the data using an objective that is formulated with respect to the learned representations directly [15, 18, 19, 21, 22].

Compared to vision, work on self-supervised learning in audio is still very limited, but there are a number of works that appeared very recently. Contrastive predictive coding is a universal approach to contrastive learning, and has been successful for speaker and phoneme classification using raw audio, among other tasks [15]. PASE [30] introduces several self-supervised workers that solve regression

or binary discrimination tasks, that jointly optimise an encoder for speech recognition. To improve the representations for mismatched acoustic conditions and their transferability, they apply augmentations to the input speech signal [31]. In music information retrieval, recent advances have been made in self-supervised pitch estimation [23], closely matching supervised, state-of-the-art baselines [32] despite being trained without ground truth labels. L^3 -Net learns deep embeddings from audio-visual correspondence in videos by way of self-supervised learning [10]. Their work uses mel-spectrograms for audio and requires more than 40 million audio-video training samples to learn optimal embeddings. Audio2Vec also operates in the time-frequency domain and learns by reconstructing spectrogram slices from past and future slices [11]. With limited data, Audio2Vec outperforms supervised models in pitch and instrument classification. CLAR also uses a contrastive learning objective, and computes a loss on a concatenation of representations learned from both raw audio and mel-spectrograms [12]. COLA uses a similar method with mel-spectrograms only, and uses bilinear comparisons instead of cosine similarity [13]. Both works are evaluated on speech command, environmental sound classification, and on pitch and instrument classification on the NSynth dataset [33].

3. METHOD

This work builds on SimCLR, a simple contrastive learning framework of visual representations [17]. Despite a task-agnostic, labelless discriminative pre-training approach, a linear classifier achieved performance comparable to fully supervised models in many image classification benchmarks. Its learning objective is to maximise the agreement of latent representations of augmented views of the same image using a contrastive loss. In Section 2, we will continue an overview of contrastive learning.

In CLMR, we adapt this framework to the domain of raw music audio. While most core components of CLMR have appeared in previous work, its ability to model waveforms of music cannot be explained by a single design choice, but by their composition. We will first elaborate the four core components in the following subsections:

- A stochastic composition of data augmentations that produces two correlated, augmented examples of the same audio fragment, the ‘positive pair’, denoted as x_i and x_j .
- An encoder neural network $g_{enc}(\cdot)$ that maps the augmented examples to their latent representations.
- A projector neural network $g_{proj}(\cdot)$ that maps the encoded representations to the latent space where the contrastive loss is formulated.
- A contrastive loss function, which aims to identify x_j from the negative examples in the batch $\{x_{k \neq i}\}$ for a given x_i .

The complete framework is visualised in Figure 2.

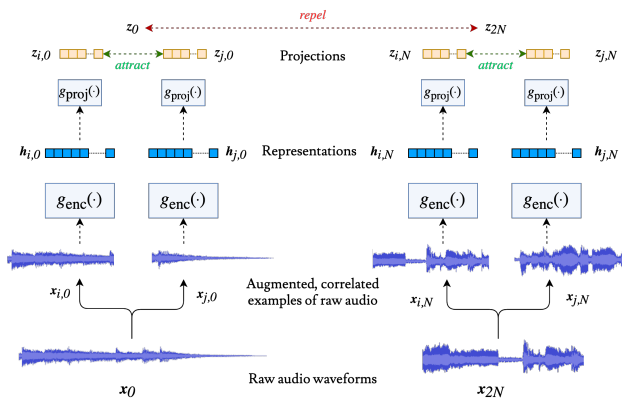


Figure 2: The complete framework operating on raw audio, in which the contrastive learning objective is directly formulated in the latent space of correlated, augmented examples of pairs of raw audio waveforms of music.

3.1 Data Augmentations

We designed a comprehensive chain of audio augmentations for raw audio waveforms of music to make it harder for the model to identify the correct pair of examples. For details, see Appendix B¹. Each consecutive augmentation is stochastically applied on x_i and x_j independently, i.e., each augmentation has an independent probability $p_{\text{transform}}$ of being applied to the audio. The order of augmentations applied to audio is carefully considered, e.g., applying a delay effect *after* reverberation empirically gives an entirely different result in music.

1. A random fragment of size s is selected from a piece of music, without trimming silence (e.g., the intro or outro of a song). The two examples x_i and x_j from the same audio fragment can overlap or be very disjoint, allowing the model to infer both local and global structures.
2. The polarity of the audio signal is inverted, i.e., the amplitude is multiplied by -1 .
3. Additive white Gaussian noise is added with a signal-to-noise ratio of 80 decibels to the original signal.
4. The gain is reduced between $[-6, 0]$ decibels.
5. A frequency filter is applied to the signal. A coin flip determines whether it is a low-pass or a high-pass filter. The cut-off frequencies are drawn from uniform distributions on $[2200, 4000]$ or $[200, 1200]$ Hz respectively.
6. The signal is delayed and added to the original signal with a volume factor of 0.5. The delay is randomly sampled between 200-500ms, in 50ms increments.
7. The signal is pitch shifted. The pitch transposition interval is drawn from a uniform distribution of semitones between $[-5, 5]$, i.e., a perfect fourth compared to the original signal’s scale.
8. Reverb is added to alter the signal’s acoustics. The impulse response’s room size, reverbation and damping factor is drawn from a uniform distribution on $[0, 100]$.

¹ The supplementary material can be found at the accompanying webpage of this paper: <https://spijkervet.github.io/CLMR>

The space of augmentations is not limited to these operations and could be extended to, e.g., randomly applying chorus, distortion and other modulations. Some of these have been shown to improve performance in self-supervised learning for automatic speech recognition in the time-domain as well [31, 34].

3.2 Batch Composition

A larger batch size N makes the contrastive learning objective harder – there are simply more negative examples the anchor sample needs to identify the positive sample from – but it can substantially improve model performance [17]. We sample one song from the batch, augment it into two examples, and treat them as the positive pair. We treated the remaining $2(N - 1)$ examples in the batch as negative examples, and did not sample the negative examples explicitly. Larger batch sizes introduces a practical problem for raw audio when training on a GPU, as their input dimensionality increases for higher sample rates. When training on multiple GPU’s, we used global batch normalisation, i.e., we aggregate the batch statistics over all devices during parallel training, to avoid potential leakage of batch statistics because the positive examples are sampled on the same device (which improves training loss, but counteracts learning of useful representations).

3.3 Encoder

To directly compare a state-of-the-art end-to-end supervised model used in music classification on raw waveforms against a self-supervised model, we use the SampleCNN architecture as our encoder [26]. Similarly, we use a fixed audio input of 59 049 samples with a sample rate of 22 050 Hz. In this configuration, the SampleCNN encoder g_{enc} consists of 9 one-dimensional convolution blocks, each with a filter size of 3, batch normalisation, ReLU activation and max pooling with pool size 3. The final output layer is removed, which yields a 512-dimensional feature vector h_i for every audio input. The feature vectors from the encoder can be directly used in the learning objective, but formulating the objective on encodings mapped to a different latent space by a parameterised function helps the effectiveness of the representations [17]. In our experiments, we use a non-linear layer $z_i = W^{(2)} \text{ReLU}(W^{(1)} h_i)$ with an output dimensionality of 128 as the projection head g_{proj} . There are 2.5 million trainable parameters in total, which is put in comparison with other state-of-the-art models in Figure 1.

We used 96 examples per batch and the afore-described encoder configuration to directly compare our self-supervised performance with the equally expressive fully supervised method [26]. We ran experiments with batch sizes of 96 on $2 \times$ NVIDIA 1080Ti, while for larger batches up to $4 \times$ Titan RTX’s were used. With 2 1080Ti’s, it takes ~ 5 days to train 1 000 epochs on our largest dataset.

3.4 Contrastive Loss Function

In keeping with recent findings on several objective functions in contrastive learning [17], the contrastive loss function used in this model is normalised temperature-scaled cross-entropy loss, commonly denoted as *NT-Xent loss*:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j) / \tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(z_i, z_k) / \tau)} \quad (1)$$

The pairwise similarity is measured using cosine similarity and the temperature parameter τ helps the model learn from hard negatives. The indicator function $\mathbb{1}_{[k \neq i]}$ evaluates to 1 iff $k \neq i$. This loss is computed for all pairs, both (z_i, z_j) and (z_j, z_i) , for $i \neq j$.

3.5 Contrastive Predictive Coding

We adjusted the original CPC encoder g_{enc} [15] to a deeper architecture for more direct comparison [26]. The encoder g_{enc} consists of 7 layers with 512 filters each, and filter sizes [10, 6, 4, 4, 4, 2, 2] and strides [5, 3, 2, 2, 2, 2, 2]. Instead of relying on max-pooling, the filter sizes and strides are adjusted to parameterise and facilitate downsampling. We also increased the number of prediction steps to 20, effectively asking the network to predict 100 ms of audio into the future. The batch size is set to 64 from which 15 negative examples in the contrastive loss are drawn.

3.6 Linear Evaluation

The evaluation of representations learned by self-supervised models is commonly done with linear evaluation [15–17], which measures how linearly separable the relevant classes are under the learned representations. We obtain the representations for all datapoints from a frozen CLMR network after pre-training has converged, and train a linear classifier using these self-supervised representations on the downstream task of music classification. For CPC, the representations are extracted from the autoregressor, yielding a context vector of size (20, 256), which is global-average pooled to obtain a single vector of 512 dimensions. For CLMR, the last 512-dimensional vector h from the encoder is used instead of z from the projection head because that yielded consistently better results for all our experiments. We compute the evaluation metrics on a held-out test set, averaged over three runs on the training set using different random seeds.

3.7 Optimisers

We use the Adam optimiser [35] with a learning rate of 0.0003 and $\beta_1 = 0.9$ and $\beta_2 = 0.999$ during pre-training and employ He initialisation for all convolutional layers. The temperature parameter τ is set to 0.5, since we observed consistent results regardless of varying batch sizes and temperature $\tau \in \{0.1, 0.5, 1.0\}$. For linear evaluation, we use the Adam optimiser with a learning rate of 0.0003 and a weight decay of 10^{-6} . Backpropagation is only done in the final (linear) head for all experiments in this paper. We also employ an early stopping mechanism when the validation scores do not improve for 5 epochs.

Model	Dataset	ROC-AUC	PR-AUC
CLMR (ours)	MTAT	88.7 (89.3)	35.6 (36.0)
Musicnn [5] [†]	MTAT	89.0	34.9
SampleCNN [26] [†]	MTAT	88.6	34.4
CPC (ours)	MTAT	86.6 (88.0)	31.0 (33.0)
1D CNN [36] [†]	MTAT	85.6	29.6
Transformer [37] ^{†§}	MSD	89.7	34.8
Musicnn [5] [†]	MSD	88.0	28.7
SampleCNN [26] [†]	MSD	87.9	28.5
CLMR (ours)	MSD	85.7	25.0

Table 1: Tag prediction performance on the MagnaTagATune (MTAT) dataset and Million Song Dataset (MSD), compared with fully supervised models^(†) trained on raw audio waveforms. We omit most works that operate on (mel-) spectrograms^(§) to make a fair comparison with our approach on raw audio. For reference, we add the Transformer model that is the current state-of-the-art in music tagging. For the self-supervised models, the scores are obtained by training a *linear*, logistic regression classifier using the frozen representations from self-supervised pre-training. Scores in brackets show performance when adding a hidden layer to the linear classifier.

4. EXPERIMENTAL RESULTS

4.1 Datasets

We evaluated the quality of our representations with music classification experiments. Predicting the top 50 semantic tags in the MagnaTagATune and Million Song datasets [38, 39] is a popular benchmark for music classification. These semantic tags are annotated by human listeners, and have a varying degree of abstraction and describe many facets of music, including genre, instrumentation and dynamics. It is a multi-label classification task: each track can have multiple tags, of which we use the 50 most frequently occurring to compare our performance against supervised benchmarks.

The MagnaTagATune dataset consists of 25k music clips from 6622 unique songs, of which we use about 187k fragments of 2.6 seconds for training, and the same train/test split as previous work [5,9,26]. The Million Song Dataset contains a million songs, of which about 240k previews of 30 seconds are available and labeled with Last.FM tag annotations. We only use the train, validation and test split of 201 680 / 11 774 / 28 435 songs as used in previous work [5, 26], not all million songs during self-supervised pre-training. This results in 2.2 million music fragments of 2.6 seconds for training, i.e., almost 1 600 hours of music. The tags for the Million Song Dataset also contain overlapping genre and semantic tags, e.g., ‘beautiful’, ‘happy’ and ‘sad’, which are arguably harder to separate during the linear evaluation phase.

We use average tag-wise area under the receiver operating characteristic curve (ROC-AUC) and average precision (PR-AUC) scores as evaluation metrics. They are measured globally for the whole dataset, i.e., for the tag metric we measure the retrieval performance on the tag dimension (column-wise) and for the clip metric we measure the

performance on the clip dimension (row-wise). PR-AUC is calculated in addition to ROC-AUC, because ROC-AUC scores can be over-optimistic for imbalanced datasets like MagnaTagATune [40].

4.2 Quantitative Evaluation

The most important goal set out in this paper is to evaluate the difference in performance between an otherwise identical, fully supervised network when learning representations using a self-supervised objective.

CLMR exceeds the supervised benchmark for the MagnaTagATune dataset with a PR-AUC of 35.6%, despite task-agnostic, self-supervised pre-training and a linear classifier for training, as shown in Table 1. An additional 0.4% PR-AUC performance gain is added by adding an extra hidden layer to the classifier. When increasing the batch size and the number of parameters, we observe another performance gain to 37.0% PR-AUC as show in Appendix C.1. The performance on the larger Million Song Dataset is lower compared to the supervised benchmark, and especially to the current state-of-the-art model that is trained using mel-spectrograms [37], but is still remarkable given the use of a linear classifier. The tags in the Million Song Dataset are semantically more complex, e.g., ‘catchy’, ‘sexy’, ‘happy’, and have more similar genre tags, e.g., ‘progressive rock’, ‘classic rock’ and ‘indie rock’, which our proposed contrastive learning method may not distinguish.

CPC also shows competitive performance with fully supervised models in the music classification task. Despite CPC’s good performance, self-supervised training does not require a memory bank or more complex loss functions, e.g., those incorporating mutual information or more explicit negative sampling strategies, to learn useful representations.

We also analyse the quality of our representations, showing they can cleanly separate audio fragments from different classes, and visualise the convolution filters of the self-supervised models in Appendix C.4.

4.3 Data Augmentations

The CLMR model relies on a pipeline of strong data augmentations to facilitate the learning of representations that are more robust and allow for better generalisation in the downstream task. In Table 2, we show the linear evaluation scores obtained by taking a random crop of audio and performing one additional, individual augmentation. While all datasets contain songs of variable length, we always sample a random crop of audio of the same size before applying other augmentations. This makes it harder to assess the individual contribution of each augmentation to the downstream task performance. We therefore consider an asymmetric data transformation setting: we only apply the augmentation(s) to one branch of the framework, while we settle with an identity function for the other branch (i.e., $t(x_j) = x_j$) [17]. The model is pre-trained from scratch for 1 000 epochs after which linear evaluation is performed.

Transform	Tag		Clip	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
Filter	87.6	33.3	92.5	67.9
Reverb	86.5	31.7	91.8	65.8
Polarity	86.3	31.5	91.7	65.7
Noise	86.1	31.5	91.5	65.5
Pitch	86.4	31.5	91.5	65.3
Gain	86.2	31.1	91.5	65.1
Delay	85.8	30.5	91.3	64.9
Crop	85.8	30.5	91.3	64.8

Table 2: CLMR music tagging performance using a random crop together with one other audio data augmentation.

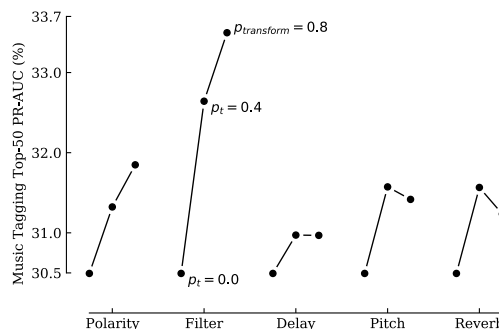


Figure 3: PR - AUC_{TAG} scores for transformations under different, consecutive probabilities $p \in \{0.0, 0.4, 0.8\}$

When only taking a random crop of audio, we achieve a PR-AUC score of 30.5. Most individual augmentations show an increase in performance, while adding gain or delay does not impact performance as much. Adding a filter to the augmentation pipeline increases the downstream performance more significantly.

Besides evaluating the individual contribution of each augmentation with augmentation probability $p_t = 1$, we also vary $p_t \in \{0, 0.4, 0.8\}$. This is done to assess the optimal amount of augmentation to each example, i.e., the contrastive learning task should neither be too hard, nor too simple, for learning effective representations for the music classification task. The linear evaluation PR-AUC score is shown for each augmentation under a different probability p_t in Figure 3. For the Polarity and Filter transformations, performing them more often with a probability of $p_t = 0.8$ is beneficial. For the Delay, Pitch and Reverb transformations, a transformation probability of $p_t = 0.4$ works better than performing them more aggressively. Generally, we find that strong data augmentations result in more robust representations and better downstream task performance.

4.4 Data Efficient Classification Experiments

To test the efficient classification capability of the CLMR model, we train the linear classifier on consecutive, class-balanced subsets of the labels in the train dataset and report its performance. During the task-agnostic, self-supervised pre-training phase, 100% of the data is used. Figures 4 and 5 show the PR-AUC scores obtained when increasing

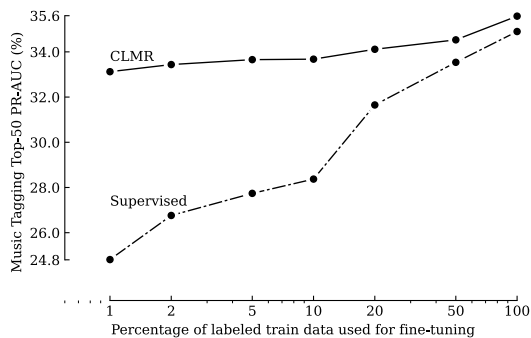


Figure 4: Percentage of labels used for training vs. the achieved PR – AUC_{TAG} score on the MTAT dataset.

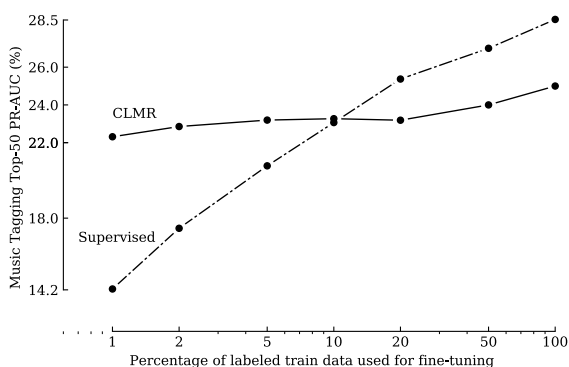


Figure 5: Percentage of labels used for training vs. the achieved PR – AUC_{TAG} score on the MSD.

the amount of labels available during training. For both datasets, self-supervised pre-training greatly improves performance when less labeled data is available. Using 100 times fewer labeled songs, i.e., only 259 songs, CLMR scores 33.1% PR-AUC compared to 24.8% PR-AUC obtained with an equivalent, end-to-end trained supervised model trained on about 25 000 songs. Pre-training using a self-supervised objective without labels therefore substantially improves efficient classification: only 1% of the labels are required while maintaining a similar performance. For the Million Song Dataset, a fully supervised model exceeds CLMR at 10% of the labels, which are 24 190 unique songs in total.

4.5 Transfer Learning Experiments

To test the out-of-domain generalisability of the learned representations, we pre-trained CLMR on entirely different music datasets. After pre-training, we freeze the weights of the network, i.e., we do not fine-tune the encoder, and subsequently perform the linear evaluation procedure outlined in Section 3.6. While originally made for chord recognition, we use 461 contemporary pop songs recorded between the 1940’s and 2000’s from the McGill Billboard dataset [41]. The Free Music Archive dataset [42] consists of 22 413 songs for the ‘medium’ version, and the fault-filtered GTZAN dataset [43, 44] contains 930 fragments of

Model	Train Dataset	ROC-AUC _{TAG}	PR-AUC _{TAG}
CLMR	MSD	87.8	33.1
CPC	FMA	86.3 (87.8)	30.7 (32.5)
CLMR	FMA	86.2 (86.6)	30.6 (31.2)
CPC	Billboard	85.8 (86.3)	29.7 (30.2)
CPC	GTZAN	83.4 (86.0)	26.9 (29.7)
CLMR	Billboard	82.7 (84.2)	26.9 (27.8)
CLMR	GTZAN	81.9 (85.4)	26.2 (29.5)

Table 3: Transfer learning experiments for CLMR and CPC, which are trained on a separate dataset and evaluated on the MagnaTagATune dataset. The reported scores are obtained with a frozen, pre-trained encoder and a linear classifier. Scores in parenthesis are obtained when adding one extra hidden layer to the classifier.

30 seconds, both popular for music classification.

The results of the transfer learning experiments are shown in Table 3. Both CPC and CLMR show the ability to learn effective representations from out-of-domain datasets without ground truth, and even exceed accuracy scores of previous, supervised end-to-end systems on raw audio [36]. Moreover, both models even demonstrate the ability to learn useful representations on the much smaller GTZAN and Billboard datasets. The CLMR model performs better when it is pre-trained on larger datasets, which is expected as it heavily relies on the number of unique, independent examples that make the contrastive learning task harder, resulting in more robust representations. When pre-training on smaller datasets, CPC can find more useful representations, especially when adding an extra hidden layer to the fine-tune head.

5. CONCLUSION

In this paper, we presented CLMR, a self-supervised contrastive learning framework that learns useful representations of raw waveforms of musical audio. The framework requires no preprocessing of the input audio and is trained without ground truth, which enables simple and straightforward pre-training on music datasets of unprecedented scale. We tested the learned, task-agnostic representations by training a linear classifier on the music classification task on the MagnaTagATune and Million Song datasets, achieving competitive performance compared to fully supervised models. We also showed that CLMR can achieve comparable performance using 100 times fewer labeled songs, and demonstrated the out-of-domain transferability of representations learned from pre-training on entirely different datasets of music. To foster reproducibility and future research on self-supervised learning in music information retrieval, we publicly release the pre-trained models and the source code of all experiments of this paper². The simplicity of training the model without any labels and without preprocessing the audio, together with encouraging results obtained with a single linear layer optimised for a challenging music task, are exciting developments towards unsupervised learning on raw musical audio.

² <https://github.com/spijkervet/clmr>

6. ACKNOWLEDGEMENTS

We would like to thank Jordan B.L. Smith, Wilker Aziz and Keunwoo Choi for their feedback on the draft. We would also like to extend our gratitude to the University of Amsterdam and SURFsara for giving us access to their Research Capacity Computing Services GPU cluster.

7. REFERENCES

- [1] F. Korzeniowski and G. Widmer, “A Fully Convolutional Deep Auditory Model for Musical Chord Recognition,” *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2016. [Online]. Available: <http://arxiv.org/abs/1612.05082>
- [2] T.-P. Chen and L. Su, “Harmony Transformer: Incorporating Chord Segmentation Into Harmony Recognition,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019.
- [3] F. Korzeniowski and G. Widmer, “End-to-End Musical Key Estimation Using a Convolutional Neural Network,” in *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, 2017. [Online]. Available: <http://arxiv.org/abs/1706.02921>
- [4] S. Böck, F. Krebs, and G. Widmer, “Joint Beat and Downbeat Tracking with Recurrent Neural Networks,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR*, 2016.
- [5] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, “End-to-End Learning for Music Audio Tagging at Scale,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 2017. [Online]. Available: <http://arxiv.org/abs/1711.02520>
- [6] A. van den Oord, S. Dieleman, and B. Schrauwen, “Deep content-based music recommendation,” in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013, pp. 2643–2651. [Online]. Available: <https://proceedings.neurips.cc/paper/2013/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf>
- [7] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, “Annotator subjectivity in harmony annotations of popular music,” *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019. [Online]. Available: <https://doi.org/10.1080/09298215.2019.1613436>
- [8] P. Hamel, S. Lemieux, Y. Bengio, and D. Eck, “Temporal Pooling and Multiscale Learning for Automatic Annotation and Ranking of Music Audio,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, 2011, pp. 729–734.
- [9] S. Dieleman and B. Schrauwen, “Multiscale Approaches to Music Audio Feature Learning,” in *Proceedings of the 14th International Society for Music Information Retrieval conference*, 2013, pp. 116–121.
- [10] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, Listen, and Learn More: Design Choices for Deep Audio Embeddings,” in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3852–3856.
- [11] M. Tagliasacchi, B. Gfeller, F. d. C. Quitry, and D. Roblek, “Pre-Training Audio Representations With Self-Supervision,” *IEEE Signal Processing Letters*, vol. 27, pp. 600–604, 2020.
- [12] H. Al-Tahan and Y. Mohsenzadeh, “CLAR: Contrastive Learning of Auditory Representations,” in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 13–15 Apr 2021, pp. 2530–2538. [Online]. Available: <http://proceedings.mlr.press/v130/al-tahan21a.html>
- [13] A. Saeed, D. Grangier, and N. Zeghidour, “Contrastive Learning of General-Purpose Audio Representations,” in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 3875–3879.
- [14] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, “Discriminative Unsupervised Feature Learning with Exemplar Convolutional Neural Networks,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 9, pp. 1734–1747, 2015.
- [15] A. van den Oord, Y. Li, and O. Vinyals, “Representation Learning with Contrastive Predictive Coding,” *arXiv:1807.03748 [cs, stat]*, 2019. [Online]. Available: <http://arxiv.org/abs/1807.03748>
- [16] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, “Learning deep representations by mutual information estimation and maximization,” *arXiv:1808.06670 [cs, stat]*, 2019. [Online]. Available: <http://arxiv.org/abs/1808.06670>
- [17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations,” *arXiv:2002.05709 [cs, stat]*, 2020, arXiv: 2002.05709. [Online]. Available: <http://arxiv.org/abs/2002.05709>
- [18] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar *et al.*, “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning,” *arXiv preprint arXiv:2006.07733*, 2020.

- [19] O. J. Hénaff, A. Razavi, C. Doersch, S. A. Eslami, and A. v. d. Oord, “Data-Efficient Image Recognition with Contrastive Predictive Coding,” *arXiv preprint arXiv:1905.09272*, 2019.
- [20] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, “Big Self-Supervised Models are Strong Semi-Supervised Learners,” *arXiv preprint arXiv:2006.10029*, 2020.
- [21] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised Visual Representation Learning by Context Prediction,” in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1422–1430. [Online]. Available: <http://ieeexplore.ieee.org/document/7410524/>
- [22] R. Zhang, P. Isola, and A. A. Efros, “Colorful Image Colorization,” in *European conference on computer vision*. Springer, 2016, pp. 649–666.
- [23] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, “Pitch Estimation Via Self-Supervision,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3527–3531.
- [24] Y. Tian, D. Krishnan, and P. Isola, “Contrastive Multi-view Coding,” *arXiv preprint arXiv:1906.05849*, 2019.
- [25] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum Contrast for Unsupervised Visual Representation Learning,” *arXiv preprint arXiv:1911.05722*, 2019.
- [26] J. Lee, J. Park, K. L. Kim, and J. Nam, “SampleCNN: End-to-End Deep Convolutional Neural Networks Using Very Small Filters for Music Classification,” *Applied Sciences*, vol. 8, no. 1, p. 150, 2018.
- [27] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [29] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, Conference Track Proceedings*, 2016. [Online]. Available: <http://arxiv.org/abs/1511.06434>
- [30] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio, “Learning Problem-Agnostic Speech Representations from Multiple Self-Supervised Tasks,” in *Proc. Interspeech 2019*, 2019, pp. 161–165. [Online]. Available: <http://dx.doi.org/10.21437/Interspeech.2019-2605>
- [31] M. Ravanelli, J. Zhong, S. Pascual, P. Swietojanski, J. Monteiro, J. Trmal, and Y. Bengio, “Multi-Task Self-Supervised Learning for Robust Speech Recognition,” *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6989–6993, 2020.
- [32] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, “Crepe: A Convolutional Representation for Pitch Estimation,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 161–165.
- [33] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, “Neural audio synthesis of musical notes with wavenet autoencoders,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML’17. JMLR.org, 2017, p. 1068–1077.
- [34] E. Kharitonov, M. Rivière, G. Synnaeve, L. Wolf, P.-E. Mazaré, M. Douze, and E. Dupoux, “Data Augmenting Contrastive Learning of Speech Representations in the Time Domain,” *arXiv preprint arXiv:2007.00991*, 2020.
- [35] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 2015*. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [36] S. Dieleman and B. Schrauwen, “End-to-End Learning for Music Audio,” in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.
- [37] M. Won, K. Choi, and X. Serra, “Semi-supervised Music Tagging Transformer,” in *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2021.
- [38] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, “Evaluation of Algorithms Using Games: The Case of Music Tagging,” in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, 2009.
- [39] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The Million Song Dataset,” in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.
- [40] J. Davis and M. Goadrich, “The Relationship between Precision-Recall and ROC Curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML ’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 233–240. [Online]. Available: <https://doi.org/10.1145/1143844.1143874>

- [41] J. A. Burgoyne, J. Wild, and I. Fujinaga, “An Expert Ground Truth Set for Audio Chord Recognition and Music Analysis,” in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, 2011.
- [42] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “FMA: A Dataset for Music Analysis,” in *18th International Society for Music Information Retrieval Conference, ISMIR*, 2017. [Online]. Available: <https://arxiv.org/abs/1612.01840>
- [43] G. Tzanetakis and P. Cook, “Musical Genre Classification of Audio Signals,” *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.
- [44] B. L. Sturm, “The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use,” *arXiv preprint arXiv:1306.1461*, 2013.
- [45] J. Spijkervet, “Spijkervet/torchaudio-augmentations,” 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5042440>
- [46] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, “ESSENTIA: An Audio Analysis Library for Music Information Retrieval,” in *International Society for Music Information Retrieval Conference (ISMIR’13)*, Curitiba, Brazil, 04/11/2013 2013, pp. 493–498. [Online]. Available: <http://hdl.handle.net/10230/32252>
- [47] U. Zölzer, X. Amatriain, D. Arfib, J. Bonada, G. De Poli, P. Dutilleul, G. Evangelista, F. Keiler, A. Loscos, D. Rocchesso *et al.*, *DAFX-Digital Audio Effects*. John Wiley & Sons, 2002.
- [48] M. R. Schroeder, “Natural Sounding Artificial Reverberation,” *Journal of the Audio Engineering Society*, vol. 10, no. 3, pp. 219–223, July 1962.
- [49] L. v. d. Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Journal of machine learning research*, vol. 9, pp. 2579–2605, 2008.
- [50] S. Stevens, J. Volkman, and E. B. Newman, “A Scale for the Measurement of the Psychological Magnitude Pitch,” *Journal of the Acoustical Society of America*, vol. 8, pp. 185–190, 1937.
- [51] E. Zwicker, “Subdivision of the Audible Frequency Range into Critical Bands (Frequenzgruppen),” *Acoustical Society of America Journal*, vol. 33, no. 2, p. 248, Jan. 1961.

MUSICAL TEMPO ESTIMATION USING A MULTI-SCALE NETWORK

Xiaoheng Sun¹

Qiqi He¹

Yongwei Gao¹

Wei Li^{1,2}

¹ School of Computer Science and Technology, Fudan University, Shanghai, China

² Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai, China

{19210240112, heqq20, ywgao16, weili-fudan}@fudan.edu.cn

ABSTRACT

Recently, some single-step systems without onset detection have shown their effectiveness in automatic musical tempo estimation. Following the success of these systems, in this paper we propose a Multi-scale Grouped Attention Network to further explore the potential of such methods. A multi-scale structure is introduced as the overall network architecture where information from different scales is aggregated to strengthen contextual feature learning. Furthermore, we propose a Grouped Attention Module as the key component of the network. The proposed module separates the input feature into several groups along the frequency axis, which makes it capable of capturing long-range dependencies from different frequency positions on the spectrogram. In comparison experiments, the results on public datasets show that the proposed model outperforms existing state-of-the-art methods on Accuracy1.

1. INTRODUCTION

Although there are many different ways to describe musical tempo (e.g., measures per minute, bars per minute, or even a range of Italian terms), beats per minute (BPM) is the most commonly used measurement unit. The estimation of BPM plays an important role in a variety of applications, such as music recommendation, automatic accompaniment, playlist generation, etc. Because of its utility, the automatic estimation of tempo has been an important task and received continuous attention in the field of music information retrieval (MIR) [1–4].

Traditional methods for automatic tempo estimation are usually based on hand-crafting signal processing. To estimate the tempo of a given audio segment, an onset strength signal (OSS) function is firstly derived, and the frequency of the major pulses is extracted and converted to BPM. The OSS function is a function whose peaks should correspond to onset times. It can be obtained by various methods, such as means of auto-correlation [5, 6], comb filters [2, 7] and Fourier analysis [8]. Machine learning techniques are also adopted for tempo estimation, including Gaus-

sian mixture models (GMM) [9], support vector machines (SVM) [10, 11], k-nearest neighbors (k-NN) [12, 13], random forests [14] and so on. Since Böck [15] proposed a recurrent neural network (RNN) model to learn beat-level representations from audio signals, attempts to use deep neural networks (DNN) for tempo estimation began to grow [16–18].

In all methods mentioned above, the extraction of BPM depends on some post-processing of OSS functions or beat activation functions. It is only in recent years that the *single-step* tempo estimation systems based on DNN appeared. As the first single-step approach for tempo estimation, the CNN model proposed by Schreiber [19] is capable of extracting BPM value directly from a Mel-scaled spectrogram. In this work, classification is proved to be an effective method for tempo estimation. Adopting a similar idea, Foroughmand [20] proposed the Harmonic-Constant-Q-Modulation (HCQM), a new representation of audio signal, as the input of a relatively simple CNN classification model. The experimental results also showed its effectiveness.

A commonly used metric in tempo estimation is Accuracy1 [3], indicating the percentage of correct estimates allowing a $\pm 4\%$ tolerance. However, automatic tempo estimation systems tend to predict a wrong tempo by a factor of 2 or 3, known as *octave errors*. As an additional measure, Accuracy2 is introduced, which ignores octave errors. In some applicational scenarios (such as DJ software), accurate tempo annotations are mandatory and octave errors are unacceptable [21], but most existing algorithms' performance on Accuracy1 is still far from satisfactory.

Previous works [19, 20] have shown the potential of CNN-based single-step approach to improve performance on Accuracy1. Following the success of these methods, in this paper we propose a CNN-based single-step model named Multi-scale Grouped Attention Network (MGANet). A multi-scale network architecture is designed to aggregate information from different scales to produce superior feature representations. Furthermore, a Grouped Attention Module (GAModule) is proposed to capture long-range dependencies and refine the feature based on the attention mechanism.

The remainder of this paper is organized as follows. In Section 2, we introduce the proposed method in detail. In Section 3, experimental results are presented to show the effectiveness of our method. Finally, we make further conclusion in Section 4.



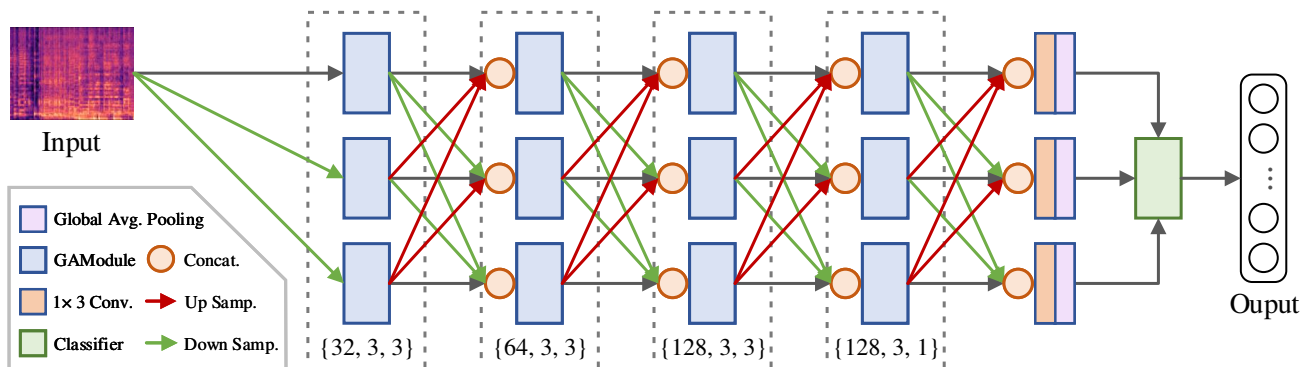


Figure 1: The overall architecture of Multi-scale Grouped Attention Network (MGANet). The numbers in dashed boxes indicate the three parameters of GAModules: {output channel number C , pooling size p , group number k }. Every concatenation operation in the figure is followed by a 1×1 convolution layer to adjust channel number. The classifier consists of a concatenation operation, a fully connected layer, and a softmax layer.

2. APPROACH

2.1 Proposed Model

Same as [19] and [20], we also treat tempo estimation as a classification problem. The output of our model is a probability distribution of 256 BPM classes (from 30 to 285 BPM). Because the Mel-scaled frequency matches closely the human auditory perception, we choose the Mel-scaled spectrogram as the raw feature. First, the original audio data is resampled to 11.025 kHz. Then, we use half-overlapping windows of 1,024 frames, and transform each window into an 81-band Mel-scaled magnitude spectrum. The input of the proposed model is designed as a spectrogram segment of 128 frames, roughly 6 seconds long.

In the rest of this section, we first present the overall architecture of the proposed MGANet. Then, we introduce the GAModule, which is the key component of the network.

2.1.1 Multi-scale Network Architecture

The goal of tempo estimation is to extract a periodic pattern from an audio signal. Therefore, global information of the input spectrogram is particularly important. Due to the characteristics of CNN, overall pattern extraction is usually achieved by stacking multiple layers. But directly repeating convolution layers makes the model difficult to design and optimize. Another way is to use large-size convolution kernels to enlarge the receptive fields. However, this is also costly because of the increase in parameters and multiply-add operations. To solve the problem, we introduce the idea of multi-scale structure, which has been proved to be effective in many classification tasks [22–24]. By downsampling / upsampling the feature to different scales and exchanging information repeatedly, high-level representations can be derived after just a few layers.

As shown in Figure 1, the overall architecture of MGANet is mainly composed of three branches for different scale. In each branch, input features are gradually downsampled over the frequency (vertical) axis, but maintains the resolution through the whole process on the time

(lateral) axis. Furthermore, these feature maps from different scales are merged repeatedly to integrate contextual information, leading to high-level representations amenable to classification.

Specifically, the input spectrogram is first downsampled by 1/2 and 1/4 over the time axis with average pooling, resulting in three representations of sizes (81, 128), (81, 64), and (81, 32). Then, the representations are fed into three parallel branches respectively to perform feature processing. The processing is mainly done by the proposed GAModule described in section 2.1.2. Through the whole structure, we repeat multi-scale fusion by rescaling and concatenation. Average pooling and transposed convolution [25] layers with kernel size of 1×3 are used to perform rescaling. For concatenation, a 1×1 convolution layer with the exponential linear unit (ELU) [26] activation is followed to adjust the channel number.

Processed by GAModules, the features are gradually downsampled over the frequency axis to summarize frequency bands, making the representations easier to detect periodicity. On each branch, the downsampling is repeated four times. Accordingly, the channel numbers of the features are increased. After the above processes, three feature maps with shapes (1, 128, 128), (1, 64, 128), and (1, 32, 128) are obtained. Then, these feature maps are fused again and fed into a 1×3 convolution layer to adjust channel numbers to 256. After global average pooling, three vectors of length 256 are concatenated together. Finally, a fully connected layer takes the vector as input and a softmax layer is used to derive the probability distribution of 256 tempo classes.

2.1.2 Grouped Attention Module

The proposed GAModule structure is shown in Figure 2. The module consists of two parts: a trunk branch performing feature processing, and k attention branches producing an attention mask to capture global context information and recalibrate the output feature map.

The structure of the attention branch is mainly inspired by the global context network (GCNet) [27], which is de-

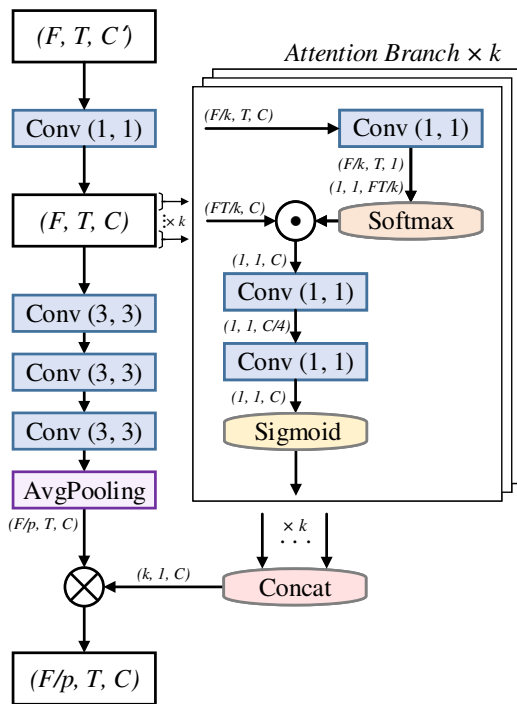


Figure 2: The structure of Grouped Attention Module (GAModule). Feature maps are shown as feature dimensions, e.g. (F, T, C) denotes a feature map with height F , width T , and channel number C . p and k denote pooling size and group number respectively. \odot denotes matrix multiplication and \otimes denotes broadcast element-wise multiplication.

signed for long-range dependency modeling through attention mechanism. The attention mechanism biases the allocation of the most informative feature expressions and suppresses the less useful ones. Recently, the benefits of the attention mechanism have been demonstrated in a series of tasks. We introduce the attention mechanism into GAModule mainly for two purposes: 1) model the long-range dependencies to obtain global context features; 2) reweight the importance of different channels to improve the representational capacity of the refined feature.

Unlike the images in the field of computer vision, the two axes of audio spectrograms have different meanings, which respectively represent frequency and time. Furthermore, it is known that different musical instruments have different frequency ranges, and different frequency ranges have a different impact on the total sound. These facts indicate that different frequency bands contain relatively independent information. Based on these observations, we believe that it's inappropriate to aggregate the whole spatial scope at once to calculate long-range dependencies. Instead, different frequency positions of the feature should be handled separately, which will help to filter the useful information more efficiently. Therefore, different from traditional channel-wise attention models that aggregate the entire feature to generate one attention map (e.g., squeeze-and-excitation networks [28]), we divide the fea-

ture equally into k groups along the frequency axis and send each fragment into an independent attention branch. We termed the operation as *grouped channel attention*.

As shown in Figure 2, the framework of the attention branch is roughly the same as the GC block in GCNet. Firstly, the feature map is squeezed into a channel descriptor by global attention pooling. The pooling is achieved by convolution, softmax, and matrix multiplication. For an input feature map x , the generated descriptor $z \in \mathbb{R}^C$ is calculated by

$$z = \sum_{j=1}^{N_p} \frac{\exp(\text{ELU}(Wx_j))}{\sum_{m=1}^{N_p} \exp(\text{ELU}(Wx_m))} x_j \quad (1)$$

where j and m enumerate all possible positions, and W denotes linear transformation matrix. We adopt ELU as the activation of the convolution layer to further increase robustness. After the pooling, global spatial information is gathered in the descriptor. Then, a bottleneck of two-layer architecture is formed to transform information. We adopt a reduction ratio of 4 and ELU activation in the first layer. A sigmoid function is then applied to rescale the transformation output. Finally, k attention maps with the shape of $(1, 1, C)$ can be obtained. We concatenate these attention maps along the frequency axis and get the output attention map of $(k, 1, C)$.

Simultaneously, in the trunk branch we simply stack three convolution layers with kernel of 3×3 and ELU activation. Because of the existence of attention branches, the trunk does not need a complex structure and too many layers, which reduces the number of parameters and the complexity of the model. We use average pooling with pooling size of $p \times 1$ to downsample the feature map to $(F/p, T, C)$. Finally, broadcast element-wise multiplication is performed to fuse the output of the trunk branch and attention branches. Through the fusion, the output feature map is refined by global contextual information gathered by grouped attention operation.

2.2 Training Data & Augmentation

For training and validation, we adopt the three training datasets used in [19]: *LMD Tempo* (3,611 items), *MTG Tempo* (1,159 items), and *Extended Ballroom* (3,826 items). However, though covering multiple musical genres, the combination of these datasets is not genre-balanced, and some common genres are even missing. It is known that tempo perception is closely related to music genre. For example, for popular music, people usually perceive tempo through drumbeats, while for classical music, people often perceive tempo from bass instruments such as double bass. To alleviate the genre imbalance, we use two additional datasets to supplement the training data:

- **RWC-popular:** To further enhance the model's ability to estimate pop music tempo, we used RWC-popular [29] (a pop music database with 100 pieces) for training. We cut the songs into 30s fragments without overlapping, resulting in 735 items.

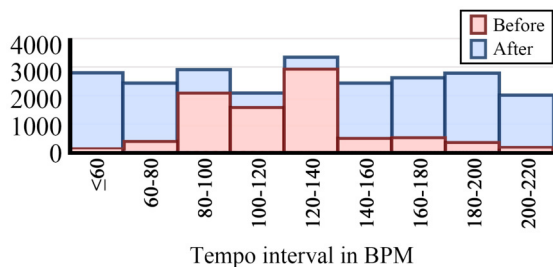


Figure 3: Tempo distribution before and after augmentation.

- **FD-Tempo:** To enrich the genres of training data, we selected some tracks of classical music. For each track, we chose several 30s excerpts with stable tempi and annotated them by manually tagging. Finally, 530 items are obtained as an additional dataset termed *FD-Tempo*.

We use the combination of the five datasets for training and validation. It contains 9,861 tracks with a total length of 41h 3min. Specifically, we randomly choose 500 tracks for validation, and the rest 9,361 tracks are used for training.

To alleviate the BPM class imbalance, we further augment the training set by speeding up / slowing down the selected tracks with factors randomly chosen from 0.7~1.4 without altering the pitch. We retain the original files and make sure that the same audio will not be selected more than 15 times. After augmentation, the number of tracks increases from 9,361 to 23,512. Note that the validation set is not augmented. The tempo distribution in the training set before and after augmentation is shown in Figure 3. Besides, we also adopt the scale-&-crop data augmentation mentioned in [19] to further increase the variability of training data.

2.3 Training Details

For training, the batch size we set is 32. In each epoch, 128 consecutive frames of each sample are randomly selected for training. We choose the categorical cross-entropy as the loss function, and an Adam optimizer [30] is applied with a learning rate of 0.001. We evaluate Accuracy1 of the validation set every 500 iterations, and save the model with the highest accuracy. The training is not stopped until Accuracy1 has not improved for 50,000 iterations.

3. EVALUATION

We choose Accuracy1 (ACC1) and Accuracy2 (ACC2) [3] as the evaluation metrics. Accuracy1 is defined as the percentage of correct estimates allowing a $\pm 4\%$ tolerance. Accuracy2 ignores octave errors by a factor of 2 and 3, and also allows a $\pm 4\%$ tolerance. As mentioned earlier, the demand for highly accurate tempo annotations has become increasingly urgent in many applicational scenarios. Hence we mainly focus on improving Accuracy1.

We focus on the performance on global tempo estimation based on the assumption the tempo of the input track stays constant, and only one BPM value will be returned by

Method	ACC1	ACC2	ACC1	ACC2
w/o AB	77.0	89.9	79.8	95.3
w/o GA	78.5	89.1	79.0	94.2
Single-scale	75.8	89.6	71.2	94.5
Proposed	78.9	91.3	82.1	95.7

(a) GTzan

(b) ACM Mirum

Table 1: Results of ablation study. "w/o AB" and "w/o GA" denote "without attention branch" and "without grouped attention" respectively. Best results are set in bold.

the estimation system. In the experiment, the global tempo is obtained by averaging the outputs of softmax layer over different parts of a full track [19].

3.1 Ablation Study

We study the effect of each idea in our approach. To simplify the discussion, we select two test datasets *GTzan* [31] and *ACM Mirum* [9] for analysis. These two datasets are relatively large (999 and 1,410 items respectively), and both cover rich genres.

To investigate how much the proposed GAModule contributes to the model, we design a set of experiments. Firstly, we remove the attention branches in the module, and only the trunk branch is remained to process features. As shown in Table 1, the performance degrades for both datasets. When focusing on Accuracy1, the performance decreases by 1.9% for *GTzan* and 2.3% for *ACM Mirum*. Then, in another experiment we keep only one attention branch in each module, which can be achieved by setting GAModules' parameter k to 1. The Accuracy1 reduced by 0.4% and 3.1% respectively. For Accuracy2, in both experiments there is also a certain degree of decline. These results indicate that the attention mechanism is helpful to capturing long-range dependencies and therefore improve the generalization of the model. But directly using existing modules may hinder the effect. The proposed grouped attention takes into account the characteristics of spectrogram and achieves further improvements of the model.

Then, we analyze the effect of the multi-scale architecture by changing the architecture to a single-scale one. We remove all downsampled subnetworks and only retain the one with the highest resolution (the topmost branch in Figure 1). As shown in Table 1, model without multi-scale architecture shows significantly worse performance on Accuracy1. The Accuracy1 decreases by 3.1% and 10.9% for *GTzan* and *ACM Mirum* respectively. For Accuracy2, there is also a certain degree of performance degradation. The results demonstrate that the multi-scale can improve the classification ability as well as robustness.

3.2 Comparison with Previous Work

To compare with previous work, we use the same test datasets as in [19] (see [14] for details): *ACM Mirum* [9] (1,410 items), *Hainsworth* [32] (222 items), *GTzan* [31] (999 items), *SMC* [33] (217 items), *GiantSteps* [34] (664

Dataset	böck	schr	foro	mgan
ACM Mirum	74.0	79.5	73.3	82.1
Hainsworth	80.6*	77.0	73.4	77.5
GTzan	69.7	69.4	69.7	78.9
SMC	44.7*	33.6	30.9	29.0
GiantSteps	58.9	73.0	83.6	90.2
Ballroom	84.0*	92.0	92.6	95.1
ISMIR04	55.0	60.6	61.2	61.7
Combined	69.5	74.2	74.4	79.8

(a) Accuracy1

Dataset	böck	schr	foro	mgan
ACM Mirum	97.7	97.4	96.5	95.7
Hainsworth	89.2*	84.2	82.9	87.8
GTzan	95.0	92.6	89.1	91.3
SMC	67.3*	50.2	50.7	44.7
GiantSteps	86.4	89.3	97.9	97.6
Ballroom	98.7*	98.4	98.7	97.7
ISMIR04	95.0	92.2	87.1	88.8
Combined	93.6	92.1	92.0	91.9

(b) Accuracy2

Table 2: Comparison with the results published by Böck (böck) [15], Schreiber (schr) [19], and Foroughmand (foro) [20]. Best results per test dataset are set in **bold**. Asterisk (*) denotes that the corresponding dataset were used for training.

items), *Ballroom* [3] (698 items), and *ISMIR04* [3] (465 items). The union of all test datasets is referred to as *Combined*. The most recent annotations available are used.

We compare our work (mgan) with previous studies by Schreiber (schr) [19] and Foroughmand (foro) [20]. These two methods are both CNN-based single-step models that we are committed to improve. We consider them as the state-of-the-art among single-step approaches. In addition, we also compare the model with an RNN-based traditional periodicity analysis approach by Böck (böck) [15]. The results are shown in Table 2. Note that *Ballroom*, *Hainsworth*, and *SMC* are used for training in böck (values marked with asterisks *).

Focusing on Accuracy1, the experimental results show that the proposed model surpasses other methods in most cases, which proves the effectiveness of the proposed idea to improve Accuracy1. Especially for *GaintSteps* (664 electronic dance music excerpts), there shows a significant improvement of over 6.6%. The richness of electronic dance music in training data can be considered as a reason. The good performance in *ACM Mirum* and *GTzan* (both multi-genre datasets) shows the generalization potential of our model. Moreover, for *Hainsworth*, the model achieves the highest Accuracy1 among single-step approaches. Finally, the proposed method also reaches the highest Accuracy1 for *Combined* (79.8%) compared with other methods, gaining improvement of 5.4%.

As for Accuracy2, it can be observed that böck achieves the highest accuracy in most cases. Ignoring böck, the proposed model shows a similar performance to other single-step methods.

Among all datasets, the worst results of our model are obtained for *SMC*. The dataset was designed to be difficult to estimate tempo, covering various genres. Although we have tried to supplement and augment the training data, the genre-imbalance problem has not been solved very well. This indicates the necessity to supplement more data with different genres in the future work.

3.3 Comparison with Multi-task Approaches

In recent years, some works [17, 18] have not only focused on a single rhythm attribute, but combined the estimation

	Accuracy1	Accuracy2
<i>ACM Mirum</i>		
böck19 [17]	0.749	0.974
böck20 [18]	0.841	0.990
mgan	0.821	0.957
mgan+	0.846	0.970
<i>GiantSteps</i>		
böck19 [17]	0.764	0.958
böck20 [18]	0.870	0.965
mgan	0.902	0.976
mgan+	0.861	0.973
<i>GTzan</i>		
böck19 [17]	0.673	0.938
böck20 [18]	0.830	0.950
mgan	0.789	0.913
mgan+	0.796	0.931

Table 3: Comparison with multi-task approaches. mgan+ is trained by multi-task learning with beat tracking. Best results per test dataset are set in **bold**.

of interconnected rhythm attributes (such as beats, downbeats, etc.) by multi-task learning, so that these highly related tasks can reinforce each other. These approaches are capable of embedding more musical knowledge into a single model, and enrich the training data of each task. In order to further explore the potential of the proposed MGANet and compare its performance with multi-task approaches, we conduct experiments with reference to [17], combining the beat tracking task to our model.

To predict beat positions, we add a branch to the original network structure. The inputs of the branch are the feature maps before sent into tempo classifier, with shapes of (1, 128, 128), (1, 64, 128), and (1, 32, 128). The low resolution feature maps are up-sampled to 128 frames length on time axis by transposed convolution layers. Then, the concatenated feature map with shape (1, 128, 384) is processed by three 1 × 3 convolution layers (output channel number are set to 128, 32, and 1 respectively). After a sigmoid operation, the beat activation function is derived. This extended network structure is trained as a multi-output model to combine the two tasks.

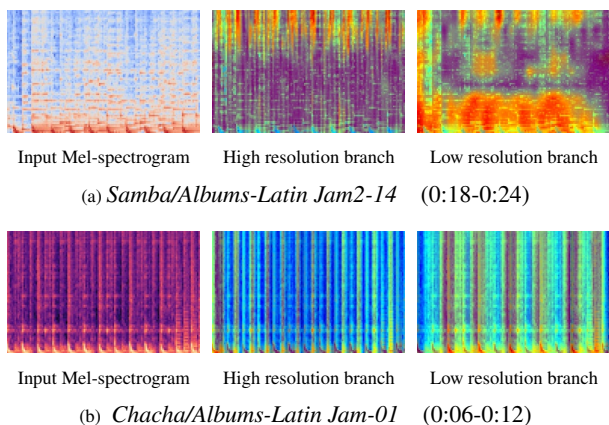


Figure 4: Grad-CAM visualizations for layers on different resolution branches.

For the training of beat tracking, we use a combination of the following datasets: *Hainsworth* [32], *SMC* [33], *Ballroom* [3], *ISMIR04* [3], *Beatles* [35], and *HJDB* [36]. As for the training of tempo estimation, the training and validation datasets in section 2.2 are used. To further enrich the data, beat annotated datasets are also adopted for the training of tempo classifier, using the average BPMs derived from beat annotations as training labels. We train the two task alternatively every epoch, without changing other experimental settings mentioned in section 2.3.

The experimental results are shown in Table 3. Three datasets *ACM Mirum* [9], *GTzan* [31], and *GiantSteps* [34] are used as test datasets. We compare our works (the original model *m_{gan}* and the multi-task model *m_{gan+}*) with two multi-task approaches *böck19* [17] and *böck20* [18]. By multi-task training, improvement can be observed on *ACM Mirum* and *GTzan*. Especially for *ACM Mirum*, the Accuracy1 is increased by 2.5%, achieving the best result among all approaches. Because the two test datasets are both multi-genre datasets, it can be considered that the good performance comes from not only the multi-task learning, but also the beat tracking datasets with rich music genres. As for *GiantSteps*, *m_{gan+}* performs better than *böck19* and *böck20*, but a bit worse than *m_{gan}*. This is also due to the supplement of data, which affects the dominant position of dance music in training data.

3.4 Grad-CAM Analysis

Gradient-weighted Class Activation Mapping (Grad-CAM) [37] is a method that can faithfully highlight the important regions in inputs for a CNN-based classification model. It uses the gradient information in back-propagation as weights (grad-weights) to explain the network’s decisions. We visualize the activation maps derived by Grad-CAM as shown in Figure 4 and Figure 5. Red indicates the part more important in predicting tempo while blue contributes less.

Figure 4 shows the activation maps on branches with different resolutions. Their inputs are two audio clips from *Ballroom* dataset. Time duration is marked below the corresponding images, following the audio title set in *italic*.

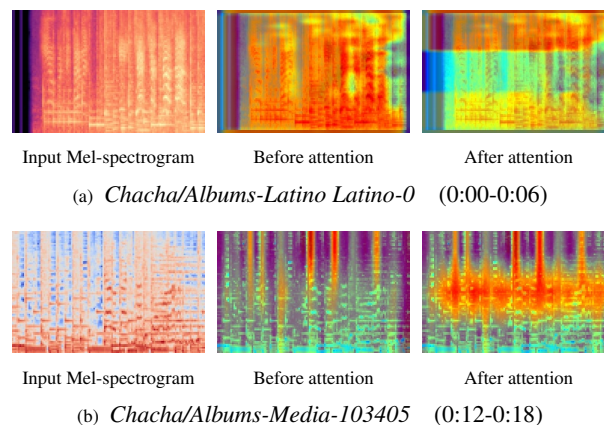


Figure 5: Grad-CAM visualizations for layers before and after grouped attention.

Figure 4a comes from a piece of Samba mainly played by piano and kick drum. The piano in the clip has a higher pitch, played with quarter notes while the kick drum falls on every beat in the bar. It can be observed from the activation maps that the model mainly focuses on short-duration parts of piano in the high-resolution branch, and the kick drum parts with long duration in the low-resolution branch. As for the second example, which is a Cha Cha song, the beat positions can be identified from kick drum in low-frequency part, vocal in middle-frequency part, and claves in high-frequency part. Figure 4b shows that the low-resolution branch considers downbeats to be important, while the high-resolution branch focus on not only downbeats but every other beat in a bar. It can be proved that the multi-scale structure is capable of integrating useful information with different granularities.

We also visualize the activation maps before and after the proposed grouped channel attention to explore its effect. The results are shown in Figure 5. The music excerpt of Figure 5a is played with regular claves and double bass, hence the high-frequency part and the low-frequency part contribute more to tempo estimation. The attention branch reweights the feature maps from the trunk branch, giving top and bottom parts higher weights to detect tempo information easier. In contrast, the vocal dominates the rhythm information in the song of Figure 5b, thus the model gives higher attention to the middle-frequency part after grouped attention. By grouped attention, the network can efficiently find which part would be considered to be important for tempo estimation.

4. CONCLUSION

In this paper, we propose a new CNN-based single-step approach for tempo estimation. We introduce the idea of multi-scale network to construct the architecture of the proposed MGANet. The GAModule with the grouped channel attention is designed to be the key component of the network. Compared with previous work, the proposed approach exhibits good performance on Accuracy1 and outperforms existing models in most cases.

5. ACKNOWLEDGEMENT

This work was supported by National Key R&D Program of China (2019YFC1711800), NSFC (61671156).

6. REFERENCES

- [1] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," in *Proc. of 2nd ACM International Conference on Multimedia*, 1994, pp. 365–372.
- [2] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *The Journal of the Acoustical Society of America*, vol. 103, no. 1, pp. 588–601, 1998.
- [3] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.
- [4] H. Schreiber, "Data-driven approaches for tempo and key estimation of music recordings," Ph.D. dissertation, Friedrich-Alexander-Universität ErlangenNürnberg (FAU), 2020.
- [5] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.
- [6] M. Alonso, G. Richard, and B. David, "Accurate tempo estimation based on harmonic + noise decomposition," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–14, 2007.
- [7] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 342–355, 2005.
- [8] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing, "On tempo tracking: Tempogram representation and kalman filtering," *Journal of New Music Research*, vol. 29, no. 4, pp. 259–273, 2000.
- [9] G. Peeters and J. Flocon-Cholet, "Perceptual tempo estimation using gmm-regression," in *Proc. of 2nd International ACM Workshop on Music Information Retrieval with user-centered and multimodal strategies*, 2012, pp. 45–50.
- [10] A. Gkiokas, V. Katsouros, and G. Carayannis, "Reducing tempo octave errors by periodicity vector coding and svm learning," in *Proc. of the 13th Int. Society for Music Information Retrieval Conf.*, 2012, pp. 301–306.
- [11] G. Percival and G. Tzanetakis, "Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses," *IEEE/ACM Transactions on Audio, Speech, and Language*, vol. 22, no. 12, pp. 1765–1776, 2014.
- [12] F.-H. F. Wu and J.-S. R. Jang, "A supervised learning method for tempo estimation of musical audio," in *Proc. of the 22nd Mediterranean Conference on Control and Automation*. IEEE, 2014, pp. 599–604.
- [13] F.-H. F. Wu, "Musical tempo octave error reducing based on the statistics of tempogram," in *Proc. of the 23rd Mediterranean Conference on Control and Automation*. IEEE, 2015, pp. 993–998.
- [14] H. Schreiber and M. Müller, "A post-processing procedure for improving music tempo estimates using supervised learning," in *Proc. of the 18th Int. Society for Music Information Retrieval Conf.*, 2017, pp. 235–242.
- [15] S. Böck, F. Krebs, and G. Widmer, "Accurate tempo estimation based on recurrent neural networks and resonating comb filters," in *Proc. of the 16th Int. Society for Music Information Retrieval Conf.*, 2015, pp. 625–631.
- [16] A. Gkiokas and V. Katsouros, "Convolutional neural networks for real-time beat tracking: A dancing robot application," in *Proc. of the 18th Int. Society for Music Information Retrieval Conf.*, 2017, pp. 286–293.
- [17] S. Böck, M. E. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, 2019, pp. 486–493.
- [18] S. Böck and M. E. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, 2020, pp. 574–582.
- [19] H. Schreiber and M. Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proc. of the 19th Int. Society for Music Information Retrieval Conf.*, 2018, pp. 98–105.
- [20] H. Foroughmand and G. Peeters, "Deep-rhythm for tempo estimation and rhythm pattern recognition," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, 2019, pp. 636–643.
- [21] D. Gärtner, "Tempo detection of urban music using tatum grid non negative matrix factorization," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, 2013, pp. 311–316.
- [22] G. Huang and D. Chen, "Multi-scale dense networks for resource efficient image classification," in *Proc. of the International Conference on Learning Representations*, 2018.
- [23] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

- [24] A. A. Adegun and S. Viriri, “Fcn-based densenet framework for automated detection and classification of skin lesions in dermoscopy images,” *IEEE Access*, vol. 8, pp. 150 377–150 396, 2020.
- [25] B. Xiao, H. Wu, and Y. Wei, “Simple baselines for human pose estimation and tracking,” in *Proc. of the European Conference on Computer Vision*, 2018, pp. 466–481.
- [26] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *arXiv preprint arXiv:1511.07289*, 2015.
- [27] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, “Gcnet: Non-local networks meet squeeze-excitation networks and beyond,” in *Proc. of the IEEE/CVF International Conference on Computer Vision Workshops*, Oct 2019.
- [28] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.
- [29] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “Rwc music database: Popular, classical and jazz music databases,” in *Proc. of the 3rd Int. Society for Music Information Retrieval Conf.*, vol. 2, 2002, pp. 287–288.
- [30] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. of the 3rd International Conference on Learning Representations*, 2015.
- [31] U. Marchand, Q. Fresnel, and G. Peeters, “Gtzan-rhythm: Extending the gtzan test-set with beat, downbeat and swing annotations,” in *Extended abstracts for the Late-Breaking Demo Session of the 16th International Society for Music Information Retrieval Conf.*, 2015.
- [32] S. W. Hainsworth, “Techniques for the automated analysis of musical audio,” Ph.D. dissertation, University of Cambridge, 2004.
- [33] A. Holzapfel, M. E. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, “Selective sampling for beat tracking evaluation,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.
- [34] P. Knees, Á. Faraldo Pérez, H. Boyer, R. Vogl, S. Böck, F. Hörschläger, M. Le Goff *et al.*, “Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections,” in *Proc. of the 16th Int. Society for Music Information Retrieval Conf.*, 2015, pp. 364–370.
- [35] M. E. Davies, N. Degara, and M. D. Plumbley, “Evaluation methods for musical audio beat tracking algorithms,” *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.
- [36] J. Hockman, M. E. Davies, and I. Fujinaga, “One in the jungle: Downbeat detection in hardcore, jungle, and drum and bass,” in *Proc. of the 13th Int. Society for Music Information Retrieval Conf.*, 2012, pp. 169–174.
- [37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proc. of the IEEE International Conference on Computer Vision*, 2017, pp. 618–626.

ON THE INTEGRATION OF LANGUAGE MODELS INTO SEQUENCE TO SEQUENCE ARCHITECTURES FOR HANDWRITTEN MUSIC RECOGNITION

Pau Torras Arnau Baró Lei Kang Alicia Fornés

Computer Vision Center, Computer Science Department

Universitat Autònoma de Barcelona

pau.torras@e-campus.uab.cat

{abaro, lkang, afornes}@cvc.uab.cat

ABSTRACT

Despite the latest advances in Deep Learning, the recognition of handwritten music scores is still a challenging endeavour. Even though the recent Sequence to Sequence (Seq2Seq) architectures have demonstrated its capacity to reliably recognise handwritten text, their performance is still far from satisfactory when applied to historical handwritten scores. Indeed, the ambiguous nature of handwriting, the non-standard musical notation employed by composers of the time and the decaying state of old paper make these scores remarkably difficult to read, sometimes even by trained humans. Thus, in this work we explore the incorporation of language models into a Seq2Seq-based architecture to try to improve transcriptions where the aforementioned unclear writing produces statistically unsound mistakes, which as far as we know, has never been attempted for this field of research on this architecture. After studying various Language Model integration techniques, the experimental evaluation on historical handwritten music scores shows a significant improvement over the state of the art, showing that this is a promising research direction for dealing with such difficult manuscripts.

1. INTRODUCTION

Optical Music Recognition (OMR) [1] is devoted to the automated transcription of musical documents. As in most document analysis subfields, OMR has gone through a revolution [2] during the last decade, spearheaded by the many advances in Deep Learning. In fact, the latest deep learning architectures are raising the bar of the state of the art, boosting the performance on many different topics of research. In particular, Sequence to Sequence (Seq2Seq) is a Deep Learning architecture that has been quite successful [3]. It was originally conceived for Natural Language Processing and applied to neural machine translation and

related subjects, but it has seen adoption in plenty of other fields, including Handwritten Text Recognition [4]. Recently, this architecture has also shown its potential for OMR, outperforming the well-known Long Short Term Memory Neural Networks with Connectionist Temporal Classification (BLSTM+CTC). [5,6].

As in BLSTM+CTC, Seq2Seq models have the advantage that they do not require symbol-level bounding boxes for training. Instead, the network can learn to identify symbols in an image from the ground-truth token sequence alone. This might not be especially relevant when working with typeset scores, since this information can be provided with relative ease, but it becomes crucial when no such information is available or it is very costly to obtain. This is the current situation for handwritten music recognition in general [7], but more remarkably so in historical music scores [8,9].

Historical handwritten scores are particularly interesting to recognise because there are many of them stored in archives, churches and libraries throughout. Most of them have never been transcribed, which makes it important to devote efforts towards their conservation, transcription, study and dissemination. However, aside from the aforementioned lack of detailed-annotated data, these scores are much harder to recognise than regular typeset ones because of hundreds of years worth of paper degradation, the evolution of music notation conventions and the irregular nature of handwriting, which leads to many ambiguities and hard-to-read passages even for trained humans.

As expected, even the recent Seq2Seq architectures fail in such scenario. Nevertheless, in the handwritten text recognition literature, we have found that the incorporation of a Language Model (LM) can tackle most of these ambiguities. This technique consists on the application of a statistical LM trained to identify probable sequences of tokens, which can then be used to assess the likelihood of the recognised sequence and perform due corrections in the case of an unreasonably unexpected output [10,11]. As in n -grams, it regulates what sequences are considered most likely.

Inspired by this idea, in this work we explore the integration of LMs into a Seq2Seq architecture to minimise the ambiguities when recognising historical handwritten



© P. Torras, A. Baró, L. Kang and A. Fornés. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** P. Torras, A. Baró, L. Kang and A. Fornés, "On the Integration of Language Models into Sequence to Sequence Architectures for Handwritten Music Recognition", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

scores. Concretely, we integrate a LM through three different techniques: Shallow, Deep [10] and Candidate Fusion [11]. From the exhaustive evaluation of their performance on historical manuscripts, we discuss the advantages and disadvantages of these models, concluding that they are capable to significantly boost the performance in the aforementioned domain.

The structure for this document is the following. An overview of current trends in music recognition is provided in Section 2. Section 3 is devoted to describing the architecture. Section 4 describes the adaptation of the input data for music score recognition, and the datasets employed to train the LMs. Section 5 summarises experiments performed to evaluate the performance of the various models. Section 6 is a discussion of the results and section 7 addresses the conclusions and closing words.

2. PREVIOUS WORK

Prior to the Deep Learning “revolution” of the last decades there was mainly one typical pipeline for OMR, which consisted on a set of well-established steps [1]: image pre-processing and staff segmentation, music symbol recognition, music notation reconstruction and final representation construction. This, however, changed with the advances made in Deep Learning, which led to two distinct kind of approaches.

On the one hand, there has been a “continuist” approach, where the aforementioned pipeline is more or less preserved, but one or more steps are implemented with Deep Neural Models. Examples of these systems can be found in plenty of works. For example, Calvo-Zaragoza *et al.* proposed a new method for staff line detection [12] through the use of Convolutional Neural Networks; Calvo-Zaragoza also presented work regarding pixel-level document binarization with Convolutional Neural Networks alongside Fujinaga and Vigiensoni [13]; Hajic *et al.* [14] proposed a way to segment musical symbols and classify them in a single step using U-Nets; Pacha *et al.* [15] proposed a method to reconstruct the relationships between segmented symbols through the use of Convolutional Neural Networks together with a novel graph-based system to represent them.

On the other hand, there have been attempts to perform the full OMR pipeline using a single neural-based end-to-end architecture. The work of van der Wel *et al.* [5] is interesting because it is the first precedent of Seq2Seq for OMR, although it was exclusively designed for typeset scores. Newer models such as Huang *et al.* [16] YOLO darknet53-based architecture seem to have dropped the recurrent aspect while improving on the state of the art in this context of typeset scores.

In terms of handwritten scores, RNNs are still being used with good results. Baró *et al.* [17] used a CRNN model on handwritten scores, which was the first single-step baseline that was established for this domain. For handwritten old scores, Calvo-Zaragoza [9] proposed a CRNN + CTC model with an n -gram LM for recognising a specific set of scores in Mensural notation. Lately,

Baró *et al.* [6] presented a single-step system based on a Seq2Seq model with an attention mechanism for recognising handwritten scores in common western notation.

The earliest instance of Language Modelling subject to a recognition task is [18], which used n -grams in order to make OCR machines context-aware and therefore more robust. Since n -grams are fairly easy to implement and give reasonably good results, they have been used quite consistently even in recent times [9], although with the rise of DNN technology other approaches based on neural LMs have emerged. Indeed, the integration of LMs into Seq2Seq architectures has also been studied through various methods that take advantage of RNN-based LMs. These were introduced for fields within or related to Natural Language Processing like neural machine translation [10], handwritten text recognition [11], or speech recognition [19], although the core idea is equally valid whenever the final target is any ordered sequence of tokens.

In summary, Seq2Seq-based recognisers are promising architectures that have shown to benefit from the integration of LMs. However, while LMs have been applied to music recognition through n -grams [9], no precedents of RNN-based LMs along with Seq2Seq OMR architectures exist. Therefore, we hypothesise that such integration has the potential to improve the current state-of-the-art results in OMR, as it has already been observed in other related fields [10, 11].

3. SEQUENCE TO SEQUENCE-BASED OMR

This section describes the core Seq2Seq system for OMR, the three LM models and their integration into the architecture.

As stated before, our architecture is inspired in the Seq2Seq OMR model described in [5, 6]. The whole architecture is depicted in Figure 1, with a reference to the LM integration step (see the dashed lines). Next, we describe its properties and its inference process.

3.1 Sequence to Sequence model

Seq2Seq models [3] are architectures capable of converting arbitrary-length input sequences into arbitrary-length output sequences. They are an Encoder-Decoder architecture: the input sequence is transformed by the Encoder into an intermediate representation that the Decoder will use to generate the output sequence.

A score image, which is treated as a sequence of column vectors, is fed into a Convolutional Neural Network based on a VGG19 [20] with its last max pooling layer removed. Then, the Encoder, a bidirectional stack of Gated Recurrent Units (GRU) [21], generates an intermediate representation comprised of as many feature vectors as the convolutional output. The idea behind this bidirectionality is that, by processing the input image from both ends of the sequence, the model has the information of the full image for all inference steps and is therefore much more context-aware.

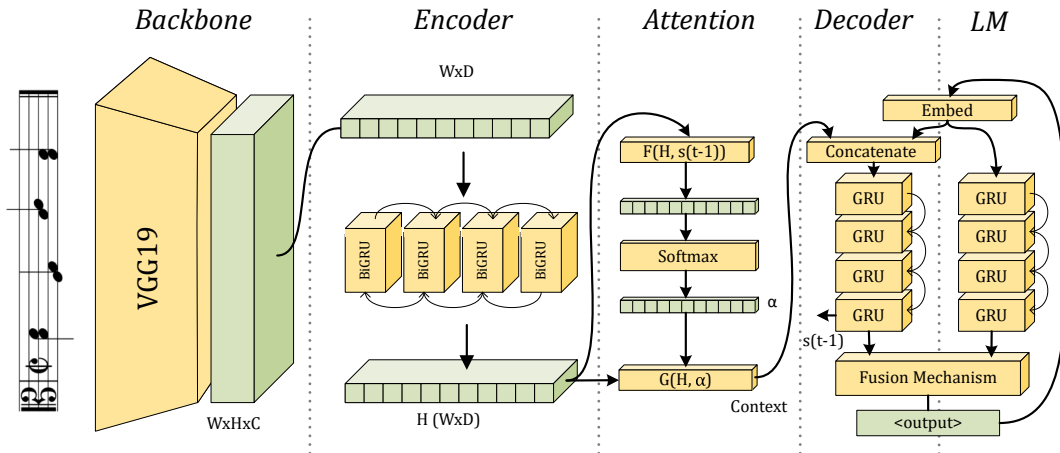


Figure 1. Summary of the Seq2Seq model used in this work.

When the Encoder has processed the input image completely, the Decoder iteratively receives the generated hidden state alongside the last predicted token in the sequence, which produces the next output token until a special “end” token is produced. In order to assess the relevance of each of the hidden state’s vectors, a location attention mechanism (Chorowsky *et al.* [22]) weights each vector in the hidden state, with the idea of making the model capable of “focusing” on specific regions of the input image.

3.2 Language Model Integration

LMs are systems that model the probability distribution of possible tokens at time step t conditioned by predictions at time steps 1 to $t - 1$. Many language modelling techniques exist throughout such as n -grams [9], but RNNs are known to be a superior choice overall [23], thus this work focuses on a single LM architecture consisting on four stacked GRUs.

LM integration with Seq2Seq models has been explored through various approaches aiming at improving recognition performance. Three of such approaches have been explored in this work: Shallow, Deep [10], which are among the most used methods, and Candidate Fusion [11], which showed good performance on handwritten text recognition.

Figure 2 shows a depiction of these methods and the following paragraphs are devoted to describing them in detail.

3.2.1 Shallow Fusion (Gulcehre *et al.* [10])

This technique was devised in the context of neural machine translation. It is a very intuitive system in which the final output is obtained by summing log probabilities from the LM and the Seq2Seq model. Let P , P_{CL} and P_{LM} be the probability distribution of tokens predicted by the full model, the Seq2Seq component and the LM respectively, and let λ be an arbitrary hyperparameter set on training, Shallow Fusion is implemented as

$$\log P(y_t|y_1 \dots y_{t-1}) = \log P_{CL}(y_t|y_1 \dots y_{t-1}) + \lambda \log P_{LM}(y_t|y_1 \dots y_{t-1}). \quad (1)$$

3.2.2 Deep Fusion (Gulcehre *et al.* [10])

This method comes from the same context as Shallow Fusion and builds further on its idea by merging both LM and Seq2Seq’s outputs in a more fine-grained manner. Essentially, the λ parameter is substituted by a coarse gating mechanism and the final output is obtained using more information from across the model. Let σ be the sigmoid activation function and W_{DF} and b_{DF} be learnable parameters, Deep Fusion is implemented as

$$P(y_t|y_1 \dots y_{t-1}) = \text{softmax}(W_{DF}h_t^{DF} + b_{DF}). \quad (2)$$

The Deep Fusion hidden state h_t^{DF} is obtained concatenating the Seq2Seq context vector c_t , the Classifier’s hidden state h_t^{CL} and a gated version of the LM’s hidden state, as seen in

$$h_t^{DF} = [c_t; h_t^{CL}; g_t h_t^{LM}]. \quad (3)$$

The coarse gate mechanism g_t is in its turn computed as

$$g_t = \sigma(v_g^T h_t^{LM} + b_g) \quad (4)$$

where v_g and b_g are learnable parameter vectors. We use the implementation seen in [24], which does not feed the previously inferred character in equation 3.

3.2.3 Candidate Fusion (Kang *et al.* [11])

This method was shown to be more suitable than Deep and Shallow fusion in the context of Handwritten Text Recognition. The core idea behind it is to reinforce the decision process of the Seq2Seq Decoder at each output time step by feeding it the output of the LM, so that both pipelines can be leveraged accordingly. It can be defined as

$$h_t = \text{Decoder}([c_t, y_{t-1}, p_{t-1}^{lm}], h_{t-1}) \quad (5)$$

where c_t is the current context vector, y_{t-1} is the previous prediction and p_{t-1}^{lm} is the probability distribution obtained by the LM with the output at the previous time step.

Some comparisons can be drawn among all three methods both from their literature and their architectures. The main selling point for Shallow Fusion is that it adds very little complexity into the model, which is compensated by

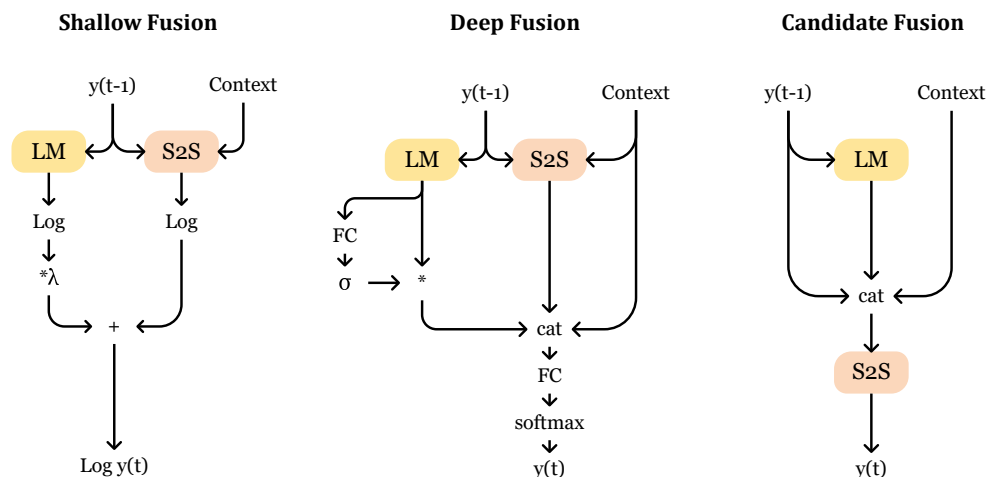


Figure 2. Dataflow graph depicting every integration method that was implemented

the fact that it requires hyperparameter tuning for its λ value and the impossibility to modify said value depending on the LM output. Deep Fusion poses as a more flexible model that can learn to weight the importance of the output of the LM, at the cost of incorporating further layers into the model. Finally, Candidate fusion boosts the communication between the LM and the Seq2Seq component and produces an output obtained not by linearly combining both outputs at the final inference step, but rather by letting the Seq2Seq combine the criteria of visual features and Language Probabilities. However, this might involve more training for the model to become acquainted with the output of the LM.

All these methods require both the classifier and the LM to be properly pretrained for successful integration. More detail is provided in section 5.

4. DATASETS

This section describes the adaptation of the data for music score recognition using the Seq2Seq architecture.



Figure 3. Sample measures from the SM, SO and HW datasets respectively.

4.1 Serialising Input Data

Input music measures are annotated at a *musical primitive* level. This means that notes are not full tokens by them-

selves, but are instead divided into their core elements: noteheads with their pitch and type (black or white), stems with their orientation, flags, beams and so on. There are also some tokens which are atomic, such as time signatures, dots, accidentals and rests, and some twin tokens that require opening and closing, such as beginning and end segments of a slur or a beam.

The `epsilon` token is a special one used to separate groups of primitives belonging to different symbols placed in adjacent columns. Thanks to this, 2D music notation can be serialised into a flat one-dimensional array of tokens that Seq2Seq can work with. An example of this format is given in Figure 4.

4.2 Training Datasets

Various datasets of differing characteristics were used to train the models, each of them for a specific task (more detail on section 5). Their description is shown below along with some examples (See Figure 3). Note also that, when referring to synthetic datasets, we imply the musical content of these scores is randomly generated (thus we assume that these datasets are, except for some trivial examples, disjoint).

Synthetic Modern (SM): Dataset comprised of polyphonic measures of synthetic typeset scores. Most usual music symbols can be found: G, C and F clefs, accidentals, note components, time signatures and barlines, to name a few. An example is shown in Figure 3a.

Synthetic Old (SO): A synthetic dataset with monophonic measures distorted with typical paper degradation effects. Similar to SM in terms of the range of tokens present. An example is shown in Figure 3b.

Handwritten (HW): A compilation of measures of real handwritten scores from a church in Barcelona called Santa María del Pi. They were composed by its Kapellmeister Pau Llinàs back in the 18th century for choral interpretation during liturgical events. An example is shown in Figure 3c.

Adjusted Synthetic Modern (ASM): A reduced version of the SM dataset (see section 5).

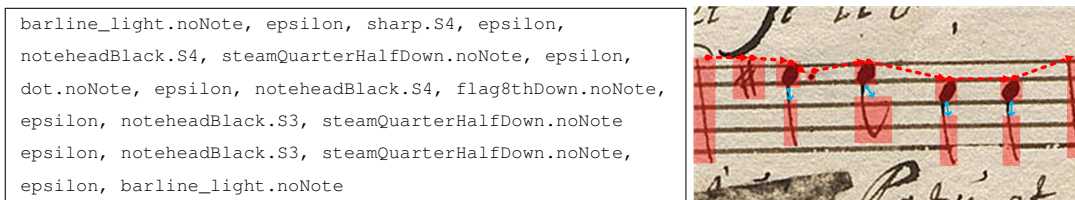


Figure 4. Sample measure from the HW dataset with its ground truth annotation. Bounding boxes indicate the boundaries of what each “atomic” token is, dotted arrows indicate epsilons in the transcription and small vertical arrows indicate symbols that are placed together between epsilons (or rather, primitives belonging to the same symbol).

5. EXPERIMENTS

The evaluation of all proposed LM integration methods was performed under two training strategies, characterised by the dataset which was used to pretrain the LM. Regardless of the LM dataset, training parameters and strategies were the same altogether. For the sake of reproducibility, Table 1 summarises these hyperparameters and characterises the various datasets employed throughout.

Since the goal for our work is to improve results on handwritten scores, a training strategy was conceived to gain the benefits of extra data from synthetic scores while preventing optimisation towards them. All integration methods tested hereby require both the LM and the recogniser to be properly pretrained. Thus, we first trained a LM with an unmodified version of the SM dataset. Since we were aware that this dataset had many tokens that were not present in the HW one, we created a version of the SM dataset comprised of the 66% of samples which contained a higher ratio of tokens also present in the HW one, which we will refer to as ASM, and we trained another LM with it. The idea was trying to “de-noise” the output of the LM in HW scores so that its predictions had a higher level of confidence.

In both cases, we trained the Seq2Seq classifier with the unmodified SM dataset until the model did not improve for 30 epochs. We then joined both models and trained them using a Curriculum Learning strategy: initially, 90% of samples in the training mix were from the SO dataset and the remaining 10% from the HW dataset. Every 10 epochs the proportion of SO scores decreased by 10% over the total, down to 10%. Since the number of SO samples is much higher than the number of HW samples, the latter were duplicated randomly to match the number of samples from the former. The incorporated image augmentation system for training was used to prevent overfitting on input images. Note also that experiments with homogeneous datasets were avoided since they were seen to decrease performance in earlier tests.

Validation and test were performed using HW dataset samples. Lastly, for Shallow Fusion we used a $\lambda = 0.1$ after testing three instances of the full architecture on the SM dataset and keeping the value that gave better output results.

6. EXPERIMENTAL RESULTS

This section is devoted to explaining the results obtained with the aforementioned training strategies. This is, Shallow, Deep and Candidate Fusion using a LM pretrained with the SM or the ASM Dataset. Numerical results are provided using the Symbol Error Rate (SER(%)) metric, which is defined as

$$SER(\%) = \frac{I + R + S}{T} \cdot 100 \tag{6}$$

where I , R and S are the number of token insertions, removals and substitutions in order to obtain the ground truth sequence from the predicted sequence and T is the length of the ground truth sequence. Lower values mean better results.

6.1 Quantitative Results

Table 2 shows the results obtained from all of our experiments. Given the fact that Seq2Seq model pre-training on the SM gave results well below 1% SER(%), we believe it is not worth to experiment with the addition of a LM when transcribing synthetic samples. Instead, we show test results using the training strategy in 5 and two baseline models: the BLSTM + CTC model and the LM-less Seq2Seq model [6]. All results are obtained using the HW test partition as input.

Best baseline results are 56.20% and 31.79% of SER(%) for BLSTM + CTC and Seq2Seq respectively. However, authors comment in the paper that there might be overfitting in the best result of the former model because training was done only with handwritten samples. When training with a mix of synthetic and real data, the authors state an increase from 56.20 SER(%) to 74.40 SER(%).

Our proposed models obtained mostly better results than those from the Baseline. Candidate and Deep Fusion are the better performing architectures, with best results (in bold in Table 2) between 5 and 6 SER(%) points below the baseline. Shallow Fusion obtained best results on par with the baseline.

The general pattern is that earlier iterations perform worse than latter ones. There are a few exceptions, which are the SM version of Deep Fusion and the ASM version of Shallow Fusion, which obtain better results in intermediate phases. This might be caused by the fact that the model might be entering local minima, which it may leave after further epochs.

Table 1. Reproducibility table. The first segment is devoted to training hyperparameters. The second one to showing relevant information about the various datasets that have been employed.

Parameters	All Training	Data	SM	SO	HW
Optimiser	Adam	Train Samples	18,900	17,872	147
Learning Rate (LR)	$3 \cdot 10^{-4}$	Valid Samples	6,300	5,957	49
LR Checkpoints	@ 20, 40, 60, 80, 100 epoch	Test Samples	6,300	5,957	49
LR Sigma	0.5	Avg. Line Length	22	15	17
Loss Function	Cross-Entropy	Classes	109	123	62

Table 2. Summary of performed experiments and results in SER(%) (Lower is better). The table header indicates the proportion of Synthetic scores against Handwritten scores. The “Pre” column indicates the LM pretraining dataset.

Model	Pre	90-10	80-20	70-30	60-40	50-50	40-60	30-70	20-80	10-90	0-100
CNN + BLSTM [6]	-	-	-	-	-	-	-	-	-	-	56.20
Seq2Seq Baseline [6]	-	60.03	-	-	66.20	-	43.38	-	37.86	34.56	31.79
Seq2Seq + Deep LM	SM	31.30	28.52	29.87	29.37	28.05	26.11	27.74	27.37	28.32	-
Seq2Seq + Shallow LM	SM	36.79	32.91	33.27	33.36	31.76	32.75	30.87	30.72	30.58	-
Seq2Seq + Cand. LM	SM	33.50	28.93	28.64	28.08	27.48	26.82	27.23	26.61	25.80	-
Seq2Seq + Deep LM	ASM	28.24	29.53	27.82	27.36	25.95	27.21	25.63	25.15	25.54	-
Seq2Seq + Shallow LM	ASM	35.34	34.75	36.67	32.42	34.23	34.52	33.76	33.79	35.13	-
Seq2Seq + Cand. LM	ASM	32.07	28.61	28.71	27.55	27.71	27.20	27.77	28.04	25.73	-

Another general remark is that models pretrained with the ASM dataset seem to perform slightly better, with a 0.96 SER(%) improvement in Deep Fusion and a 0.07 one in Candidate Fusion, although this difference could be also attributed to optimisation since it is not substantial.

6.2 Discussion

Numerical proof is found that a LM does help improve recognition results in historical handwritten music scores, especially when using Candidate or Deep Fusion. However, we agree that it is not easy to assess their differences outside of a subjective qualitative study.

Expectedly, LM lowers the presence of certain syntactic mistakes (for instance, tokens that require a specific successor) or provides information on tokens that appear frequently. There is, however, a set of possible recognition mistakes that the LM was initially presumed to be able to correct which we found it unable to. The most relevant was enforcing the beat of the bar that is being recognised. It can be argued that at no point in the measures that comprise the dataset the time signature is indicated aside from its very beginning, but since the training dataset is written exclusively in a 4/4 time signature, the LM might have adapted to measures adding up to a beat value. Perhaps this is due to the purely statistical approach taken with the LM, so some postprocessing (based on music notation rules) may be needed for approaching such consistency checks.

Other “artistic” aspects of music cannot be corrected with the LM, such as the pitch and duration of notes, which can only be predicted up to a certain point based on its frequency of appearance. This was expected and, unsurprisingly, most noteheads have been predicted on the most common range within the original score.

A final remark is that we have observed that the adjustment strategy attempted with the ASM dataset showed no significant improvement. Instead, in order to better align training and test datasets without overfitting, more data should be used for training. A common issue when trying to collect data for this purpose is that most common transcriptions of old music adapt their notation style to current trends, which defeats the purpose of using such data for recognition.

7. CONCLUSION

This work successfully explored the integration of LMs into a Seq2Seq OMR architecture for recognising historical handwritten scores. An improvement of around 6 SER(%) points from the baseline was obtained when using a Deep Fusion mechanism, lowering it to 25.15 SER(%). This was achieved by reinforcing the model’s capacity to keep consistency on predicted sequences. Thus, we can conclude that the integration of language models into OMR Seq2Seq architectures is a promising research direction worth exploring.

From the results we obtained, we propose some future work avenues. Since language models do not seem to enforce key global aspects like beat, a grammar-based parser might be implemented on top of the neural model in order to correct syntactical mistakes. This could use the probability distribution produced by the neural model to weight all possible corrections. Another improvement could be to use the extra information the LM provides in order to reinforce specific steps within the model, such as the attention mechanism. Perhaps this preemptive information might point the model where to look at in the score image.

8. ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish project RTI2018-095645-B-C21, the CERCA Program/Generalitat de Catalunya, and the FI fellowship AGAUR 2020 FI_B2 00149 (with the support of the Secretaria d'Universitats i Recerca of the Generalitat de Catalunya and the Fons Social Europeu). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

9. REFERENCES

- [1] A. Rebelo, I. Fujinaga, F. Paszkiewicz, A. R. Marcal, C. Guedes, and J. S. Cardoso, "Optical music recognition: state-of-the-art and open issues," *IJMIR*, vol. 1, no. 3, pp. 173–190, 2012.
- [2] J. Calvo-Zaragoza, J. H. Jr, and A. Pacha, "Understanding optical music recognition," *CSUR*, vol. 53, no. 4, pp. 1–35, 2020.
- [3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NeurIPS*, 2014, pp. 3104–3112.
- [4] J. Michael, R. Labahn, T. Grüning, and J. Zöllner, "Evaluating sequence-to-sequence models for handwritten text recognition," in *ICDAR*. IEEE, 2019, pp. 1286–1293.
- [5] E. van der Wel and K. Ullrich, "Optical music recognition with convolutional sequence-to-sequence models," in *ISMIR*, 2017, pp. 731–737.
- [6] A. Baró, C. Badal, and A. Fornés, "Handwritten historical music recognition by sequence-to-sequence with attention mechanism," in *ICFHR*, 2020, pp. 205–210.
- [7] J. Hajič and P. Pecina, "The MUSCIMA++ dataset for handwritten optical music recognition," in *ICDAR*, vol. 1, 2017, pp. 39–46.
- [8] A. Pacha and J. Calvo-Zaragoza, "Optical music recognition in mensural notation with region-based convolutional neural networks," in *ISMIR*, 2018, pp. 23–27.
- [9] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Handwritten music recognition for mensural notation with convolutional recurrent neural networks," *PRL*, vol. 128, pp. 115–121, 2019.
- [10] C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.
- [11] L. Kang, P. Riba, M. Villegas, A. Fornés, and M. Rusiñol, "Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture," *PR*, vol. 112, p. 107790, 2021.
- [12] J. Calvo-Zaragoza, A. Pertusa, and J. Oncina, "Staff-line detection and removal using a convolutional neural network," *MVA*, vol. 28, no. 5-6, pp. 665–674, 2017.
- [13] J. Calvo-Zaragoza, G. Vighienoni, and I. Fujinaga, "Pixel-wise binarization of musical documents with convolutional neural networks," in *MVA*, 2017, pp. 362–365.
- [14] J. Hajic Jr, M. Dorfer, G. Widmer, and P. Pecina, "Towards full-pipeline handwritten omr with musical symbol detection by u-nets," in *ISMIR*, 2018, pp. 225–232.
- [15] A. Pacha, J. Calvo-Zaragoza, and J. Hajic Jr, "Learning notation graph construction for full-pipeline optical music recognition," in *ISMIR*, 2019, pp. 75–82.
- [16] Z. Huang, X. Jia, and Y. Guo, "State-of-the-art model for music object recognition with deep learning," *Appl. Sci.*, vol. 9, no. 13, p. 2645, 2019.
- [17] A. Baró, P. Riba, J. Calvo-Zaragoza, and A. Fornés, "From optical music recognition to handwritten music recognition: A baseline," *PRL*, vol. 123, pp. 1–8, 2019.
- [18] C. Y. Suen, "n-gram statistics for natural language understanding and text processing," *PAMI*, vol. 1, no. 2, pp. 164–172, 1979.
- [19] T. Hori, J. Cho, and S. Watanabe, "End-to-end speech recognition with word-based rnn language models," in *SLT*, 2018, pp. 389–396.
- [20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [21] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*, 2014, pp. 1724–1734.
- [22] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NeurIPS*, vol. 28, 2015, pp. 577–585.
- [23] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.
- [24] S. Toshniwal, A. Kannan, C.-C. Chiu, Y. Wu, T. N. Sainath, and K. Livescu, "A comparison of techniques for language model integration in encoder-decoder speech recognition," in *SLT*, 2018, pp. 369–375.

KIITE CAFE: A WEB SERVICE FOR GETTING TOGETHER VIRTUALLY TO LISTEN TO MUSIC

Kosetsu Tsukuda Keisuke Ishida Masahiro Hamasaki Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Japan

{k.tsukuda, ksuke-ishida, masahiro.hamasaki, m.goto}@aist.go.jp

ABSTRACT

In light of the COVID-19 pandemic making it difficult for people to get together in person, this paper describes a public web service called *Kiite Cafe* that lets users get together virtually to listen to music. When users listen to music on *Kiite Cafe*, their experiences are characterized by two architectures: (i) visualization of each user's reactions, and (ii) selection of songs from users' favorite songs. These architectures enable users to feel social connection with others and the joy of introducing others to their favorite songs as if they were together in person to listen to music. In addition, the architectures provide three user experiences: (1) motivation to react to played songs, (2) the opportunity to listen to a diverse range of songs, and (3) the opportunity to contribute as curators. By analyzing the behavior logs of 1,760 *Kiite Cafe* users over about five months, we quantitatively show that these user experiences can generate various effects (e.g., users react to a more diverse range of songs on *Kiite Cafe* than when listening alone). We also discuss how our proposed architectures can continue to enrich music listening experiences with others even after the pandemic's resolution.

1. INTRODUCTION

Unlike listening to music alone, listening to music with others adds the qualities of feeling social connection and letting others listen to one's favorite songs. For example, the former quality occurs when attending a live concert and sharing the experience with other audience members [1, 2], while the latter quality occurs when people introduce others to their favorite songs [3–5].

These qualities have become hard to enjoy since the COVID-19 pandemic has made it difficult to get together in person and listen to music with others. Instead of attending a live concert, people can listen to the same music at the same time via TV, radio, or live streaming on the web. However, such media provide a poor alternative because the former quality of social connection requires audiences to get together in the same place so that they can see each other's reactions to the music. Similarly, instead of directly introducing others to favorite songs, people can post URL links to them (e.g., YouTube videos of songs) to social net-

working services (SNSs) such as Twitter and Facebook. Even if many SNS users react to a song post (e.g., with a “thumbs up”), there is no guarantee that they actually listened to the song and liked it. Rather, the latter quality of sharing a favorite song with others requires knowing that people who react actually listened to the song.

In light of the above, we propose a web service called *Kiite Cafe*^{1 2} that enables people to get together virtually to listen to music without losing the above qualities. *Kiite Cafe* is characterized by the following two architectures: (i) when users listen to songs on *Kiite Cafe*, each user's reactions are visualized, and (ii) songs played on *Kiite Cafe* are selected from users' favorite songs. To facilitate an intuitive understanding of the user experiences provided by these architectures, we give the following example.

Suppose that Emily is a *Kiite Cafe* user. One day, she logs in to *Kiite Cafe* and finds that 14 users are logged in. Each user is identified by his/her own icon. The users, including Emily, can simultaneously listen to the same song, which is automatically selected and played. Even if the played song has a different mood from songs that Emily usually listens to, if she likes it, she can add it to her list of favorite songs (i.e., her *favorites list*). Because she has encountered a new favorite song, she feels happy to listen to a diverse range of songs. Moreover, when the currently played song is added to her favorites list, architecture (i) visualizes her reaction by displaying a heart symbol on her icon. Because other users' reactions are also visualized, she can see their reactions to feel social connection. For example, one of Emily's favorite songs is played when it is automatically selected by architecture (ii). While her favorite song is playing, she is pleased to notice that a heart symbol is displayed on another user's icon. Then, other users also react to the song, and eventually the heart symbol is displayed on eight users' icons. Architecture (i) thus enables Emily to see the moments when other users start liking one of her favorite songs. This experience makes her feel happy and want other users to listen to another of her favorite songs. Thus, Emily looks forward to another favorite song being played; until then, she stays on *Kiite Cafe* and enjoys other users' favorite songs.

Our contributions can be summarized as follows.

- We propose two architectures for enabling people to simultaneously listen to the same music online while achieving the qualities of social connection and the joy of introducing other people to favorite songs.

¹ “Kiite” means “Listen” in Japanese.

² <https://cafe.kiite.jp>



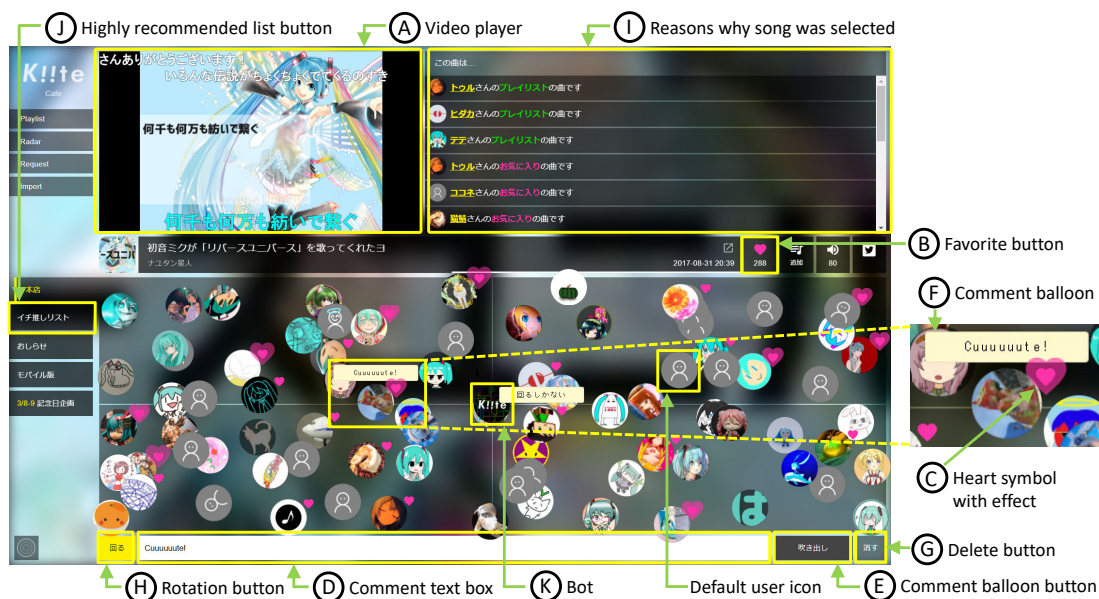


Figure 1. Screenshot of Kiite Cafe.

- We implemented and released a web service, called Kiite Cafe, that applies these architectures.
- We describe three user experiences in which users (1) are motivated to react to songs, (2) can listen to a diverse range of songs, and (3) can contribute as curators; we also discuss the effects of these experiences on users as a result of the two proposed architectures.
- By analyzing logs of user behavior on Kiite Cafe, we show that the architectures do provide the above effects. Specifically, users (1) react to songs more actively as the number of users on Kiite Cafe increases, (2) react to a more diverse range of songs on Kiite Cafe than when they listen to songs alone, and (3) stay on Kiite Cafe longer when they contribute more as curators.

2. OVERVIEW OF KIITE CAFE

Kiite Cafe is implemented as a novel function on Kiite, which is an existing web service for exploring and discovering music. Any Kiite user can use Kiite Cafe. Below, we introduce Kiite’s functions related to Kiite Cafe and then give an overview of Kiite Cafe.

2.1 Kiite

Song data on Kiite are routinely collected from *Nico Nico Douga*, which is one of the most popular video sharing services in Japan. On *Nico Nico Douga*, it is quite popular for both amateur and professional musicians to upload songs created with singing voice synthesizer software called *VOCALOID* [6]. As of the end of Mar. 2021, more than 220,000 songs can be played back on Kiite. When a Kiite user listens to a song, its video clip is played on Kiite by an embedded video player³.

Kiite enables users to effectively find favorite songs by providing novel functions such as exploration of songs based on their emotions and continuous listening to only the choruses of multiple songs. A registered user can set her own icon image, add songs to her favorites list, create playlists, listen to other users’ playlists, and so on.

³ On *Nico Nico Douga*, all songs are uploaded as music videos.

2.2 Kiite Cafe

Fig. 1 shows an overview of Kiite Cafe. When a user logs in, her icon is displayed at a random position in a two-dimensional space that also displays other logged-in users. All of the users listen to the same song played in a video player (A in the figure) at the same time, like a live concert. As mentioned in section 1, Kiite Cafe has two architectures, for visualizing users’ reactions and selecting songs to play from users’ favorite songs. In the rest of this section, we describe the details of each architecture.

2.2.1 Architecture (i): User Reaction Visualization

We visualize the following four kinds of reactions so that users can see each other’s reactions to a played song.

Favorite. When a user likes a played song, she can add it to her favorites list by clicking the “favorite” button (B). When the button is clicked, a heart symbol with an animation effect is displayed at the top right of the user’s icon while the song is playing (C). This enables users to quickly see how many users like a song. When the user had already added the played song to her favorites list, the heart symbol is displayed without the effect.

Comment. When a comment is entered in a text box (D) and the “comment balloon” button (E) is clicked, a comment balloon is displayed above the user’s icon for 90 seconds (F). The user can also manually delete her comment by clicking the “delete” button (G). Users can thus use this function to express their impressions of a played song or have simple communication with each other.

Rotation. A user can rotate her icon by clicking the “rotation” button (H). The icon then rotates clockwise at a uniform rate until the played song ends. The user can also manually stop the rotation by clicking the “rotation” button again. Users can use this function to express feelings like a sense of excitement. However, note that Kiite Cafe does not provide any guidance on when users should use this function, because we want them to use it as they please.

Move. By clicking an arbitrary position in the two-dimensional space, a user can move her icon to the clicked

position. The icon is animated to move to the position in a straight line at a uniform rate. Kiite Cafe does not display any meaning for the quadrants and axes in the two-dimensional space. Instead, as with the *Rotation* function, we leave the usage of the *Move* function to users.

2.2.2 Architecture (ii): Song Selection from Users' Favorite Songs

Let U denote a set of users who are logged in to Kiite Cafe. For each user u , we define S_u as the set of songs included in u 's favorites list or playlists. A played song is selected from $\bigcup_{u \in U} S_u$. The automatic song selection process is invoked before the end of the currently played song, and it consists of the following two steps: (1) selection of a user and (2) selection of a song from the user's favorite songs.

In the first step, if there are biases toward certain selected users, then the selected songs may also be biased. Moreover, some users may become frustrated if their favorite songs are not selected at all. To avoid such biases and satisfy every user, we developed an algorithm that can randomly but fairly select users and thus diversify the played songs. Suppose that user u is selected in the first step, such that every user has an equal chance to be selected. When song $s \in S_u$ is randomly selected in the second step, the reason for its selection is displayed, e.g., "This song is in u 's playlist" (first row of ① in Fig. 1). If s is also among other users' favorite songs, that information is displayed (second and later rows of ①) so that those users can notice that one of their favorite songs is being played. Moreover, a user can set one of her playlists as a "highly recommended list" by clicking a button ②. When the selected user sets a list as "highly recommended," a song in that list is randomly but preferentially selected in the second step. By setting such a list, a user can specify the songs that she wants other users to listen to.

Note that the implemented selection algorithm described above for our service is just an example, and other algorithms can be used as long as they balance the fairness and randomness of selecting both users and songs.

In addition, we created a bot account ③ that is always logged in. The bot periodically creates playlists according to a daily/weekly popularity ranking of VOCALOID songs on Nico Nico Douga. The bot is treated as one of the users, and songs in its playlists can also be selected by the song selection process. This gives a user a chance to listen to the latest popular songs and find new favorite songs even when no other human users are logged in. Note that the bot does not show any reactions to played songs.

3. USER EXPERIENCES AND EFFECTS

As mentioned in section 1, the proposed architectures add two qualities: social connection and the joy of introducing others to favorite songs. In addition, the Kiite Cafe architectures provide three kinds of user experiences. This section describes those experiences and their effects on users.

3.1 Motivation to React to Songs

Although many studies have been conducted on enabling users to listen to music together, most of them have focused on visualizing the song selection process or propos-

ing methods for that process [7–11]. A system that can show a summary of listeners' feedback on a song (total numbers of likes and dislikes) has been proposed [12]; however, little attention has been paid to visualizing each user's reactions. In contrast, Kiite Cafe visualizes users' reactions via their icons, as in section 2.2.1. By sharing all the users' reactions with each other, Kiite Cafe motivates them to react to the currently played song. Accordingly, we expect that, the more people get together on Kiite Cafe, the more meaningful it will be to show their reactions, and the more actively they will react to songs. In the long term, this would enable users to develop the habit of actively listening to music and enrich their listening experiences [13].

3.2 Diversification of Song Listening

Many studies have sought to play songs that match the musical preferences of as many users as possible [7, 9, 14]. In the short term, this approach may be able to increase users' satisfaction. In the long term, however, as is known from the negative effects of a filter bubble [15, 16], this approach could narrow users' musical interests. On the other hand, because Kiite Cafe plays songs selected from various users' favorite songs, it may not be able to always match most users' musical preferences. However, listening to a more diverse range of songs enables users to find not only songs that match their preferences well but also unexpected or serendipitous songs [17] that do not match their usual preferences. In other words, we expect that a user will react to a more diverse range of songs on Kiite Cafe than when she listens alone. In the long term, this experience would expand the user's horizons.

3.3 Contribution as Curators

According to architecture (ii), suppose that a song in user u 's playlist is selected and played on Kiite Cafe. Because of architecture (i), u can see the moment when other users start liking or show interest in that song (e.g., u can see when other users add the song to their favorites list or rotate their icon). In substance, for other users, u plays a role as a curator. That is, the two architectures enable every user to naturally contribute as a curator. We expect that when a user experiences the joy of contributing as a curator, she will look forward to the curation opportunity when another of her favorite songs is played and thus increase her dwell time on Kiite Cafe. It has been reported that acting as a curator increases music listening activity (e.g., listening to more songs and making playlists for curation) [18]. Therefore, in the long term, this experience would promote users' daily music listening activity.

4. EXPERIMENT

We launched the Kiite Cafe service on Aug. 5, 2020. In this section, we evaluate the three expected effects discussed in the previous section. To this end, we analyzed user behavior logs for the period between Aug. 5, 2020 and Jan. 14, 2021. The number of unique users who logged in during this period was 1,760. The *Favorite*, *Comment*, *Rotation*, and *Move* reactions were used 29,127, 9,826, 59,983, and 45,353 times, respectively.

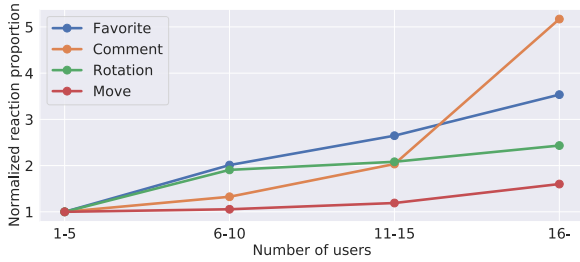


Figure 2. Relation between the number of users on Kiite Cafe and the normalized reaction proportion.

4.1 Frequency of User Reactions

As a result of users sharing their reactions with each other on Kiite Cafe, we expect that they will be more motivated to react as the number of users increases. To verify this effect, we evaluated the following research question: *Does a user react to a played song more frequently as the number of users on Kiite Cafe increases?* (**RQ1**)

Settings. We considered the four kinds of reactions: $R = \{\textit{Favorite}, \textit{Comment}, \textit{Rotation}, \textit{Move}\}$. First, for each played song, we obtained U_s , the set of users except the bot who were on Kiite Cafe when song s started playing. According to the number of users (i.e., $|U_s|$), we categorized songs into four classes (C_1 : $1 \leq |U_s| \leq 5$; C_2 : $6 \leq |U_s| \leq 10$; C_3 : $11 \leq |U_s| \leq 15$; C_4 : $16 \leq |U_s|$). To answer **RQ1**, for each reaction, we compared the average proportion of users who reacted to a song among the classes. More formally, let S_{C_i} denote a list of songs in C_i ($1 \leq i \leq 4$)⁴. Given song $s \in S_{C_i}$ and reaction $r \in R$, let U_s^r denote the set of users who gave r as a reaction to s . Then, the proportion of such users is given by $ratio(s, r) = \frac{|U_s^r|}{|U_s|}$. Finally, the average proportion over S_{C_i} was computed as follow.

$$avgratio(S_{C_i}, r) = \frac{1}{|S_{C_i}|} \sum_{s \in S_{C_i}} ratio(s, r).$$

Results. Fig. 2 shows the results. For visibility, $avgratio(S_{C_i}, r)$ was normalized by $avgratio(S_{C_1}, r)$ for each reaction. For all the reactions, the reaction proportion monotonically increased as the number of users increased; thus, the answer to **RQ1** is “Yes.” Because the *Favorite* function was obviously used to add a song to a user’s favorites list, we discuss how the users used the other three functions. Regarding the *Rotation* function, although Kiite Cafe does not explain its purpose, we searched Kiite Cafe users’ tweets on Twitter⁵ and found that a number of users used it to express their feelings of excitement. Next, by analyzing tweets about the *Move* function, we found that it was used mainly for two purposes. First, users moved their icons as if they were dancing. Second, users regarded the top left of the two-dimensional space (i.e., near the video player) as the front row at a live concert venue and moved there when their favorite songs were played. It is interesting that such a culture was created by the users and spread among them. Finally, regarding the *Comment* function,

⁴ Because the same song can be played multiple times on Kiite Cafe, the same song can appear multiple times in S_{C_i} .

⁵ We assumed that Twitter users who tweeted about the function were Kiite Cafe users.

Reaction r	$ U^r $	$avgdiv(S_u^{org})$	$avgdiv(S_u^r)$	p-value
Favorite	130	10.493	10.960	1.99×10^{-6}
Comment	56	10.384	10.920	4.40×10^{-3}
Rotation	118	10.502	10.918	5.80×10^{-6}
Move	110	10.559	11.050	8.21×10^{-9}

Table 1. Diversity of musical preferences.

although the average length of the played songs was 237 seconds, 10.1% of comments were posted within the first 15 seconds of a song. In such comments, users often expressed the joy of having their favorite songs played (e.g., “Come oooooon!” and “Yeeees!”). This was similar to the phenomenon at live concerts in which the audience gets excited when a favorite song starts. In summary, as the number of users increased, they were more likely to express their excitement and behave as if they were attending a live concert.

4.2 Diversity of Reacted Songs

Because Kiite Cafe enables users to listen to songs that do not always match their musical preferences, we expect that they will react to a more diverse range of songs. To verify this effect, we evaluated the following research question: *Does a user react to a more diverse range of songs on Kiite Cafe as compared to her musical preferences before she started using the service?* (**RQ2**)

Settings. Let t_u denote the time when user u initially logged in to Kiite Cafe. We assumed that songs added to u ’s favorites list before t_u (i.e., before using Kiite Cafe), denoted by S_u^{org} , represented u ’s original musical preferences; these were collected on the original Kiite service, which was launched on Aug. 30, 2019, and described in section 2.1. Moreover, we assumed that songs for which u gave reaction r , denoted by S_u^r , represented u ’s musical preferences in terms of r after starting to use Kiite Cafe. To answer **RQ2**, we compared the diversity of S_u^r with that of S_u^{org} for each reaction. Formally, the diversity was computed as the intra-list diversity [19]. In the case of S_u^{org} ,

$$div(S_u^{org}) = \frac{\sum_{s_i \in S_u^{org}} \sum_{s_j \in S_u^{org} \setminus \{s_i\}} dist(s_i, s_j)}{|S_u^{org}|(|S_u^{org}| - 1)},$$

where $dist(s_i, s_j)$ is the Euclidean distance between the feature vectors of s_i and s_j . For the diversity, we obtained a song’s feature vector by using OpenL3 [20]. For each reaction r , we considered only users who had more than nine songs in both S_u^{org} and S_u^r so that we could appropriately measure the users’ musical preferences⁶. Let U^r denote the set of such users. Then, given r , the average diversity of S_u^{org} was computed as

$$avgdiv(S_u^{org}) = \frac{1}{|U^r|} \sum_{u \in U^r} div(S_u^{org}).$$

Similarly, $avgdiv(S_u^r) = \frac{1}{|U^r|} \sum_{u \in U^r} div(S_u^r)$.

Results. Table 1 lists the results. We can say that for all reactions, the diversity of songs producing reactions statistically increased in comparison to the diversity of favorite songs before starting to use Kiite Cafe; thus, the answer to **RQ2** is “Yes.” These results indicate that Kiite Cafe is also

⁶ Because we released a beta version of Kiite Cafe on May 1, 2020, users who logged in to Kiite Cafe for the first time between May 1, 2020 and Aug. 4, 2020 were not included in this analysis.

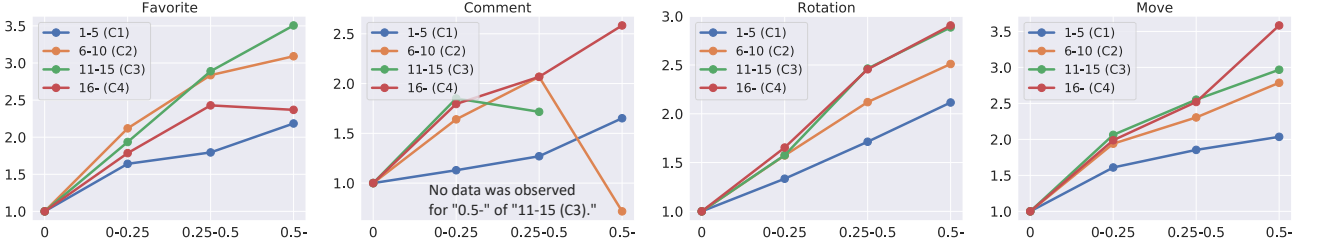


Figure 3. Relation between the proportion of users who gave reactions (x-axis) and the normalized dwell time (y-axis).

useful as a service for users to find songs that are different from their daily musical preferences.

4.3 Dwell Time

Because Kiite Cafe enables users to experience the joy of contributing as curators, we expect that they will stay longer as their contributions increase. To verify this, we evaluated the following research question: *Does a user stay on Kiite Cafe for a longer time as the proportion of users who react to her favorite songs increases?* (**RQ3**)

Settings. We define user u 's session on Kiite Cafe as the duration between u 's login and logout times. In u 's k th session, suppose that three of u 's favorite songs were played, and that 0%, 40%, and 16% of users gave reaction r to those songs. Following the assumption that the maximum percentage (in this case, 40%) influenced u 's dwell time, we categorized the maximum value of $ratio(s, r)$ (defined in section 4.1) into four classes (G_1 : $ratio(s, r) = 0$; G_2 : $0 < ratio(s, r) \leq 0.25$; G_3 : $0.25 < ratio(s, r) \leq 0.5$; G_4 : $0.5 < ratio(s, r)$). However, for a proportion of 0.4, eight reacting users among 20 users would have a higher impact on u than two reacting users among five users. Therefore, we also considered the classes of $|U_s|$ as in section 4.1. That is, to answer **RQ3**, given a class of the number of users, we compared the average session lengths between the reaction proportion classes for each reaction. Formally, let D_u denote a list of u 's sessions. The k th session $T_{u,k} \in D_u$ represents a list of songs from u 's favorite songs (S_u) that were played in the session. For that session, we selected the song $s_{u,k}^{max} \in T_{u,k}$ that had the highest proportion of users who gave reaction r (i.e., $s_{u,k}^{max} = \arg \max_{s \in T_{u,k}} ratio(s, r)$). Given C_i and G_j , we define the set of $s_{u,k}^{max}$ belonging to C_i and G_j in all users' sessions as $S_{i,j} = \{s_{u,k}^{max} | u \in U \wedge 1 \leq k \leq |D_u| \wedge s_{u,k}^{max} \in T_{u,k} \wedge s_{u,k}^{max} \in C_i \wedge s_{u,k}^{max} \in G_j\}$. Let $len(u, k)$ denote the length in seconds of u 's k th session. Then, the average session length was computed as

$$avglen(S_{i,j}) = \frac{1}{|S_{i,j}|} \sum_{s_{u,k}^{max} \in S_{i,j}} len(u, k).$$

Results. Fig. 3 shows the results; $avglen(S_{i,j})$ was normalized by $avglen(S_{i,1})$ for each reaction for visibility. For the *Favorite*, *Rotation*, and *Move* functions, we can see that the dwell time tended to increase as the proportion of users who gave that reaction increased. In these graphs, the line for the class of 1-5 users is located at the lowest position among the four classes ($C_1 - C_4$). Especially for *Rotation* and *Move*, at each reaction proportion, the normalized dwell time tended to increase with the number of

users. These results indicate that not only the proportion of users who gave a reaction but also the absolute number of such users influenced the dwell time. On the other hand, no clear tendency was observed for the *Comment* function when the number of users was 6-10 or 11-15. Still, it is possible that *Comment* also had a positive effect on the dwell time, because it monotonically increased when the number of users was 1-5 or at least 16. Detailed analysis with more user behavior logs will be required to verify this effect, and we leave that for a future work. In summary, the answer to **RQ3** is “Yes” for *Favorite*, *Rotation*, and *Move*.

5. DISCUSSION

In section 3, we described the user experiences provided by Kiite Cafe and their effects. We believe that Kiite Cafe has even more potential to diversify and enrich users' music listening experiences. In this section, to demonstrate that potential, we discuss three themes.

5.1 Application Examples for Online Events

Kiite Cafe has been used for several online events including VOCALOID-related events. At an event on Aug. 29, 2020, for example, a famous creator of VOCALOID songs made a special playlist that consisted of songs that the creator liked or had created. During the one-hour event, as many as 140 Kiite Cafe users enjoyed simultaneously listening to the songs in the playlist, and used the reaction functions of Kiite Cafe to communicate with the creator in real time. For another event on Feb. 11, 2021, a questionnaire was conducted on favorite songs related to winter or snow in the VOCALOID event. During the 90-minute event, 77 users enjoyed listening to songs in a playlist created according to the questionnaire answers.

Although it has become difficult for people to get together in person and communicate with each other and with artists because of the COVID-19 pandemic, we demonstrated a new style of online music events through these examples. Moreover, even after the pandemic's resolution, we believe that this kind of online event will be valuable for users who cannot easily attend physical events for reasons such as geographic remoteness.

5.2 Additional Service Functions

Although all users on Kiite Cafe get together in one online space, it would be interesting to provide additional spaces for different purposes in the future (we could call the main space and additional spaces the “main cafe” and “branches,” respectively). For example, for a branch on the theme of “time,” we could put a higher priority on songs related to time (e.g., playing night-related songs at night) by

analyzing song lyrics if they are available in the song selection process. We could also consider a function that allows any user to conduct a questionnaire by displaying possible responses in each quadrant of the two-dimensional space. For example, a user might ask “Who would you like to listen to the played song with?” and assign responses of “family,” “lover,” “friend,” and “other” to the quadrants. Other users could answer this question by moving their icons. This function would provide a good opportunity to see how other users perceive a song.

5.3 Reusable Insights

The reusable insights can be summarized as follows.

- Through the experiments, we verified that the two architectures are effective in promoting users’ music listening activity. These architectures can be helpful for other researchers and companies to develop interfaces that enable users to listen to music together.
- The examples of successful online events showed that the architectures can offer new ways to enjoy music with other people even during the COVID-19 pandemic. We thus opened up a new research theme to support interactions among creators, audiences, and music.
- We clarified the value of visualizing the moments when users start liking a song. In contrast to traditional curation on an SNS, the approach of Kiite Cafe guarantees that users who liked a user’s favorite song did listen to it. This insight could also be beneficial in designing other music listening systems or services.

6. RELATED WORK

6.1 Music Listening Systems for Group of Users

Music listening systems for a single user were reviewed by Goto and Dannenberg [21] and Knees *et al.* [22]. In contrast, systems for a group of users can be classified into two types. The first type aims to enable users to listen to music at the same time. Most studies on this type assume that users get together in person at a public space such as a fitness center [7], a party [10], a bar [9], or a room [8]. In MusicFX [7] and Flytrap [8], the system reads users’ musical preferences from each user’s device, and songs stored in the system are played by taking those preferences into account, while in Jukola [9], PartyVote [10], and WePlay [12], users nominate songs to be played, like a jukebox. In the second type of group listening system, users share songs with other users. Sharing music with others is an important activity to expand listeners’ horizons [4]. Studies on this type do not assume that users listen to a song at the same time. Push!Music [4] and tunA [3] are mobile music players that let users share songs via Wi-Fi with others who are nearby. The user studies on those systems showed that users are comfortable sharing their favorite songs with others whether they are friends or strangers. It has also been reported that users share songs mainly because they want to recommend songs that others would like, disseminate their favorite songs, talk about shared songs with others, and so on.

Some applications designed for listening to music together have also been released (e.g., Group Session by

Spotify [23] and JQBX [24]). In these applications, any user can let other users listen to her favorite songs by acting like a DJ. Users can also communicate with each other via a text chat system while listening to songs.

Our study is different from the above studies and services in that we introduce the two architectures for reaction visualization and song selection from users’ favorite songs. In most of the above cases, because users’ reactions are not visualized or are visualized only when chatting with text messages, it is difficult for users to feel social connection with each other. On the other hand, because the first architecture on Kiite Cafe visualizes four kinds of reactions, users can more strongly feel that they are enjoying music with others. In addition, existing systems require users to actively nominate or share songs or act like a DJ, but some users may hesitate to do that, especially if there is a large audience. In contrast, the second architecture on Kiite Cafe enables a user’s favorite songs to be automatically played. This lets any user share her favorite songs with other users and see the moment when they start liking those songs.

6.2 Group Recommendation Algorithms

Various song recommendation methods for a single user have been proposed [25–32]. One of the biggest differences between the methods for a single user and those for a group of users is that the latter methods need to take multiple users’ preferences into account. A general approach is to aggregate each user’s preferences by, for example, merging recommendation results generated for each user according to voting strategies [14, 33]. However, such an approach cannot always reflect minority preferences.

To solve this problem, a concept of *fairness* has been recently introduced into group recommendation algorithms [34–38]. The basic idea of fairness is that a list of items recommended to a group is fair when each user in the group can find at least one item in the list that she finds satisfying. In the context of music recommendation, existing studies have only considered the fairness for users as audiences. On the other hand, fairness for users as curators as well as audiences is achieved by Kiite Cafe because of the second architecture in which each user’s favorite songs are fairly selected and played as described in section 2.2.2. In particular, the “highly recommended list” plays an important role in achieving fairness as curators. When a user’s favorite and/or recommended song can be listened to with other users, the user is satisfied from audience and curator viewpoints.

7. CONCLUSION

In this paper, we described Kiite Cafe, a web service that enables users to communicate while listening to the same music online. Kiite Cafe is characterized by two proposed architectures for visualizing each user’s reactions and selecting played songs from users’ favorite songs. Our experimental results quantitatively showed three effects provided by the proposed architectures. We believe that these architectures are also useful for different types of music interfaces, including a three-dimensional interface where user avatars could listen to the same music in a virtual reality (VR) venue.

8. ACKNOWLEDGMENTS

We thank Crypton Future Media, INC. for developing Kite with us and Nico Nico Douga for initially tacitly (later explicitly) encouraging us to work on this research. We also would like to extend our appreciation to users of Kite and Nico Nico Douga, creators of VOCALOID songs, creators of popularity rankings of VOCALOID songs, and all people who have created, supported, and enjoyed the VOCALOID culture. This work was supported in part by JST ACCEL Grant Number JPMJAC1602, JST CREST Grant Number JPMJCR20D4, and JSPS KAKENHI Grant Number 20K19934, Japan.

9. REFERENCES

- [1] K. Sedgman, “Coughing and clapping: investigating audience experience,” *Cultural Trends*, vol. 24, no. 4, pp. 324–326, 2015.
- [2] S. C. Brown and D. Knox, “Why go to pop concerts? The motivations behind live music attendance,” *Musicae Scientiae*, vol. 21, no. 3, pp. 233–249, 2017.
- [3] A. Bassoli, J. Moore, S. Agamanolis, and H. C. Group, “tunA: Local music sharing with handheld Wi-Fi devices,” in *Proceedings of the 5th Wireless World Conference 2004*, ser. WWC ’04, 2004.
- [4] M. Håkansson, M. Rost, and L. E. Holmquist, “Gifts from friends and strangers: A study of mobile music sharing,” in *Proceedings of the 10th European Conference on Computer-Supported Cooperative Work*, ser. ECSCW ’07, 2007, pp. 311–330.
- [5] M. Håkansson, M. Rost, M. Jacobsson, and L. E. Holmquist, “Facilitating mobile music sharing and social interaction with Push!Music,” in *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, ser. HICSS ’07, 2007, pp. 87–96.
- [6] H. Kenmochi and H. Ohshita, “VOCALOID - commercial singing synthesizer based on sample concatenation,” in *Proceedings of the 8th Annual Conference of the International Speech Communication Association*, ser. INTERSPEECH ’07, 2007, pp. 4009–4010.
- [7] J. F. McCarthy and T. D. Anagnost, “MusicFX: An arbiter of group preferences for computer supported collaborative workouts,” in *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW ’98, 1998, pp. 363–372.
- [8] A. Crossen, J. Budzik, and K. J. Hammond, “Flytrap: Intelligent group music recommendation,” in *Proceedings of the 7th International Conference on Intelligent User Interfaces*, ser. IUI ’02, 2002, pp. 184–185.
- [9] K. O’Hara, M. Lipson, M. Jansen, A. Unger, H. Jeffries, and P. Macer, “Jukola: Democratic music choice in a public space,” in *Proceedings of the 5th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, ser. DIS ’04, 2004, pp. 145–154.
- [10] D. Sprague, F. Wu, and M. Tory, “Music selection using the PartyVote democratic jukebox,” in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI ’08, 2008, pp. 433–436.
- [11] G. Popescu and P. Pu, “What’s the best music you have? Designing music recommendation for group enjoyment in GroupFun,” in *Proceedings of the CHI ’12 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA ’12, 2012, pp. 1673–1678.
- [12] F. Vieira and N. Andrade, “Evaluating conflict management mechanisms for online social jukeboxes,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ser. ISMIR ’15, 2015, pp. 190–196.
- [13] M. Goto, “Active music listening interfaces based on signal processing,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, ser. ICASSP ’07, 2007, pp. IV-1441–IV-1444.
- [14] M. Kompan and M. Bielikova, “Group recommendations: Survey and perspectives,” *Computing and Informatics*, vol. 33, no. 2, pp. 446–476, 2014.
- [15] E. Pariser, *The filter bubble: What the Internet is hiding from you*. Penguin Press, 2011.
- [16] M. Taramigkou, E. Bothos, K. Christidis, D. Apostolou, and G. Mentzas, “Escape the bubble: Guided exploration of music preferences for serendipity and novelty,” in *Proceedings of the 7th ACM Conference on Recommender Systems*, ser. RecSys ’13, 2013, pp. 335–338.
- [17] Y. C. Zhang, D. O. Séaghdha, D. Quercia, and T. Jambor, “Auralist: Introducing serendipity into music recommendation,” in *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, ser. WSDM ’12, 2012, pp. 13–22.
- [18] J. Fuller, L. Hubener, Y. Kim, and J. H. Lee, “Elucidating user behavior in music services through persona and gender,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, ser. ISMIR ’16, 2016, pp. 626–632.
- [19] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, “Improving recommendation lists through topic diversification,” in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW ’05, 2005, pp. 22–32.
- [20] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, listen, and learn more: Design choices for deep audio embeddings,” in *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*, ser. ICASSP ’19, 2019, pp. 3852–3856.

- [21] M. Goto and R. B. Dannenberg, “Music interfaces based on automatic music signal analysis: New ways to create and listen to music,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 74–81, 2019.
- [22] P. Knees, M. Schedl, and M. Goto, “Intelligent user interfaces for music discovery,” *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 165–179, 2020.
- [23] “Group session - Spotify,” <https://support.spotify.com/us/article/group-session/>.
- [24] “JQBX - Listen Together. DJ Online. Discover New Music.” <https://www.jqbx.fm/>.
- [25] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. G. Okuno, “Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences,” in *Proceedings of the 7th International Conference on Music Information Retrieval*, ser. ISMIR ’06, 2006, pp. 296–301.
- [26] M. Tiemann, S. Pauws, and F. Vignoli, “Ensemble learning for hybrid music recommendation,” in *Proceedings of the 8th International Conference on Music Information Retrieval*, ser. ISMIR ’07, 2007, pp. 179–180.
- [27] K. Yoshii and M. Goto, “Continuous pLSI and smoothing techniques for hybrid music recommendation,” in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, ser. ISMIR ’09, 2009, pp. 339–344.
- [28] Z. Xing, X. Wang, and Y. Wang, “Enhancing collaborative filtering music recommendation by balancing exploration and exploitation,” in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ser. ISMIR ’14, 2014, pp. 445–450.
- [29] A. Vall, M. Skowron, P. Knees, and M. Schedl, “Improving music recommendations with a weighted factorization of the tagging activity,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ser. ISMIR ’15, 2015, pp. 65–71.
- [30] D. Liang, M. Zhan, and D. P. W. Ellis, “Content-aware collaborative music recommendation using pre-trained neural networks,” in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ser. ISMIR ’15, 2015, pp. 295–301.
- [31] R. S. Oliveira, C. Nóbrega, L. B. Marinho, and N. Andrade, “A multiobjective music recommendation approach for aspect-based diversification,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR ’17, 2017, pp. 414–420.
- [32] O. Gouvert, T. Oberlin, and C. Févotte, “Matrix co-factorization for cold-start recommendation,” in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, ser. ISMIR ’18, 2018, pp. 792–798.
- [33] L. Baltrunas, T. Makcinskas, and F. Ricci, “Group recommendations with rank aggregation and collaborative filtering,” in *Proceedings of the Fourth ACM Conference on Recommender Systems*, ser. RecSys ’10, 2010, pp. 119–126.
- [34] S. Qi, N. Mamoulis, E. Pitoura, and P. Tsaparas, “Recommending packages to groups,” in *Proceedings of the IEEE 16th International Conference on Data Mining*, ser. ICDM ’16, 2016, pp. 449–458.
- [35] D. Serbos, S. Qi, N. Mamoulis, E. Pitoura, and P. Tsaparas, “Fairness in package-to-group recommendations,” in *Proceedings of the 26th International Conference on World Wide Web*, ser. WWW ’17, 2017, pp. 371–379.
- [36] L. Xiao, Z. Min, Z. Yongfeng, G. Zhaoquan, L. Yiqun, and M. Shaoping, “Fairness-aware group recommendation with pareto-efficiency,” in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys ’17, 2017, pp. 107–115.
- [37] D. Sacharidis, “Top-N group recommendations with fairness,” in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC ’19, 2019, pp. 1663–1670.
- [38] M. Stratigi, J. Nummenmaa, E. Pitoura, and K. Stefanidis, “Fair sequential group recommendations,” in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, ser. SAC ’20, 2020, pp. 1443–1452.

TOWARD AN UNDERSTANDING OF LYRICS-VIEWING BEHAVIOR WHILE LISTENING TO MUSIC ON A SMARTPHONE

Kosetsu Tsukuda Masahiro Hamasaki Masataka Goto

National Institute of Advanced Industrial Science and Technology (AIST), Japan

{k.tsukuda, masahiro.hamasaki, m.goto}@aist.go.jp

ABSTRACT

Why and how do people view lyrics? Although various lyrics-based systems have been proposed in MIR community, this fundamental question remains unexplored. Better understanding of lyrics viewing behavior would be beneficial for both researchers and music streaming platforms to improve their lyrics-based systems. Therefore, in this paper, we investigate why and how people view lyrics, especially when they listen to music on a smartphone. To answer “why,” we conduct a questionnaire-based online user survey involving 206 participants. To answer “how,” we analyze over 23 million lyrics request logs sent from the smartphone application of a music streaming service. Our analysis results suggest several reusable insights, including the following: (1) People have high demand for viewing lyrics to confirm what the artist sings, more deeply understand the lyrics, sing the song, and figure out the structure such as verse and chorus. (2) People like to view lyrics after returning home at night and before going to sleep rather than during the daytime. (3) People usually view the same lyrics repeatedly over time. Applying these insights, we also discuss application examples that could enable people to more actively view lyrics and listen to new songs, which would not only diversify and enrich people’s music listening experiences but also be beneficial especially for music streaming platforms.

1. INTRODUCTION

When people seek help in identifying a particular song that they have listened to, they often provide words in the song’s lyrics as a clue for identification [1,2]. In other situations when people listen to music, it has been reported that they choose songs according to not only the musical audio content, such as the music genre, mood, melody, vocal timbre, and rhythm, but also the topics of lyrics [3,4]. To meet these demands, in the field of Music Information Retrieval (MIR), researchers have proposed systems for identifying a song by using the words in lyrics as a query [5–8] and systems for exploring songs according to the topics estimated from lyrics [9–12]. As illustrated here, lyrics are

an essential element of music for both listeners and MIR researchers.

Despite the importance of lyrics in the MIR community, more fundamental investigation of lyrics remains an under-addressed topic: why and how do people view lyrics? In this paper, we aim to answer these questions. Investigating people’s lyrics-viewing behavior and revealing reusable insights would be beneficial for researchers and music streaming platforms to implement lyrics-related systems and functions, such as viewing support for lyrics and song recommendation based on lyrics. With regard to music listening, researchers have investigated why and how people listen to music [13–20], and the obtained insights have contributed to later studies in the MIR community. Although listening to music includes listening to sung lyrics, our study differs from these studies in that we focus on lyrics-viewing behavior.

Users can view lyrics in various ways, such as a lyrics sheet included with a compact disc (CD), a web service for lyrics search, and a YouTube video with lyrics overlaid [21]. Recently, some smartphone applications for online music services (e.g., Spotify and Apple Music) have provided a function that enables a user to view song’s lyrics while listening to the song. Such a function will become one of the main means for viewing lyrics, given the current situation in which music streaming services on smartphones have become a mainstream format for listening to music [15]. In light of the above, we investigate the behavior of viewing lyrics on a smartphone while listening to music, because we can make the obtained insights more reusable for future work in MIR community.

Our main contributions can be summarized as follows:

- To our knowledge, this is the first study on the interactions between users and lyrics in terms of why and how users view lyrics when they listen to music.
- To investigate why users view lyrics, we conducted a large-scale questionnaire-based online survey involving 206 participants. In the survey, more than 75% of the participants answered that they often view lyrics to confirm what an artist sings or more deeply understand lyrics. Moreover, over 50% of the participants often view lyrics to sing a song or figure out the structure of the lyrics (verse, chorus, etc.). These results are beneficial for both MIR researchers and music streaming platforms to implement their systems or functions. In fact, in this paper, we suggest examples of functions to support



© Kosetsu Tsukuda, Masataka Goto. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Kosetsu Tsukuda, Masataka Goto, “Toward an Understanding of Lyrics-viewing Behavior While Listening to Music on a Smartphone”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf., Online*, 2021.

users according to their reasons for viewing lyrics, such as a function that displays tips to sing each part of the song’s lyrics for users who want to sing.

- We investigated how users view lyrics by analyzing over 23 million lyrics request logs for over 600 thousand smartphone users for a year on a music streaming service. The data shows that people tend to view more lyrics after coming back home at night and before going to bed. In addition, an average of 37.8% of user’s viewed lyrics have already been viewed by the user, and eventually the user gets bored with viewing the same lyrics. Considering these findings, we make several proposals for music streaming platforms to attract users (e.g., when a user gets bored with the lyrics of a song, the platform could suggest related lyrics in terms of the topic).

2. RELATED WORK

2.1 User Behavior in Music Listening

One approach to analyze user behavior in music listening is conducting user studies based on questionnaires and interviews. Typical questions about music listening ask why people listen to music [16–18, 20] and how they use music websites, services, and applications [1, 15, 22]. Regarding the former question, the main reasons include emotional reasons such as relaxation [17] (even at work [18]) and relief [16]. People also listen to music to concentrate and to pass time [20]. Regarding the latter question, Lee and Waterman [15] revealed that people use music websites and applications for various reasons such as discovering new music and learning about the artists. They also compared their results with those in 2004 [1] and showed increases in the popularity of music streaming and mobile music consumption. A more recent work conducted a survey on the use of cloud music services and considered the future design of such services [22]. Moreover, Lee and Price [14] conducted interviews with music listeners and revealed seven typical personas, such as a user who enjoys curating music that is already familiar and a user who enjoys serendipitous music discovery.

Another approach is analyzing users’ play logs. These logs are typically collected from (1) APIs provided by online music services [23, 24] or (2) Twitter, where tweets related to music listening are gathered via specific tags such as “#nowplaying” and “#itunes” [25–27]. Logs have been analyzed in terms of various aspects, including the long tail distribution of listening events per user, track, and artist [23, 24, 27], the popularity of genres, moods, and tags [25–27], and the temporal distribution (hour of day and day of week) [23, 26], etc. One characteristic of music listening behavior is *repeat consumption* [28]. Reports have indicated that, in a user’s music play logs, about 70% of played songs have already been played before, and this percentage is much higher than for other domains such as viewing videos and visiting restaurants [28, 29]. In repeat consumption, the number of times a song is played is heavy-tailed (i.e., a user repeatedly listens to a small proportion of songs again and again). Benson *et al.* [29] re-

ported that each song has its own *lifetime* for a user: at the beginning of the lifetime, the temporal gap between listening events is small; but at the end of the lifetime, the gap becomes large, and eventually the user becomes bored with the song.

Although listening to sung lyrics is one factor in listening to music, our study differs from the above studies in that we particularly focus on lyrics viewing behavior. Focusing on a particular element of music is beneficial to suggest new possibilities for future research as was indicated by Demetriou *et al.* [4] who focused on vocals. Research on *why* people listen to music has tended to involve user studies, because they have the advantage of enabling researchers to ask questions to analyze people’s intent. In contrast, research on *how* people listen to music has often analyzed large log data to take advantage of statistical processing. Applying both of these advantages, in this paper, we investigate why and how people view lyrics by using questionnaires and logs, respectively.

2.2 Lyrics in MIR

Researchers have considered lyrics in various studies, including lyrics-to-audio alignment [30–36], analysis of lyrics characteristics [37–42], accurate lyrics retrieval [43–45], and genre and mood classification [46–51]. Below, we review more related studies that aim to support user activity by using lyrics.

One major approach is enabling users to search for songs by words in lyrics, in which a query can be text [5, 6] or user’s sung lyrics [7, 8]. Systems have also been proposed for exploring songs according to topics estimated from lyrics [9–12]. Fujihara *et al.* [52] proposed the concept of a “Music Web” in which songs are hyperlinked to each other based on phrases of lyrics. Visualization is also a useful approach to browse a music collection. *SongWords* [53] displays a music collection on a two-dimensional canvas based on self-organizing maps for lyrics and tags. *Lyricon* [54] is a system for displaying icons that match the word sequences of lyrics so that users can intuitively understand the lyrics. Moreover, Funasawa *et al.* [55] implemented a system that automatically generates slideshows for music by generating queries from lyrics and searching for images. O’Hara *et al.* [56] demonstrated how to learn the meanings of chord sequences from lyrics annotated with chords. Ibrahim *et al.* [57] proposed a method for estimating the intelligibility of lyrics in a given song to help users learn a second language.

In this paper, we investigate more fundamental questions about lyrics: why and how people view them. For researchers, the insights of our analysis can be used in implementing lyrics-based systems. For example, when researchers propose systems to support understanding lyrics, they can claim these systems’ importance based on the high demand for deeply understanding lyrics, as we will report in Section 3.2.1. In Sections 3 and 4, we also suggest application examples such as recommending songs according to lyrics and supporting lyrics viewing. We believe that our suggestions are also beneficial for music streaming platforms to make their smartphone applications more

attractive to users.

3. WHY PEOPLE VIEW LYRICS

In this section, we report why people view lyrics by conducting an online survey involving 206 participants.

3.1 Participants

We recruited participants for our survey via an online research company. We limited the participants to those who listen to music on average at least one day per week on a smartphone application via any online music service and have viewed lyrics on the application while listening to music at least 10 times in their lifetime. In addition, to align with the user nationality in the lyrics viewing log data, as described in Section 4.1.1, all participants were Japanese. The participants answered our questionnaire through a web browser. We paid about 21.1 USD (2,275 JPY) to each participant. Although 297 participants joined the survey, to make the analysis results more reliable, we removed the answers from 91 participants: 14 of them gave the same answers to all questions (e.g., choosing “1” for all questions), and 77 of them finished answering the questions in a very short time¹. The remaining 206 participants were well balanced in gender and age range: 95 males (10s: 2; 20s: 20; 30s: 22; 40s: 26; 50s: 25) and 111 females (10s: 4; 20s: 21; 30s: 27; 40s: 28; 50s: 31).

3.2 Results and Discussion

3.2.1 Reasons

To understand why people view lyrics on a smartphone while listening to music, we listed the following eight candidate reasons². (1) *Confirmation*: The user wants to confirm what the artist sings. (2) *Understanding*: The user wants to more deeply understand the lyrics. (3) *Singing*: The user wants to sing to herself (not in public). (4) *Structure*: The user wants to figure out the structure of the lyrics, such as verse and chorus. (5) *Karaoke*: The user wants to practice for singing in public, as in karaoke. (6) *Boredom*: The user wants to get rid of her boredom by viewing lyrics. (7) *Language*: The user wants to learn a language with the lyrics. (8) *Writing*: The user wants to study for writing lyrics. The participants were asked to rate the frequency of viewing lyrics for each reason on a scale of 1 to 5 (1: never; 5: very often). The reasons were displayed in a random order to each participant³.

For each reason, Figure 1 shows the frequency distribution and the number of users whose score was 4 or 5 (i.e., the number who often viewed lyrics for that reason). We can see that the ratings for *Confirmation* and *Understanding* are high: in fact, the paired Wilcoxon signed-

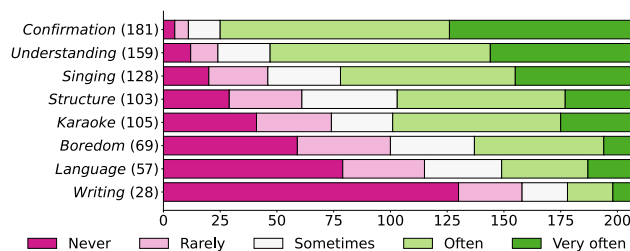


Figure 1. Frequency of reasons why people view lyrics on a smartphone while listening to music (1: never; 5: very often). The number in parentheses represents the number of participants rating 4 or 5.

rank tests with Bonferroni correction reveal that the medians of *Confirmation* and *Understanding* are statistically higher than the remaining six reasons at $p < 0.01$. It would be beneficial to provide additional functions according to users’ reasons for viewing lyrics. For example, for a user whose reason is *Understanding*, displaying diverse interpretations of lyrics could help her understand them more deeply. An interesting future work would be to automatically mine web pages that describe interpretations of given song’s lyrics and display the collected interpretations along with the lyrics.

Among the remaining six reasons, more than half of the participants gave a rating of 4 or 5 for *Singing*, *Structure*, and *Karaoke*. For users who view lyrics to sing (*Singing* and *Karaoke*), some smartphone applications already provide a function that automatically scrolls lyrics by synchronizing them with the playback time. To improve their singing performance, we suggest more advanced functions that display tips for singing each part of the lyrics and automatically judge their singing skill [58]. As for the *Structure* reason, one possible application is coloring blocks of lyrics according to the estimated structure [59, 60]; this would enable the user to quickly figure out the structure.

Although *Boredom*, *Language*, and *Writing* are relatively minor reasons, it is still worth considering functions for them, not only because it is important to build systems to support niche uses but also because more users may begin to view lyrics to use such functions. This may give users chances to listen to music more frequently and eventually provide benefits for music streaming platforms. For a user who views lyrics because of *Boredom*, displaying information related to the played song, such as similar songs by different artists, may help her discover unfamiliar songs. When a user views lyrics for learning (*Language* and *Writing*), she may want to use functions that improve the efficiency of the learning process. Examples for *Language* include enabling the user to see the meaning of a word in lyrics just by tapping the word and recommending a song by the same artist with more intelligible lyrics [57]. Examples for *Writing* include explaining poetic and rhetorical techniques used in writing lyrics and recommending songs with the same techniques.

Finally, Table 1 lists the number of participants who gave a rating of 4 or 5 to at least k reasons. Because 89.8% of the participants gave high scores for more than one reason and over 60% of them often view lyrics for more than

¹ We applied a tight rule for this filtering to reduce the risk of noisy answers as much as possible. Nonetheless, the remaining 206 participants are sufficient to discuss the general tendency of people’s behavior [15].

² The eight candidate reasons were decided through discussions among the authors.

³ We also provided an open-ended answer format for asking the participants to freely describe other reasons. However, only three participants used it; their answers are omitted here due to space limitations. We therefore think that the eight candidate reasons covered the possible reasons well. Using a fully open-ended answer format to compare results could be an interesting future work.

Table 1. Number of participants who gave a rating of 4 or 5 to at least k reasons.

k	1	2	3	4	5	6	7	8
#participants	196	185	168	125	87	39	22	2
Percentage	95.1	89.8	81.6	60.7	42.2	18.9	10.7	3.88

three reasons, we can say that the reason for viewing lyrics is not exclusive; rather, it is common to have multiple reasons. An interesting future work would be to predict and recommend lyrics-related functions (like those described above) to use next according to those already used.

3.2.2 Behavior

We now investigate users’ detailed behavior in viewing lyrics for different reasons in terms of three aspects. Note that, for each reason, we asked follow-up questions to participants who gave a rating of 4 or 5 so that we could interpret the characteristics of the reasons more accurately (see Figure 1 for the number of such participants for each reason).

Aspect 1: timing. First, to each participant, we showed a reason for which she gave a rating of 4 or 5 and asked, “When you view lyrics for this reason, do you decide to do so (a) before playing a song or (b) after playing a song?” The possible answers were (1) mostly (a), (2) moderately (a), (3) about the same, (4) moderately (b), and (5) mostly (b). Answers (1) and (2) ((4) and (5)) were then merged into a “Before” (“After”) group. The “Timing” column of Table 2 lists the frequency of responses in each group for each reason. For *Structure* that has a statistically high frequency in the “After” group, it would be effective to enable users to more quickly execute the corresponding function proposed in Section 3.2.1 while listening to a song, as compared to the functions for other reasons. On the other hand, *Karaoke* has a statistically high frequency in the “Before” group. Therefore, if a smartphone application provided an option to play a song in the setting of the *Karaoke* function explained in Section 3.2.1, users would be expected to use the application more frequently to practice for karaoke. In Table 2, although both *Singing* and *Karaoke* are related to singing a song, it is interesting that *Singing* has almost the same frequencies in the “Before” and “After” groups.

Aspect 2: repetition. Our next question was “When you view lyrics for this reason, how many times do you continuously view them while repeatedly playing a song?” The answers consisted of (1) mostly once (i.e., no repetition), (2) mostly two or three times, and (3) mostly more than three times. Because no significant difference was observed between answers (2) and (3), we report the results with answers (2) and (3) merged as a “Many” group, while answer (1) is labeled as “Once.” The “Repetition” column of Table 2 lists the results. It can be observed that, for all reasons, the “Many” group has higher frequency. It is thus common behavior to continuously view lyrics while repeating a song. Therefore, it would be helpful for users to change the displayed information according to the number of repetitions (e.g., when a user listens to a song for the *Understanding* reason, different interpretations of the lyrics can be shown every time she plays it).

Table 2. Behavior frequency in terms of three aspects: timing, repetition, and percentage.

Reason	Timing		Repetition		Percentage	
	Before	After	Once	Many	Partial	Most
<i>Confirmation</i>	49	95**	70	111**	53	84**
<i>Understanding</i>	60	70	38	121**	20	116**
<i>Singing</i>	50	51	36	92**	16	85**
<i>Structure</i>	29	46*	33	70**	18	57**
<i>Karaoke</i>	55*	33	14	91**	13	78**
<i>Boredom</i>	12	39**	29	40	19	31
<i>Language</i>	27	18	12	45**	2	39**
<i>Writing</i>	11	10	3	25**	2	17**

* (**) denotes the statistical difference at $p < 0.05$ ($p < 0.01$) based on a two-tailed z-test.

Aspect 3: percentage. In our last question, we asked, “When you view the lyrics for this reason, what percentage of the lyrics do you view?” The answers were (1) $\leq 20\%$, (2) 21%–40%, (3) 41%–60%, (4) 61%–80%, and (5) $\geq 81\%$. We merged answers (1) and (2) ((4) and (5)) into a “Partial” (“Most”) group. The “Percentage” column of Table 2 lists the frequency of responses in each group. Because “Most” was more popular for all reasons, people tend to view most of the lyrics in any situation. However, a significant difference between “Partial” and “Most” was not observed for *Boredom* only. This result indicates that when a user stops viewing lyrics within a short time, she is likely bored. Therefore, music streaming platforms have a big opportunity to give such users valuable information, as illustrated in Section 3.2.1.

4. HOW PEOPLE VIEW LYRICS

In this section, we report how people view lyrics based on over 23 million lyrics request logs sent from smartphone applications for playing music.

4.1 Dataset

4.1.1 Lyrics Viewing Log

For lyrics viewing, we used log data given by a lyrics distribution company (SyncPower Corporation) in Japan. Although this company provides lyrics text to various music-listening smartphone applications, we focused on the iOS application of a Japanese online music service and used logs collected from it. In the application, a user can view the lyrics of a played song while listening to the song. The application gets the lyrics by using an API provided by the lyrics distribution company. The company stores request logs that include the timestamp, user ID, and song ID. Note that the application does not automatically get lyrics when a song is played; rather, it only gets them when a user explicitly requests them. Therefore, the logs are suitable for analyzing how users view lyrics.

We first collected logs whose timestamp was between 1/1/2018 and 12/31/2018. We then removed logs whose duration was less than 30 seconds, because such short-term logs may have resulted from users’ wrong operations. Finally, our dataset (hereafter, *LyLog*) consisted of 611,895 users, 214,434 unique songs, and 23,034,417 logs.

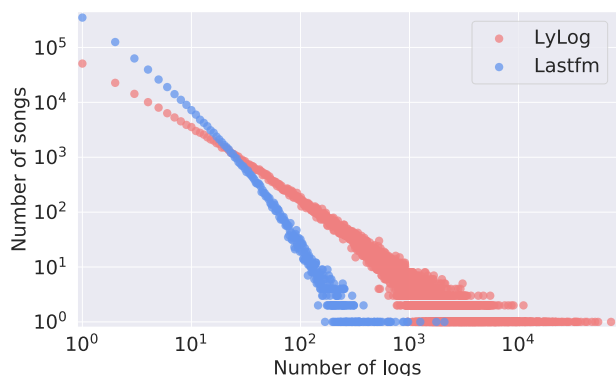


Figure 2. Distribution of the number of logs per song. There are y songs that have x logs.

4.1.2 Music Listening Log

To investigate the difference between lyrics viewing behavior and music listening behavior, we used the Last.fm dataset released by Schedl [23]. This dataset consists of users’ play logs, each of which includes the timestamp, user ID, song ID, and artist ID. To align the users’ nationality with the *LyLog* dataset, we first extracted logs of Japanese users (the dataset also includes each user’s nationality). We then collected logs whose timestamp was between 1/1/2013 and 12/31/2013 and removed logs whose duration was less than 30 seconds. This gave us a music listening dataset (hereafter, *Lastfm*) consisting of 660 users⁴, 718,466 unique songs, and 2,932,430 logs⁵.

We do acknowledge some limitations of using *Lastfm* for comparison. For example, the years in *Lastfm* are different from those in *LyLog*, and *Lastfm* includes play histories from not only smartphones but also PCs. Therefore, it should be noted that the purpose of the comparison in this paper is not to provide generalizable insights about the differences between lyrics viewing and music listening. Nonetheless, we think it is still worth comparing the differences as a first step toward understanding the characteristics of lyrics viewing behavior. We leave it as a future work to compare lyrics viewing logs and music listening logs from the same platform⁶.

4.2 Basic Statistics

We first investigated several basic characteristics of lyrics viewing. Figure 2 shows the distribution of the number of consumption logs per song⁷. Although the curves of both *LyLog* and *Lastfm* show the heavy tail of their consumption patterns, lyrics viewing behavior is more biased to popular songs: in *Lastfm*, 80% of the whole logs are dominated by the top 34.8% of the songs in terms of the number of logs, while in *LyLog*, those are dominated by only the top 6.64% of the songs.

⁴ There is no correspondence between the users in *LyLog* and those in *Lastfm*.

⁵ A similar Last.fm dataset was released more recently [24], but the included logs are older than those in Schedl’s dataset [23]. Therefore, we decided to use the latter dataset.

⁶ We cannot do so in this paper because the lyrics distribution company mentioned in Section 4.1.1 cannot store music play logs that do not include lyrics requests.

⁷ Throughout our investigation, the word “consumption” refers to viewing lyrics in *LyLog* or listening to music in *Lastfm*.

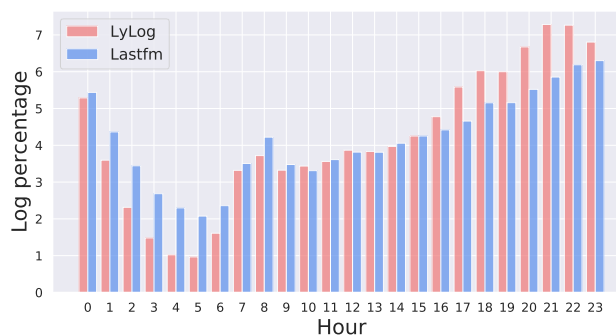


Figure 3. Distribution of logs over the hours of the day.

In Figure 3, we show the distribution of logs over the hours of the day⁸. According to a survey on time use by the Statistics Bureau of Japan [61], the average Japanese person gets up at 6:32 am, commutes to school or work between 7:30 am and 8:30 am, commutes from school or work between 6:00 pm and 7:00 pm, and goes to sleep at 11:15 pm. Referring to this time schedule, we can see some common characteristics in both datasets: the number of logs increases during the morning commute and after returning home in the evening; then, the number gradually decreases as people go to sleep. Between 5:00 pm and 11:59 pm, however, *LyLog* has a higher percentage than *Lastfm* does. Viewing lyrics on a smartphone requires users to interact with the application more actively, as in tapping the screen to request and look at lyrics; in contrast, users can listen to songs even with a smartphone in a pocket. Therefore, we can guess that users often view lyrics in a relaxed state after coming back home. When a smartphone application recommends some of the functions described in Section 3.2.1 to a user, night would be a more suitable time, because the user would engage more actively in viewing lyrics than during the daytime: it would be an interesting future work to verify the usefulness of changing the recommendation frequency of each function according to time. Regarding the distribution of logs over the days of week, although we do not show a chart due to the space limitation, people view lyrics and listening to music 6.64% and 6.53% more often on weekends than on weekdays, respectively; and no significant difference was observed between the datasets.

4.3 Repeat Consumption

We next investigated repeat consumption behavior in which a user consumes the same song repeatedly over time. We first computed the fraction of repeat consumption for each user. For example, if a user’s fraction is 0.4, then 40% of viewed lyrics have been already viewed by her. Figure 4 shows this fraction’s distribution. It can be observed that the fraction for *LyLog* tends to be lower than that for *Lastfm*; in fact, the average fractions for *LyLog* and *Lastfm* are 0.378 and 0.604, respectively. However, we can say that the fraction of repeat consumption for lyrics viewing is still high compared to that of other domains such as watching videos (fraction: 0.26) and clicking on English Wikipedia pages (fraction: 0.15) [29]. The above analysis

⁸ Note that Japan does not observe daylight saving time.

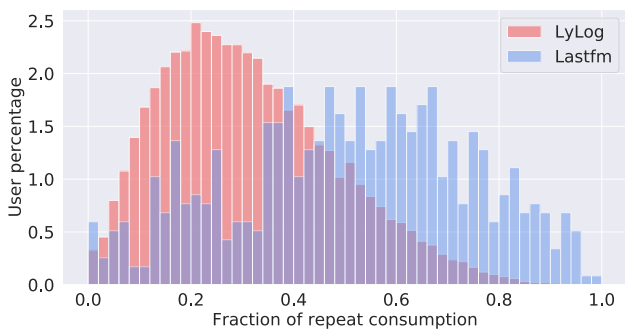


Figure 4. Fraction of repeat consumption for each user.

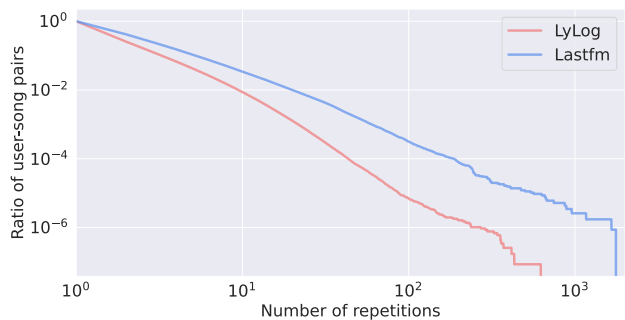


Figure 5. Distribution of the ratio of user-song pairs that are repeatedly consumed x or more times.

did not consider how many times each song was repeatedly consumed. Thus, we also computed the ratio of user-song pairs, in which each song was repeatedly consumed x or more times by each user, to all user-song pairs, as shown in Figure 5. We can see that the numbers of repetitions for both *LyLog* and *Lastfm* have a heavy tail. However, because the *LyLog* curve is located below and to the left of the *Lastfm* curve, people do not repeatedly view the same lyrics as many times as they listen to the same song.

Benson *et al.* [29] reported that, in repeat consumption, each item has its own lifetime for a user, as described in Section 2.1. Following their processes, we investigated the lifetime characteristics of lyrics viewing as follows. Given a user, we first sorted all songs for which she requested lyrics in ascending order of the timestamp. We then extracted songs whose first and last consumption events were in the middle 60% of the list, so that we could consider songs that certainly began and ended their lifetimes during the period of data collection. Suppose that a user’s extracted consumption list consists of N songs and is represented by $L = \{i_1, \dots, i_N\}$. When a particular song s is consumed k times at indices $\{i_1^s, \dots, i_k^s\} \in L$, the index gap between the j th and $j + 1$ th consumption events is defined by $g_j = i_{j+1}^s - i_j^s$. Figure 6 shows the transition of the mean gap, with all gaps normalized by the first gap g_1 (the average values of g_1 for *LyLog* and *Lastfm* were 19.0 and 248, respectively). As in the report by Benson *et al.* [29], in lyrics viewing behavior, too, the gap tends to grow over time. This means that when a user repeatedly views the lyrics of a song, she views it again within a short span at the beginning; the span gradually increases as she gets bored with it, and eventually she stops viewing the lyrics. As can be seen in Figure 6, the gap increase rate for *LyLog* was smaller than that for *Lastfm*.

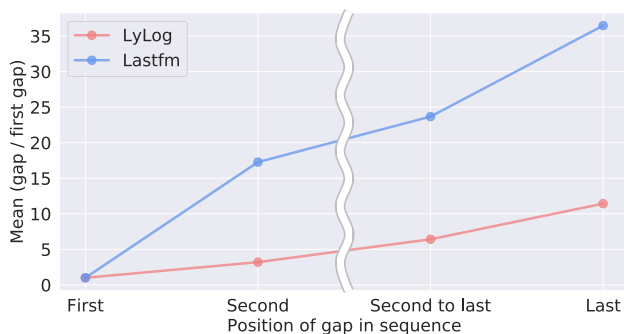


Figure 6. Normalized mean index gaps.

Because the gap grows over time, there is a possibility that we can detect a user who begins to get bored with particular lyrics by using the method proposed by Benson *et al.* [29]. When such a user is detected, suggesting functions (from those described in Section 3.2.1) that she has not used for the lyrics is one possible way to hold her attention on the lyrics for a longer time. In contrast, recommending novel lyrics related in terms of, say, the topic [9–12] would be a good trigger for the user to listen to new songs and expand her interest to other artists; this would also be beneficial for music streaming platforms.

5. CONCLUSION

In this paper, we investigated why and how people view lyrics while listening to music on a smartphone. Regarding the “why” part, we conducted an online user survey involving 206 participants; regarding the “how” part, we analyzed over 23 million lyrics request logs. From the results, we discussed reusable insights that are beneficial for researchers and music streaming platforms, such as the extent of the demand for the eight major reasons to view lyrics and the generality of repeatedly viewing the same lyrics. We also suggested several functions according to users’ reasons for viewing lyrics. We believe that realizing the functions would diversify and enrich users’ music listening experiences. Some of the reported findings might be obvious (e.g., people view lyrics more often at night). However, in this kind of study that investigates research questions on an unexplored topic, it is valuable to report not only unexpected results but also such obvious results based on the data; obvious but verified results can then be used as evidence for claiming the appropriateness of proposed methods or systems in later studies.

We acknowledge a limitation of this paper in that we investigated lyrics viewing behavior by only Japanese people in both the “why” and “how” parts. Nonetheless, we believe that our study is a worthwhile contribution to MIR community, because this is the first attempt to reveal lyrics viewing behavior and verifies the fundamental characteristics of the behavior. At the same time, this limitation indicates the possibilities of this research topic and guides future work such as investigating the differences in lyrics viewing behavior among countries. It would also be an important future work to investigate lyrics viewing behavior on other devices (e.g., PCs and tablets) and at various locations (e.g., homes, restaurants, and public transportation).

6. ACKNOWLEDGMENTS

The authors would like to extend their appreciation to SyncPower Corporation for providing the lyrics request logs. This work was supported in part by JSPS KAKENHI Grant Number 20K19934 and JST CREST Grant Number JPMJCR20D4, Japan.

7. REFERENCES

- [1] J. H. Lee and J. S. Downie, "Survey of music information needs, uses, and seeking behaviours: Preliminary findings," in *Proceedings of the 5th International Conference on Music Information Retrieval*, ser. ISMIR '04, 2004, pp. 989–992.
- [2] J. H. Lee, J. S. Downie, and S. J. Cunningham, "Challenges in cross-cultural/multilingual music information seeking," in *Proceedings of the 6th International Conference on Music Information Retrieval*, ser. ISMIR '05, 2005, pp. 1–7.
- [3] D. Bainbridge, S. J. Cunningham, and J. S. Downie, "How people describe their music information needs: A grounded theory analysis of music queries," in *Proceedings of the 4th International Conference on Music Information Retrieval*, ser. ISMIR '03, 2003, pp. 221–222.
- [4] A. Demetriou, A. Jansson, A. Kumar, and R. M. Bitner, "Vocals in music matter: The relevance of vocals in the minds of listeners," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, ser. ISMIR '18, 2018, pp. 514–520.
- [5] E. Brochu and N. de Freitas, "'Name that song!'" a probabilistic approach to querying on music and text," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS '02, 2002, pp. 1505–1512.
- [6] M. Müller, F. Kurth, D. Damm, C. Fremerey, and M. Clausen, "Lyrics-based audio retrieval and multimodal navigation in music collections," in *Proceedings of the 11th European Conference on Digital Libraries*, ser. ECDL '07, 2007, pp. 112–123.
- [7] T. Hosoya, M. Suzuki, A. Ito, and S. Makino, "Lyrics recognition from a singing voice based on finite state automaton for music information retrieval," in *Proceedings of the 6th International Conference on Music Information Retrieval*, ser. ISMIR '05, 2005, pp. 532–535.
- [8] C. Wang, J. R. Jang, and W. Wang, "An improved query by singing/humming system using melody and lyrics information," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ser. ISMIR '10, 2010, pp. 45–50.
- [9] S. Sasaki, K. Yoshii, T. Nakano, M. Goto, and S. Morishima, "LyricsRadar: A lyrics retrieval system based on latent topics of lyrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ser. ISMIR '14, 2014, pp. 585–590.
- [10] T. Nakano and M. Goto, "LyricListPlayer: A consecutive-query-by-playback interface for retrieving similar word sequences from different song lyrics," in *Proceedings of the Sound and Music Computing Conference 2016*, ser. SMC '16, 2016, pp. 344–349.
- [11] K. Tsukuda, K. Ishida, and M. Goto, "Lyric Jumper: A lyrics-based music exploratory web service by modeling lyrics generative process," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR '17, 2017, pp. 544–551.
- [12] K. Watanabe and M. Goto, "Query-by-Blending: A music exploration system blending latent vector representations of lyric word, song audio, and artist," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, ser. ISMIR '19, 2019, pp. 144–151.
- [13] D. Baur, J. Büttgen, and A. Butz, "Listening factors: A large-scale principal components analysis of long-term music listening histories," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12, 2012, pp. 1273–1276.
- [14] J. H. Lee and R. Price, "Understanding users of commercial music services through personas: Design implications," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ser. ISMIR '15, 2015, pp. 476–482.
- [15] J. H. Lee and N. M. Waterman, "Understanding user requirements for music information services," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, ser. ISMIR '12, 2012, pp. 253–258.
- [16] A. J. Lonsdale and A. C. North, "Why do we listen to music? A uses and gratifications analysis," *British Journal of Psychology*, vol. 102, no. 1, pp. 108–134, 2011.
- [17] W. M. Randall and N. S. Rickard, "Reasons for personal music listening: A mobile experience sampling study of emotional outcomes," *Psychology of Music*, vol. 45, no. 4, pp. 479–495, 2017.
- [18] A. B. Haake, "Individual music listening in workplace settings: An exploratory survey of offices in the UK," *Musicae Scientiae*, vol. 15, no. 1, pp. 107–129, 2011.
- [19] J. H. Lee, L. Pritchard, and C. Hubbles, "Can we listen to it together?: Factors influencing reception of music recommendations and post-recommendation behavior," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, ser. ISMIR '19, 2019, pp. 663–669.

- [20] A. C. North, D. J. Hargreaves, and J. J. Hargreaves, "Uses of music in everyday life," *Music Perception: An Interdisciplinary Journal*, vol. 22, no. 1, pp. 41–77, 2004.
- [21] L. A. Liikkanen and A. Salovaara, "Music on YouTube: User engagement with traditional, user-appropriated and derivative videos," *Computers in Human Behavior*, vol. 50, pp. 108–124, 2015.
- [22] J. H. Lee, R. Wishkoski, L. Aase, P. Meas, and C. Hubbles, "Understanding users of cloud music services: Selection factors, management and access behavior, and perceptions," *Journal of the Association for Information Science and Technology*, vol. 68, no. 5, pp. 1186–1200, 2017.
- [23] M. Schedl, "The LFM-1b dataset for music retrieval and recommendation," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '16, 2016, pp. 103–110.
- [24] G. Vigiensoni and I. Fujinaga, "The music listening histories dataset," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR '17, 2017, pp. 96–102.
- [25] M. Schedl, "Leveraging microblogs for spatiotemporal music information retrieval," in *Proceedings of the 35th European Conference on Advances in Information Retrieval*, ser. ECIR '13, 2013, pp. 796–799.
- [26] D. Hauger, M. Schedl, A. Kosir, and M. Tkalcic, "The million musical tweet dataset: What we can learn from microblogs," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, ser. ISMIR '13, 2013, pp. 189–194.
- [27] E. Zangerle, M. Pichl, W. Gassler, and G. Specht, "#nowplaying music dataset: Extracting listening behavior from twitter," in *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*, ser. WISMM '14, 2014, pp. 21–26.
- [28] A. Anderson, R. Kumar, A. Tomkins, and S. Vas-silvitskii, "The dynamics of repeat consumption," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14, 2014, pp. 419–430.
- [29] A. R. Benson, R. Kumar, and A. Tomkins, "Modeling user consumption sequences," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16, 2016, pp. 519–529.
- [30] G. Dzhabazov, A. Srinivasamurthy, S. Sentürk, and X. Serra, "On the use of note onsets for improved lyrics-to-audio alignment in Turkish makam music," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, ser. ISMIR '16, 2016, pp. 716–722.
- [31] K. Lee and M. Cremer, "Segmentation-based lyrics-audio alignment using dynamic programming," in *Proceedings of the 9th International Conference on Music Information Retrieval*, ser. ISMIR '08, 2008, pp. 395–400.
- [32] V. Thomas, C. Fremerey, D. Damm, and M. Clausen, "Slave: A score-lyrics-audio-video-explorer," in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, ser. ISMIR '09, 2009, pp. 717–722.
- [33] H. Fujihara, M. Goto, J. Ogata, K. Komatani, T. Ogata, and H. G. Okuno, "Automatic synchronization between lyrics and music CD recordings based on viterbi alignment of segregated vocal signals," in *Proceedings of the 8th IEEE International Symposium on Multimedia*, ser. ISM '06, 2006, pp. 257–264.
- [34] M.-Y. Kan, Y. Wang, D. Iskandar, T. L. Nwe, and A. Shenoy, "LyricAlly: Automatic synchronization of textual lyrics to acoustic music signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 338–349, 2008.
- [35] H. Fujihara, M. Goto, J. Ogata, and H. G. Okuno, "LyricSynchronizer: Automatic synchronization system between musical audio signals and lyrics," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1252–1261, 2011.
- [36] M. Mauch, H. Fujihara, and M. Goto, "Integrating additional chord information into HMM-based lyrics-to-audio alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 200–210, 2012.
- [37] R. J. Ellis, Z. Xing, J. Fang, and Y. Wang, "Quantifying lexical novelty in song lyrics," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, ser. ISMIR '15, 2015, pp. 694–700.
- [38] H. Hirjee and D. G. Brown, "Automatic detection of internal and imperfect rhymes in rap lyrics," in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, ser. ISMIR '09, 2009, pp. 711–716.
- [39] X. Hu and B. Yu, "Exploring the relationship between mood and creativity in rock lyrics," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ser. ISMIR '11, 2011, pp. 789–794.
- [40] E. Nichols, D. Morris, S. Basu, and C. Raphael, "Relationships between lyrics and melody in popular music," in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, ser. ISMIR '09, 2009, pp. 471–476.

- [41] A. Singhi and D. G. Brown, "Are poetry and lyrics all that different?" in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, ser. ISMIR '14, 2014, pp. 471–476.
- [42] C. Johnson-Roberson and M. Johnson-Roberson, "Temporal and regional variation in rap lyrics," in *NIPS Workshop on Topic Models: Computation, Application and Evaluation*, ser. NIPSW '13, 2013.
- [43] P. Knees, M. Schedl, and G. Widmer, "Multiple lyrics alignment: Automatic retrieval of song lyrics," in *Proceedings of the 6th International Conference on Music Information Retrieval*, ser. ISMIR '05, 2005, pp. 564–569.
- [44] G. Geleijnse and J. H. M. Korst, "Efficient lyrics extraction from the web," in *Proceedings of the 7th International Conference on Music Information Retrieval*, ser. ISMIR '06, 2006, pp. 371–372.
- [45] R. Macrae and S. Dixon, "Ranking lyrics for online search," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, ser. ISMIR '12, 2012, pp. 361–366.
- [46] X. Hu and J. S. Downie, "When lyrics outperform audio for music mood classification: A feature analysis," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ser. ISMIR '10, 2010, pp. 619–624.
- [47] R. Mayer, R. Neumayer, and A. Rauber, "Rhyme and style features for musical genre classification by song lyrics," in *Proceedings of the 9th International Conference on Music Information Retrieval*, ser. ISMIR '08, 2008, pp. 337–342.
- [48] R. Mayer and A. Rauber, "Music genre classification by ensembles of audio and lyrics features," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ser. ISMIR '11, 2011, pp. 675–680.
- [49] X. Wang, X. Chen, D. Yang, and Y. Wu, "Music emotion classification of Chinese songs based on lyrics using TF*IDF and rhyme," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, ser. ISMIR '11, 2011, pp. 765–770.
- [50] B. Wei, C. Zhang, and M. Ogihara, "Keyword generation for lyrics," in *Proceedings of the 8th International Conference on Music Information Retrieval*, ser. ISMIR '07, 2007, pp. 121–122.
- [51] M. van Zaanen and P. Kanters, "Automatic mood classification using TF*IDF based on lyrics," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ser. ISMIR '10, 2010, pp. 75–80.
- [52] H. Fujihara, M. Goto, and J. Ogata, "Hyperlinking lyrics: A method for creating hyperlinks between phrases in song lyrics," in *Proceedings of the 9th International Conference on Music Information Retrieval*, ser. ISMIR '08, 2008, pp. 281–286.
- [53] D. Baur, B. Steinmayr, and A. Butz, "SongWords: Exploring music collections through lyrics," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ser. ISMIR '10, 2010, pp. 531–536.
- [54] W. Machida and T. Itoh, "Lyricon: A visual music selection interface featuring multiple icons," in *Proceedings of the 15th International Conference on Information Visualisation*, ser. IV '11, 2011, pp. 145–150.
- [55] S. Funasawa, H. Ishizaki, K. Hoashi, Y. Takishima, and J. Katto, "Automated music slideshow generation using web images based on lyrics," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, ser. ISMIR '10, 2010, pp. 63–68.
- [56] T. O'Hara, N. Schüler, Y. Lu, and D. Tamir, "Inferring chord sequence meanings via lyrics: Process and evaluation," in *Proceedings of the 13th International Society for Music Information Retrieval Conference*, ser. ISMIR '12, 2012, pp. 463–468.
- [57] K. M. Ibrahim, D. Grunberg, K. Agres, C. Gupta, and Y. Wang, "Intelligibility of sung lyrics: A pilot study," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, ser. ISMIR '17, 2017, pp. 686–693.
- [58] T. Nakano, M. Goto, and Y. Hiraga, "An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features," in *Proceedings of the 9th International Conference on Spoken Language Processing*, ser. INTERSPEECH '06, 2006, pp. 1706–1709.
- [59] M. Goto, "A chorus section detection method for musical audio signals and its application to a music listening station." *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1783–1794, 2006.
- [60] K. Watanabe and M. Goto, "A chorus-section detection method for lyrics text," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, ser. ISMIR '20, 2020, pp. 351–359.
- [61] Statistics Bureau of Japan, *Survey on time use and leisure activities*. Japan Statistical Association, Tokyo, 2016.

THE WORDS REMAIN THE SAME: COVER DETECTION WITH LYRICS TRANSCRIPTION

Andrea Vaglio^{1,2}

Romain Hennequin¹

Manuel Moussallam¹

Gaël Richard²

¹ Deezer R&D

² LTCI, Télécom Paris, Institut Polytechnique de Paris

research@deezer.com

ABSTRACT

Cover detection has gained sustained interest in the scientific community and has recently made significant progress both in terms of scalability and accuracy. However, most approaches are based on the estimation of harmonic and melodic features and neglect lyrics information although it is an important invariant across covers. In this work, we propose a novel approach leveraging lyrics without requiring access to full texts through the use of lyrics recognition on audio. Our approach relies on the fusion of a singing voice recognition framework and a more classic tonal-based cover detection method. To the best of our knowledge, this is the first time that lyrics estimation from audio has been explicitly used for cover detection. Furthermore, we exploit efficient string matching and an approximated nearest neighbors search algorithm which lead to a scalable system which is able to operate on very large databases. Extensive experiments on the largest publicly available cover detection dataset demonstrate the validity of using lyrics information for this task.

1. INTRODUCTION

Cover detection, also known as version identification, aims at detecting whether two recordings are of the same underlying musical work. A cover can be played by the same artist as the original song, or by another artist, and can be quite similar or vastly different. Generally, it is assumed, as in [1], that tonal progression features (chord, melody, and harmony) are mostly preserved between covers of the same work. Inversely, musical attributes such as key, timbre, tempo, and structure significantly vary across covers [1]. Variations of these features between covers were extensively studied in [2]. Cover detection systems are then built to be insensitive to these variations and exploit tonal progression features. The task has been frequently studied as a *query and answer* [1] one, i.e. given an input query, the system outputs a ranked list of possible covers from a

music collection. True covers are to be ranked as highly as possible while other songs should be ranked low. This list is usually obtained by computing pairwise similarities between the query and each song of a pre-defined dataset [1]. If earlier cover detection systems were shown to be highly efficient on small datasets (1000 songs or less) [3], performances quickly dropped on larger ones [2, 4]. Recent works have made significant advances in scalability and accuracy, for larger datasets, taking inspiration from metric learning [5] and knowledge distillation [6].

Almost none of the existing approaches explicitly consider the textual information provided by the lyrics. To the best of our knowledge, it is only used in [4], in which lyrics are assumed to be available for a significant part of the dataset. In this paper, the authors use metadata and lyrics alongside audio to perform cover detection. The textual similarity of lyrics and song titles is computed using a plain Bag-of-words *Term Frequency–Inverse Document Frequency* (TFIDF). The authors show that results obtained with lyrics are on par with those given by audio-based features on a large-scale dataset. Moreover, the best results are obtained when combining all of the features. However, each feature is only used in a separate part of a multi-layer database pruning method; the information carried by each modality is thus not optimally combined. One limitation of this work is that it assumes that the lyrics of most songs are available. Considering the task of query by singing, which may be regarded as a related task to the cover detection one, authors in [7] employed lyrics and melody recognition to recognize a singing query and match it against a collection of songs. They employed a basic bigram *Hidden Markov model* (HMM) model that is trained on speech and adapted to singing voice. However, this approach also presupposes that the lyrics of songs are available. Lyrics from the considered dataset are, in fact, utilized to inform singing voice recognition.

While this assumption arguably does not hold for large musical collections, one could turn to *Singing Voice Recognition* (SVR) frameworks to retrieve a noisy estimate of the lyrics. We thus propose a novel cover detection approach leveraging lyrics information extracted from audio. It is based on the fusion of a SVR framework and a more classic tonal-based cover detection system. Based on our review of the literature, this is the first time that an estimation of lyrics transcripts from audio has been explicitly leveraged



to perform cover detection. Our assumption, based on the results of [4], is that lyrics are often preserved between covers in popular western music. For the first modality of our fused system, we thus propose using transcription methods to obtain estimates of these lyrics for all songs. The cover song here is framed as a noisy text-matching task. We expect a lyrics-recognition based system to be particularly relevant for pairs of covers displaying hugely different tonal features while using the same lyrics. An example of such cases is the cover of *Summertime* by *Janis Joplin* where the harmony and melody are considerably different from the original score, but the lyrics remain quite similar. Nevertheless, it is clear that a pure lyrics-based system is inadequate for instrumental music (e.g. without a singing voice). Therefore, we use a tonal-based system such as the second modality of our fused system. An instrumental detector is applied on the output of the lyrics recognition framework to inform the fusion strategy. We provide extensive empirical evidence that both modalities are indeed complementary. Extra attention is placed on the scalability of our proposed approach using *Approximated Nearest Neighbors* (ANN) methods.

2. RELATED WORKS

Classically, cover song detection systems use tonal features, which are thought to be the least altered between a song and its covers. Chroma [8] and derived features such as *Harmonic Pitch Class Profil* (HPCP) [3] and *CremaPCP* [5] are among most effective examples. Before computing the similarity between two songs, multiple preprocessing steps can be applied to obtain features that are invariant to the key [9], the tempo [10], or the structure of the song [5]. After extracting the features to be compared for both songs, a cross similarity matrix [11], or a cross recurrent plot [9], is then generally computed. A similarity score is then computed using dynamic programming like *Dynamic Time Warping* (DTW) [12] and recurrence quantification analysis [3]. For a given query, this score is calculated for all tracks in a pre-defined dataset and thus yields the desired sorted list. These methods achieve satisfactory results for small datasets of up to a thousand songs [3], but are computationally costly for larger datasets.

To address this issue, some authors have attempted to reduce the size of the input representation to obtain a low-dimensional fixed size representation for each track. The similarity comparison thus boils down to a basic distance metric such as Euclidean distance or cosine similarity [5] that are much faster than dynamic programming algorithms of quadratic complexity. Early approaches of this type include using fingerprinting in the form of Chroma landmarks [10] and *2D* Fourier transform of Chroma vectors [13], both obtaining low performances. More recent approaches using metric learning, triplet loss, and distillation methods show greater improvement [5, 6] in terms of computation speed and retrieval performances. Database pruning was also used to decrease the overall complexity in [4, 14]. A first fast global candidate selection using text and metadata was performed, followed by a more

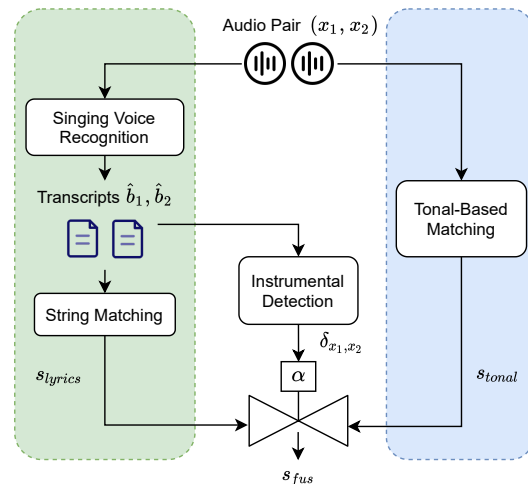


Figure 1. Audio from a pair of tracks is processed in parallel by two branches computing lyrics and tonal-based similarities respectively. The fusion mechanism is informed by an *instrumental* detection on the transcripts

complex similarity function to re-rank the subset. Most of these approaches, which are based on low-dimensional embeddings and simple distance functions, are then simply exploited into existing scalable nearest-neighbors methods. For example, the authors in [10, 15] use index-based matching on extracted audio fingerprinting. Scalable nearest-neighbors methods are more broadly discussed in Section 3.6.

3. PROPOSED APPROACH

A general overview of our approach is described in Figure 1. It is composed of a lyrics-recognition based cover detection system and a classic tonal-based system. The first branch is constituted of a lyrics recognition framework and a string matching function. It takes two songs x_1 and x_2 as input and outputs the respective estimated lyrics \hat{b}_1 and \hat{b}_2 . A similarity estimation s_{lyrics} is then obtained using these transcripts. The second branch of our approach, the classic tonal-based system, also takes these two songs as input and outputs a similarity estimation s_{tonal} . They are then fused using a fusion function to obtain a new similarity estimation s_{fus} . Extra input is added to the fusion function α to weigh the participation of both modalities during the fusion. The value of this input depends on the instrumental detector taking as input both transcripts and outputting the probability that at least one of the tracks is purely instrumental. This avoids using the lyrics-based recognition system during the fusion in the absence of lyrics. To obtain the desired sorted list, for a given query, a similarity is then computed for the considered system between the query and each track of the dataset. Finally, a fast approximate index search technique is used on our system to make it scalable. In our work, we rely on the ANN approach where the similarity is only computed between the query and the nearest neighbors returned by the method.

3.1 Lyrics recognition

We choose a state-of-the-art framework [16] that obtained the best results in the *Music Information Retrieval Evaluation eXchange* (MIREX) 2020 lyrics transcription challenge¹. It uses an acoustic model composed of several layers of *Time Delay Neural Network* (TDNN) that are trained using the English tracks of the DALI dataset [17]. Background music is directly modeled as an output of the acoustic model as such that it does not use any preprocessing step of *Singing Voice Separation* (SVS). Moreover, phoneme units are annotated with genre labelling information. An extended lexicon is also employed to handle long-vowel duration. Finally, a 3-gram word language model with interpolated Kneser-Ney smoothing is trained on the English portion of DALI lyrics. The complete framework extracts *Mel-Frequency Cepstral Coefficients* (MFCC) of dimension 40 from the input audio and outputs transcribed English words. As this model cannot output non-English words, extra care on the results of non-English tracks will be considered later in this paper. The acoustic model and lexicon are collected from the code implementation of the authors². We compute the language model with the kenLM toolkit [18]. The vocabulary of the language model is restricted to the 6000 most frequent words, thus reducing overfitting. Obtained transcription results are on par with those in MIREX with a *Word Error rate* (WER) of 62% on the Jamendo dataset [19].

3.2 String matching

To allow for a swift computation of the similarity between pairs of estimated transcripts, each string is transformed to a vector using a TFIDF based on a 3-gram at character level with IDF values computed from the DALI dataset. The complexity of this type of algorithm is $O(m + n)$ with m and n , which are the respective length of each transcript. The similarity is then simply given using a cosine similarity, which is independent of the length of each transcript. Using a word level 3-gram was not shown to improve performances on a cover song tuning set described in Section 4. We also considered the Levenshtein distance for the string matching, but it did not yield significant gains in performances while inducing a quadratic complexity.

3.3 Detecting instrumentals

Looking at various transcripts given by our SVR framework, we notice that, for most instrumental tracks, the transcript obtained is composed of either a very few number of words, or highly repeated ones such as onomatopoeia. Therefore, we consider a track as instrumental if the respective transcription is composed of less than l different words with l tuned on the cover song tuning set. The module outputs $\delta_{x_1, x_2} = 1$ if both tracks are not detected as instrumentals, and 0 otherwise. For some rare cases where the SVR framework is truly performing poorly, it is also

¹ https://www.music-ir.org/mirex/wiki/2020:MIREX2020_Results

² <https://github.com/chitralkhal18/AutoLyrixAlign>

only outputting a few words. The instrumental detector then helps with additionally filtering some marginal cases where the lyrics transcription fails completely. We chose to keep this very simple as it performed sufficiently well for our purposes and allowed for improvements in future works.

3.4 Tonal-based cover detection

The tonal-based cover detection method selected is described in [6]. This system, called Re-MOVE [6], is an updated version of MOVE [5] and obtains the second most accurate benchmark on the Da-Tacos dataset [2]. Compared to the best one reported [20], it has the advantage of being publicly available³. The system is trained using the training part of Da-Tacos, as described in Section 4.1, and early stopping is performed using its validation component. For a given track, the system takes CremaPCP extracted from the audio as input and outputs a corresponding compact embedding. The CremaPCP feature is an intermediate representation of a chord estimation model. It is considered an efficient way to capture the tonal information of music and is shown to outperform more classic HPCP features for cover detection [2]. The similarity between the query and each track of the dataset is then the cosine similarity of their respective embeddings.

The Re-MOVE system uses a latent space reconfiguration technique on top of MOVE in order to reduce the embedding dimension (and then reduce memory requirements and retrieval time) while maintaining high detection performances. This technique reconfigures a pre-trained learned distance metric into a more compact embedding space with the same learned semantic relation.

3.5 Fusion

It has been shown in multiple domains that the fusion of different modalities can yield better performances than those obtained with each single modality [21]. For cover detection, fusing modalities, features or similarities matrix have already shown to improve results [22, 23], notably using rank aggregation methods [24]. The fusion function chosen here is a weighted sum. It is more precisely described by:

$$s_{fus} = \begin{cases} \alpha s_{lyrics} + (1 - \alpha) s_{tonal} & \text{if } \delta_{x_1, x_2} = 1 \\ s_{tonal} & \text{otherwise} \end{cases} \quad (1)$$

α is a simple scalar defined as an hyperparameter to tune. As the distributions of both similarities are very different, calibration before fusion was also tested. However, no improvement was shown on the cover song tuning set. Other fusion functions, such as linear regression or max function, did not lead to improvements in our simulations.

3.6 Scalability

Pairwise comparisons between a given query and all tracks in a dataset are linearly dependent on the size of the dataset

³ <https://github.com/furkanyesiler/re-move>

without optimization, which cannot be considered scalable. In fact, a linear complexity for the queries involves a quadratic complexity for retrieving all musical works in the dataset, which can quickly become prohibitive for large collections. To achieve better scalability properties, most cover detection studies use ANN methods such as *Locality Sensitive Hashing* (LSH) [25, 26]. The idea behind ANN is that for a given query x and a database D the method outputs an approximation of the k nearest neighbors of the query in the database with the complexity being sublinear in the size of the database. For a given query, in contrast with classic *K-Nearest Neighbors* (KNN), ANN methods are only browsing a subset of the complete search graph. All these methods are based on an index table allowing fast queries by outputting a "good" guess of the k nearest neighbors of a given query, making it possible to recover the most highly classified covers in the ranked list obtained with all candidate points. The recall is used to quantify the quality of an ANN method by averaging percentages obtained, for various queries, of true k-nearest-neighbors from k points returned by the method. In our case, we use the Hierarchical Navigable Small World Graph (HNSW) state-of-the-art ANN method; an extensive study of it is given in [27]. This algorithm gives logarithmic complexity for a query in terms of the size of the dataset. This method is directly applied on Re-MOVE and TFIDF embeddings, outputting for a given query k nearest neighbors for each of them. Both sets of points are then concatenated and merged, obtaining a maximum of $2k$ points to consider for the fusion. Pairwise similarities between the query and these points are then generated using the Re-MOVE system and our lyrics-recognition pipeline.

4. EXPERIMENTAL EVALUATION

4.1 Dataset

Da-Tacos [2, 6] is the largest publicly available dataset for cover detection; the training set is composed of 83904 songs in 14999 cliques and the validation set of 14000 songs in 3500 cliques. A clique is defined as a cover group gathering multiple recordings of the same underlying "piece". The Da-Tacos benchmark test subset is a 15000 tracks dataset composed of 1000 cliques with 13 songs each and 2000 noise songs (i.e. that are in a single-song clique) that are not queried. To avoid overfitting, no clique overlaps with any set of Da-Tacos. Instrumentals represent around 20% of the dataset which motivates our choice of using an instrumental detection process. Currently, only a set of precomputed audio features are publicly available for the benchmarking subset test dataset. The dataset is mainly composed of English tracks and popular western music with a few non-English cliques. All hyperparameter tuning made during this paper is carried out on a subpart of the Da-Tacos validation that we choose to refer to as a *Da-Tacos tuning* set. We verified that no clique of this subset overlapped with any clique present in the dataset used to train the tonal-based cover detection system, i.e. the Da-Tacos training set. Also, a clique is dis-

carded if it possesses one track present in the dataset used to train the SVR framework, i.e. DALI dataset. Detection of overlapping tracks and cliques is made using metadata, i.e. titles and artists names. *Da-Tacos tuning* is notably used to choose the string matching algorithm and the fusion function. We recover audio of 12862 tracks from the test dataset. 1849 are in single-song cliques and thus are not queried and only used as noise songs. We make sure no clique of this dataset overlaps with cliques in the Da-Tacos train and validation and that cliques possessing tracks existing in the DALI dataset are discarded. It will be simply referred to as *Da-Tacos test* for the rest of the document.

4.2 Fusion parameters

A track is classified as instrumental if the number of different words of its transcript is less than $l = 8$. This number is adjusted using *Da-Tacos tuning* as the value that maximizes the recall for the highest $F1$ score. Emphasis is put on the recall in order to avoid taking into account the lyrics-recognition based similarity for an instrumental track that has been misclassified as non-instrumental. An α value of 0.6 for fusing both system is tuned on *Da-Tacos tuning*.

4.3 Parameters of ANN

We use the HNSW implementation of the NMSLIB similarity search library [28]. For each query, we return the $k = 100$ nearest neighbors. This choice is derived from [4], which shows performance does not evolve significantly after the top-100 pruning. We use an approximated cosine similarity function to retrieve 100 candidates for each branch which results in, at most, 200 items for the fused model after concatenating and merging both sets.

4.4 Evaluation

The empirical evaluation of the cover detection task performances is given using the *Mean Average Precision* (MAP)⁴. For a query, *Average Precision* (AP) is quantifying the number of actual covers that are highly ranked. The AP score increases when actual covers are detected in the top ranks. The MAP is then simply obtained by averaging on the AP of all queries. As the MAP is not properly defined for systems that do not score every track (such as ANN), we report MAP@100 for these cases considering only the top-100 ranked item of each query. In any case, the MAP does not significantly evolve after the top-100 pruning as explained in the previous section.

5. RESULTS AND DISCUSSION

5.1 Lyrics-recognition based system results

5.1.1 Instrumental detection

Among the 12862 tracks in the test set, 3269 are detected as instrumentals. We compared this with the "Instrumental" tag available in the Da-tacos for all tracks. We obtain a

⁴ Computed using the Metrics toolkit from <https://github.com/benhamner/Metrics>

Query	System	MAP (%)
Da-Tacos-voice	Lyrics	66.4 (0.4)
	Tonal	54.0 (0.4)
Da-Tacos-instr	Lyrics	0.45 (0.06)
	Tonal	47.8 (0.7)

Table 1. Results of lyrics-recognition based and tonal-based cover detection system on *Da-Tacos-voice*. *Da-Tacos-instr* is the subset of the *Da-Tacos test* restricted to instrumental tracks. Standard errors are given in parenthesis

precision of 82.86% for the instrumental detection, a recall of 96.68% and a F1 score of 89.24%. A closer look at misclassified tracks showed that there is some annotation noise in the *Da-Tacos* annotations which could artificially lower the previous metrics. As simple as it is, the instrumental detection performance seems suitable for our application. After filtering detected instrumentals, we obtain a subset of 9593 tracks that we label *Da-Tacos-voice*. 1582 tracks are in single-song cliques. 8011 tracks are then queried.

5.1.2 Lyrics-based cover detection

We first evaluate our lyrics-recognition based system on the *Da-Tacos-voice*. Our results, displayed in Table 1, show that it is generally performing better than the tonal-based one in terms of MAP. They validate the assumption that lyrics can be considered as a strong invariant between covers. It also proves that the most recent state-of-the-art singing voice recognition framework produces transcriptions of sufficiently good quality to perform the cover song as a noisy text matching task. Looking empirically at results coming from both systems, most improvements of the lyrics-recognition system over the tonal-based system come, as expected, from covers with highly different tonal-content and lyrics being roughly the same.

We also query the tracks detected as instrumental and not from single-song cliques. Results are also displayed in Table 1. As expected, performance of the lyrics-recognition based system is almost close to zero. For the tonal-based system, results seem to degrade when compared to non-instrumentals tracks. This suggests that the system has either learned characteristics of the melody carried by the singing voice or implicitly estimated some of the lyrics information to perform cover detection.

5.1.3 The case of non-English tracks

As stated in Section 3.1, our lyrics recognition framework cannot output non-English words, therefore non-English tracks may produce unexpected results. In order to assess the impact of this issue, we predicted a language label for every track of the *Da-Tacos-voice* using a language classifier [29] taking track metadata as input. Results show that the dataset is largely composed of English with more than 92.4% of the tracks being detected as English. Looking at tracks outside single-song cliques detected as non-

Dataset	System	MAP (%)
Da-Tacos test	Fused	62.7 (0.3)
	Fused-wo-inst	50.2 (0.3)
	Tonal	50.6 (0.3)
Da-Tacos-voice	Fused	80.4 (0.3)

Table 2. Results of fused, with and without instrumental detection, tonal and lyrics-recognition based cover detection system on various datasets

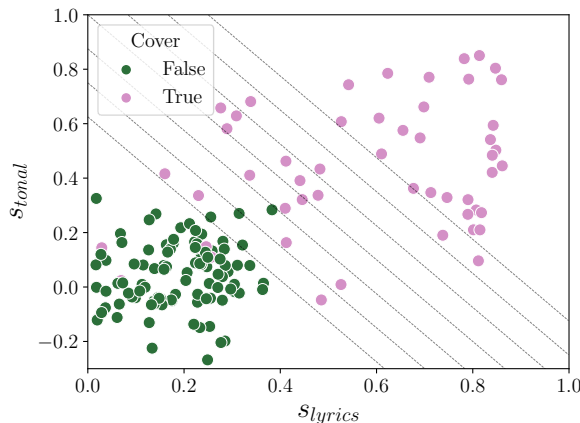


Figure 2. Similarities of sampled pairs of tracks from the *Da-Tacos-voice*. Here, each point is a pair of tracks. Each color indicates a same-clique belonging status. Some level curves of s_{fus} are also displayed

English, half of them are false positives. We query non-English tracks of the resulting 44 cliques on the *Da-Tacos-voice*, representing 299 tracks. It is interesting to report that almost all cliques are homogeneous in terms of language. We obtain a MAP of 28% (2). Results show that even if performances deteriorate for these cases, our system is often able to correctly classify these tracks. It can be explained as the chosen singing voice framework is transcribing something similar from one cover to another even for non-English lyrics. Considering the small quantity of non-English tracks and results on these tracks, we consider that this issue has a limited impact on performance in our evaluation setup.

5.2 Fused system results

Results for the fused system on the full *Da-Tacos test* and its *Da-Tacos-voice* subset are given in Table 2. The fused system significantly outperforms the results of the tonal-based one alone showing the validity of our assumption of both systems being highly complementary. The use of the instrumental detection module to inform the fusion strategy is empirically validated, with a major drop of performances occurring when it is not considered. The gain in performance comes essentially for increased accuracy on the *Da-Tacos-voice* subset, where information from both

System	SVR	MAP (%)
Lyrics	CTC	40.3 (0.7)
	Our	79.0 (0.6)
	Lyrics-informed	89.7 (0.4)
Fused	CTC	71.1 (0.6)
	Our	88.5 (0.4)
	Lyrics-informed	93.6 (0.4)

Table 3. Performances of lyrics-recognition and fused based cover detection system on *Da-Tacos-lyrics* with various SVR framework. Lyrics-informed framework are informed by lyrics at test time

branches is available and the MAP reaches around 80%. To highlight this complementarity, similarities for sampled pairs of tracks from the *Da-Tacos-voice* are displayed in Figure 2. While the majority of same-clique pair lyrics and tonal similarities are significantly higher than non-matching pairs, there are multiple cases where one modality seems more indicative than the other. Level curves of s_{fus} are also displayed illustrating most pairs being linearly separable in the combined modality plane.

5.3 ANN results

We first evaluate the impact of pruning results to the first 100 candidates by computing the MAP@100 of the fused system on the *Da-Tacos test* dataset. A small decrease is observed with a MAP@100 of 62.4% (0.3). After applying an ANN to our fused system, results remain the same with a MAP@100 of 62.4% (0.3). This result can be explained as the recall of the HNSW for both a tonal-based and lyrics-recognition system being more than 99.5%. Thus, the scalability of our system is assured while maintaining the cover detection performances.

5.4 Impact of the SVR framework

A detailed analysis of failing samples of the lyrics-recognition based system shows that the main cause for failure is the low quality of the transcriptions. To further investigate this impact, we introduce two baselines by changing the SVR framework part of our system. In the first, an alternative *Connectionist Temporal Classification* (CTC) based SVR framework is used. The acoustic model of this framework is described in [30]. It consists of several *Bidirectional Long Short-Term Memory* (BiLSTM) layers, is trained on a multilingual subpart of the DALI dataset with a CTC algorithm and relies on a pre-processing step of singing voice separation. The language model used is the same as the one described in Section 3.1. Decoding is performed, after tuning the language model weight and insertion penalty value using a validation dataset, with a CTC beam search decoding toolkit⁵. The transcription of the results obtained on Jamendo dataset [19] are significantly lower than our current singing voice recognition

framework with a WER of 84.4%. We thus expect this CTC-baseline to obtain results far below our system for cover detection tasks.

In the second baseline, we simulate an "ideal" SVR framework outputting an exact transcription. It can be considered as an oracle system, yielding an upper bound for performances of lyrics-recognition based systems. To compare these three systems, we retrieve the lyrics text information for part of the *Da-Tacos test*. The subset obtained is labeled *Da-Tacos-lyrics* and is composed of 3467 tracks for which we found matching lyrics. Considering that this subset only contains non-instrumental tracks, we discard the instrumental detector for this section. Again, tracks from single-song cliques are not queried and are used as noise songs.

The results obtained on *Da-Tacos-lyrics* are given in Table 3. These results confirm the intuition that the lyrics-recognition system’s strength for covering detection task directly depends on the quality of the lyrics transcription. Ranking performances on *Da-Tacos-lyrics* for these systems are conserved after fusing them with the tonal-based branch. In comparison to the oracle system, our fused system shows excellent results even if there is still some room for improvement. With the transcription performances of our SVR framework being as low as 62% WER, it certainly indicates that a perfect transcription is not needed for the cover detection task. Interestingly, even an oracle system informed by the true lyrics benefits from being fused with a tonal-based one. This, once again, demonstrates both branches are acutely complementary to address the cover song detection problem. Future works will extend our system to take into account cases where lyrics are available for a part of the dataset.

6. CONCLUSION

Using only audio, we have proposed a framework that explicitly leverages two types of similarities, tonal and lyrics based, and reach high accuracy levels while remaining simple and scalable. With that said, work on more diverse data still remains to be done, notably on non-English tracks where performances seem to be limited.

Future work will include replacing the current monolingual lyrics recognition with a multilingual framework. A multilingual similarity, capable of detecting the similarity of two texts based on their semantic content, independently of their language, will also be defined and evaluated. More generally, the *Da-Tacos* dataset is quite biased towards popular western music. Additional experimentation on a wider range of genres, notably none western music, and cover types (e.g. karaoke, renditions, etc.) remains to be conducted. Finally, we will explore more elaborate fusion schemes, specifically, a mid-level fusion which can be further optimized and possibly lead to improved performance.

⁵ <https://github.com/parlance/ctcdecode>

7. REFERENCES

- [1] J. Serra, E. Gómez, and P. Herrera, "Audio cover song identification and similarity: background, approaches, evaluation, and beyond," in *Advances in Music Information Retrieval*. Springer, 2010, pp. 307–332.
- [2] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. Gómez Gutiérrez, and X. Serra, "Da-tacos: A dataset for cover song identification and understanding," in *Int. Soc. for Music Information Retrieval (ISMIR)*, 2019.
- [3] J. Serra, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, no. 9, 2009.
- [4] A. A. Correya, R. Hennequin, and M. Arcos, "Large-scale cover song detection in digital music libraries using metadata, lyrics and audio features," *arXiv:1808.10351*, 2018.
- [5] F. Yesiler, J. Serra, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 21–25.
- [6] F. Yesiler, J. Serra, and E. Gomez, "Less is more: Faster and better music version identification with embedding distillation," in *Int. Soc. for Music Information Retrieval (ISMIR)*, 2020.
- [7] C.-C. Wang and J.-S. R. Jang, "Improving query-by-singing/humming by combining melody and lyric information," *IEEE/ACM Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 23, no. 4, pp. 798–806, 2015.
- [8] D. P. Ellis and G. E. Poliner, "Identifying cover songs' with chroma features and dynamic programming beat tracking," in *IEEE Int. Conf. on Acoustics, Speech and Signal (ICASSP)*, vol. 4, 2007.
- [9] J. Serra, E. Gómez, and P. Herrera, "Transposing chroma representations to a common key," in *IEEE CS Conf. on The Use of Symbols to Represent Music and Multimedia Objects*, 2008, pp. 45–48.
- [10] T. Bertin-Mahieux and D. P. Ellis, "Large-scale cover song recognition using hashed chroma landmarks," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2011, pp. 117–120.
- [11] J. Lee, S. Chang, S. K. Choe, and K. Lee, "Cover song identification using song-to-song cross-similarity matrix with convolutional neural network," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 396–400.
- [12] J. Serra, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Trans. on Audio, Speech, and Language Processing (TASLP)*, vol. 16, no. 6, pp. 1138–1151, 2008.
- [13] D. P. Ellis and B.-M. Thierry, "Large-scale cover song recognition using the 2d fourier transform magnitude," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2012.
- [14] J. B. Smith, M. Hamasaki, and M. Goto, "Classifying derivative works with search, text, audio and video features," in *IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2017, pp. 1422–1427.
- [15] T. J. Tsai, T. Prätzlich, and M. Müller, "Known artist live song id: A hashprint approach," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 427–433.
- [16] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics transcription in polyphonic music: Does background music help?" in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [17] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Trans. of the Int. Soc. for Music Information Retrieval (TISMIR)*, vol. 3, no. 1, 2020.
- [18] K. Heafield, "KenLM: Faster and smaller language model queries," in *Workshop on Statistical Machine Translation (SMT)*, 2011.
- [19] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 181–185.
- [20] X. Du, Z. Yu, B. Zhu, X. Chen, and Z. Ma, "Byte-cover: Cover song identification via multi-loss training," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [21] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 3, pp. 226–239, 1998.
- [22] N. Chen, W. Li, and H. Xiao, "Fusing similarity functions for cover song identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2629–2652, 2018.
- [23] C. J. Tralie, "Early mfcc and hpcp fusion for robust cover song identification," in *Int. Soc. for Music Information Retrieval (ISMIR)*, 2017.
- [24] J. Osmalsky, J.-J. Embrechts, P. Foster, and S. Dixon, "Combining features for cover song identification," in *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2015.
- [25] M. Khadkevich and M. Omologo, "Large-scale cover song identification using chord profiles," in *Int. Soc. for Music Information Retrieval (ISMIR)*, vol. 13, 2013, pp. 233–238.

- [26] P. Grosche and M. Müller, “Toward characteristic audio shingles for efficient cross-version music retrieval,” in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 473–476.
- [27] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.
- [28] L. Boytsov and B. Naidan, “Engineering efficient and effective non-metric space library,” in *Similarity Search and Applications (SISAP)*, 2013.
- [29] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” *arXiv:1607.01759*, 2016.
- [30] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. D’alché-Buc, “Multilingual lyrics-to-audio alignment,” in *Int. Soc. for Music Information Retrieval Conference (ISMIR)*, 2020.

MUSEBERT: PRE-TRAINING OF MUSIC REPRESENTATION FOR MUSIC UNDERSTANDING AND CONTROLLABLE GENERATION

Ziyu Wang

Music X Lab, NYU Shanghai
ziyu.wang@nyu.edu

Gus Xia

Music X Lab, NYU Shanghai
gxia@nyu.edu

ABSTRACT

BERT has proven to be a powerful language model in natural language processing and established an effective pre-training & fine-tuning methodology. We see that music, as a special form of language, can benefit from such methodology if we carefully handle its highly-structured and polyphonic properties. To this end, we propose MuseBERT and show that: 1) MuseBERT has detailed specification of note attributes and explicit encoding of music relations, without presuming any pre-defined sequential event order, 2) the pre-trained MuseBERT is not merely a language model, but also a controllable music generator, and 3) MuseBERT gives birth to various downstream music generation and analysis tasks with practical value. Experiment shows that the pre-trained model outperforms the baselines in terms of reconstruction likelihood and generation quality. We also demonstrate downstream applications including chord analysis, chord-conditioned texture generation, and accompaniment refinement.

1. INTRODUCTION

BERT [1] has proven to be one of the leading language models in natural language processing, which learns natural language representation in an unsupervised manner and achieved state-of-the-art results in many downstream language understanding tasks. The methods involved in BERT are not unfamiliar in the music domain. For example, the “masked language model” (MLM) objective in BERT is similar to the Bach chorale inpainting studies [2–4], where the grids in a four-part piano-roll are randomly masked and the model is trained to reconstruct them from context. The Transformer architecture has also been applied to different styles of music generation [5, 6].

We aim to develop a pre-trained music-domain BERT for better music understanding and generation. A straightforward approach is to use existing music representations used in existing Transformer-based models, such as MIDI-like representations [5, 7], REMI [6], or CP [8], and simply train the original BERT model. However, we argue that

such approach overlooks major distinctions between music and natural language. Firstly, unlike natural language, music (especially polyphony) does not follow a unique sequential order; abruptly flatten music into a 1-d sequence imposes extra protocol and often undermines the local structure of music, adding extra burden to the model. Secondly, music involves rich relations and contexts. A single music event is barely meaningful, and common music concepts such as rhythmic patterns and harmonies are established upon relative positions of notes. However, current sequential models, in particular Transformer-based models, still struggle to capture these relations [9].

To overcome the limits above, we propose MuseBERT, a Transformer-based pre-trained model with tailored handling of music positional information and music relations.¹ Since the innate music positions are expressed in the time-frequency space, we use onset and pitch information as the absolute positional encoding, which can be masked at the input and then reconstructed at the output. Moreover, multiple musical relations are represented via the design of *generalized* relative positional encoding (RPE) modified from the original RPE design [10]. In our implementation, we find the traditional sequential positional encoding unnecessary, and thus our model does not rely on any pre-defined sequential assumptions.

Under such tailored design, we show the pre-trained MuseBERT is not merely a language model but also a controllable music generator. In the fine-tuning stage, the model naturally empowers a wide spectrum of fine-tuning tasks, involving both music analysis and generation. Specifically, we showcase that MuseBERT can perform polyphonic music generation controlled by chord and texture, chord extraction, and accompaniment refinement. In summary, the contributions of this paper are:

- We demonstrate the possibility and importance of training a Transformer-based music language model in an *unordered* approach.
- We develop generalized RPE for BERT-like models, which not only reveals music relations from multiple perspectives but also has a potential to be adopted to other domains.
- We show MuseBERT, as a controllable music generator, giving the fine-tuning procedure practical music meanings.



© Z. Wang, and G. Xia. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Z. Wang, and G. Xia, “MuseBERT: Pre-training of Music Representation for Music Understanding and Controllable Generation”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ Code and demos can be accessed via <https://github.com/ZZWang/musebert>.

2. DATA REPRESENTATION

Unlike natural language processing, there is not a common way to tokenize symbolic music. A music piece is generally considered as a combination of note events, where each note event is a tuple of attributes. Note-based tokenization is commonly seen in music-tailored packages [11, 12], softwares [13–15], and researches [8, 16, 17]. On other occasions, mainly for the ease of neural network modeling, music is also simplified as a sequence of controls with pre-defined order, such as MIDI-like event representation [5, 7] (including its follow-up methods [6]), and frame-wise piano-roll representations [2, 18–20].

In MuseBERT, we consider note-based tokenization without presumption of any sequential order. Specifically, we invent two note-based music representations for MuseBERT: *basic data representation* and *factorized data representation*, denoted by $\mathcal{R}_{\text{base}}$ and \mathcal{R}_{fac} respectively. A music input is first encoded as $\mathcal{R}_{\text{base}}$, and then converted into \mathcal{R}_{fac} , serving as the input format to MuseBERT.

In this paper, we primarily consider 2-bar music segments in 4/4 time signature. Longer music and richer time signatures can be generalized and we leave it for future study.

2.1 $\mathcal{R}_{\text{base}}$ and \mathcal{R}_{fac}

$\mathcal{R}_{\text{base}}$ represents a music segment X as an *unordered set* of note events $\{x_i\}_{i=1}^N$, and a note event $x_i \in X$ consists of three attributes: onset (o), pitch (p), and duration (d), where $o \in [0..31]$ and $d \in [0..32]$ are counted by semi-quavers, and $p \in [0..127]$ is the MIDI note number.² We denote the *attribute set* of $\mathcal{R}_{\text{base}}$ by $\mathcal{A}_{\text{base}} := \{o, p, d\}$ and use $x_i.a, a \in \mathcal{A}_{\text{base}}$ to indicate the attribute a of a note event x_i .

$\mathcal{R}_{\text{base}}$ is not the MuseBERT input data format because the representation is *ambiguous* for BERT-like models to reconstruct well. Consider, for example, two simultaneous quarter notes in a music segment that are unordered (by definition of $\mathcal{R}_{\text{base}}$) and whose attributes except p are the same. When their pitch attributes are masked at the input, the two tokens have completely the same encoding and are unable to be distinguished by BERT, in which case BERT will yield the same output distributions.

We therefore invent \mathcal{R}_{fac} which is unambiguous for MuseBERT (proved in section 4). \mathcal{R}_{fac} also represents a music segment as an unordered set of note events, but it uses a more detailed attribute set \mathcal{A}_{fac} (discussed in section 2.2), and contains a stack of relation matrices storing pairwise note relations (discussed in section 2.3).

2.2 Factorized Attributes

In \mathcal{R}_{fac} , onset, pitch, and duration, due to their hierarchical nature [21, 22], are regarded as meta-attributes to be factorized. The factorization operation we will introduce is analogous to storing a number by its digits (e.g., $49 \mapsto [4, 9]$),

² We use $[a..b]$ to denote the integer interval $\{x | a \leq x \leq b, x \in \mathbb{Z}\}$ including both endpoints.

so that we can mask partial information at certain hierarchy (e.g., $[4, \text{MASK}]$) and the remaining information is still retained. In music, musicians sometimes prefer to interpret 49 as $50 - 1$ rather than $40 + 9$, depending on the context. Such subtlety is welcomed by our design.

2.2.1 Onset and Duration Factorization

We factorize onset (o) into *beat position* ($o_{\text{bt}} \in [0..8]$) and *subdivision* ($o_{\text{sub}} \in [0..4]$), and duration d into *half note counts* ($d_{\text{hlf}} \in [0..4]$) and *the remainder semiquaver counts* ($d_{\text{sqv}} \in [0..7]$), satisfying

$$o = (4 \times o_{\text{bt}} + o_{\text{sub}}), \quad (1)$$

$$d = (8 \times d_{\text{hlf}} + d_{\text{sqv}}). \quad (2)$$

We allow o_{sub} to take both positive and negative values, since a subdivision can be interpreted as an “off-beat” of the current downbeat, or an “upbeat” preceding the next downbeat. Consequently, the o attribute can be mapped to multiple $\{o_{\text{bt}}, o_{\text{sub}}\}$ pairs, e.g., $\{o:1\}$ maps to $\{o_{\text{bt}}:0, o_{\text{sub}}:1\}$ or $\{o_{\text{bt}}:1, o_{\text{sub}}:-3\}$.

In our implementation, we *non-deterministically* sample one of the factorizations. In the future, a posterior distribution of $p(o_{\text{bt}}, o_{\text{sub}} | o, X)$ can be explored in finer detail.

2.2.2 Pitch Factorization

We factorize pitch in a similar fashion into three attributes: 1) *pitch highness* ($p_{\text{hig}} \in [0..6]$): voice types (e.g., SATB), 2) *pitch register* ($p_{\text{reg}} \in [0..2]$): the relative octave given p_{hig} , and 3) *pitch degree* ($p_{\text{deg}} \in [0..11]$), satisfying:

$$p = \begin{cases} 24 + 12 \times (p_{\text{hig}} + p_{\text{reg}}) & 0 \leq p_{\text{hig}} \leq 4 \\ \quad \quad \quad + p_{\text{deg}}, & \\ 12 \times p_{\text{reg}} + p_{\text{deg}}, & p_{\text{hig}} = 5 \\ 108 + 12 \times p_{\text{reg}} + p_{\text{deg}}, & p_{\text{hig}} = 6. \end{cases} \quad (3)$$

Here, $0 \leq p_{\text{hig}} \leq 4$ corresponds to the pitch range of a standard piano (MIDI pitch 24-107), and $p_{\text{hig}} = 5$ and $p_{\text{hig}} = 6$ handles the extra-low or extra-high regions, respectively. Similar to onset, pitch factorization is also handled non-deterministically, since a pitch can be interpreted as a lower voice type having a high register, or a higher voice type having a low register.

To summarize, \mathcal{R}_{fac} uses the attribute set:

$$\mathcal{A}_{\text{fac}} := \{o_{\text{bt}}, o_{\text{sub}}, p_{\text{hig}}, p_{\text{reg}}, p_{\text{deg}}, d_{\text{hlf}}, d_{\text{sqv}}\}. \quad (4)$$

2.3 Note Event Relations

Music is rich in relations and we explicitly represent the most fundamental relations in \mathcal{R}_{fac} . We consider less-than, equal-to, and greater-than relations on a subset of attributes $\mathcal{S} := \{o, o_{\text{bt}}, p, p_{\text{hig}}\}$. Specifically, $\forall a \in \mathcal{S}$, we define a mapping γ_a from an input note event pair (x_i, x_j) to their relation symbol:³

³ When $a = o$, $x_i.o$ is a shorthand notation interpreted as the return value of Eqn (1). A similar case holds when $a = p$.

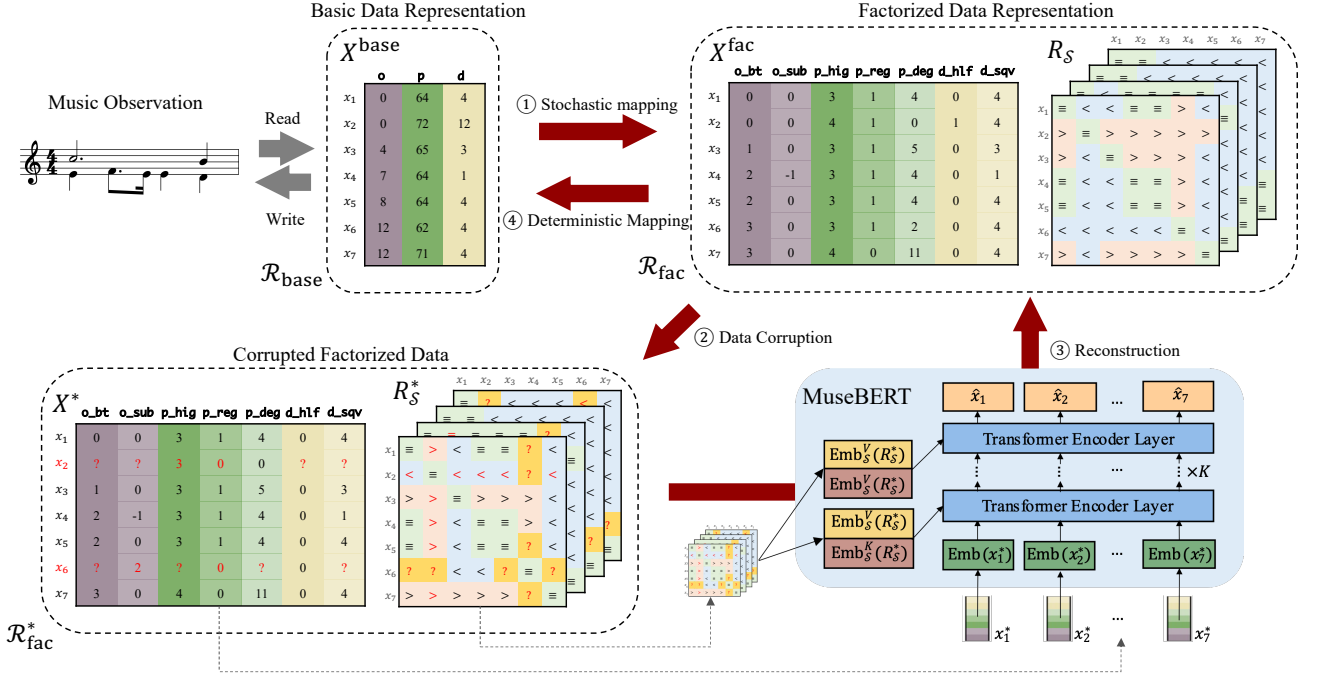


Figure 1. The workflow of MuseBERT pre-training.

$$\gamma_a(x_i, x_j) = \begin{cases} <^a, & x_i.a < x_j.a \\ \equiv^a, & x_i.a \equiv x_j.a \\ >^a, & x_i.a > x_j.a. \end{cases} \quad (5)$$

The relation symbols are stored using a stack of $N \times N$ relation matrices, denoted by $R_S := \{R_a | a \in \mathcal{S}\}$, where $R_a = \{r_{ij}^a\}$ and $r_{ij}^a = \gamma_a(x_i, x_j)$.

3. PRE-TRAINING MUSEBERT

Figure 1 shows the overall pre-training stage. A music data is first read as $\mathcal{R}_{\text{base}}$ format as X^{base} , and then stochastically converted to \mathcal{R}_{fac} consisting of unordered note events X^{fac} and relation matrices R_S . Similar to BERT, the pre-training stage uses ‘‘masked language model’’ (MLM) training objective, where X^{fac} and R_S will be corrupted before fed into MuseBERT, which is trained to reconstruct the original music segment.

3.1 Data Corruption

The original BERT randomly selects a part of the input tokens to be corrupted. In MuseBERT, data corruption has finer controls: 1) it operates on individual attributes of a note event rather than an entire token, and 2) it corrupts relation matrices as well.

For note events X^{fac} , the corrupter randomly selects 15% of the note events $C \subset X$, and corrupts their attributes $\{x_i.a | x_i \in C, a \in \mathcal{A}_{\text{fac}}\}$ in one of the three ways: 1) masked with $[\text{MASK}]_a$ 80% of the time, 2) replaced with a random token 10% of the time, and 3) kept unchanged 10% of the time. Each attribute is corrupted independently of the choice of the other attributes. For relation matrices R_S , each matrix entry will be masked 30% of the time independently. Before corruption, a re-computation of the

matrix may be required so as to maintain matrix symmetry and align with the attributes that are replaced with other values.

We denote the representation after corruption as $\mathcal{R}_{\text{fac}}^*$, and the resulting note events and relation matrices as X^* and R_S^* , respectively.

3.2 Overall Model Architecture

The model adopts the bi-directional Transformer encoder architecture based on the original Transformer implementation described in [23]. The input to the model is the embedding of a corrupted music segment $X^* = \{x_i^*\}_{i=1}^N$. The embedding is computed as the sum of all attribute embeddings:

$$\text{Emb}(x_i^*) = \sum_{a \in \mathcal{A}_{\text{fac}}} \text{Emb}_a(x_i^*.a). \quad (6)$$

Here $\text{Emb}_a(\cdot)$ are linear embedding layers and the default absolute positional encoding is *not* applied due to unorderedness. R_S^* is fed into the model as *generalized relative positional encoding* to be discussed in detail in section 3.3.

The output is a distribution of reconstructed note events in \mathcal{R}_{fac} , denoted by $\hat{X} = \{\hat{x}_i\}_{i=1}^N$. Specifically, let $h_{1:N}$ be the last hidden vector of the Transformer encoder, each note attribute is reconstructed by a separate linear transformation normalized with a softmax layer:

$$p_{\text{model}}(\hat{x}_i.a | X^*, R_S^*) = \text{softmax}(\text{FFN}_a(h_i)). \quad (7)$$

The training loss is the mean of the negative log-likelihood on corrupted attributes $\{x_i.a | x_i \in C, a \in \mathcal{A}_{\text{fac}}\}$ only:

$$\mathcal{L}(\hat{X}|X^*, R_S^*) = -\frac{1}{|C|} \sum_{\{i|x_i^* \in C\}} \left(\sum_{a \in \mathcal{A}_{\text{fac}}} \log p_{\text{model}}(x_i^{\text{fac}}.a|X^*, R_S^*) \right), \quad (8)$$

From the denoising autoencoder perspective [24], we define the reconstruction distribution $p_{\text{recon}}(\hat{X}|X^*, R_S^*)$ induced from Eqn (7) for later analysis purposes:

$$\begin{aligned} p_{\text{recon}}(\hat{X}|X^*, R_S^*) &:= \prod_{i=1}^N p_{\text{recon}}(\hat{x}_i|X^*, R_S^*), \text{ where} \\ p_{\text{recon}}(\hat{x}_i|X^*, R_S^*) &:= \prod_{a \in \mathcal{A}_{\text{fac}}} p_{\text{recon}}(\hat{x}_i.a|X^*, R_S^*) \text{ and} \\ p_{\text{recon}}(\hat{x}_i.a|X^*, R_S^*) &:= \begin{cases} p_{\text{model}}(\hat{x}_i.a|X^*, R_S^*), x_i \in C \\ \mathbb{1}_{\{\hat{x}_i.a \equiv x_i^*.a\}}, \text{ otherwise.} \end{cases} \end{aligned} \quad (9)$$

In other words, during inference, we sample from p_{model} for corrupted tokens while simply keeping the uncorrupted tokens.

3.3 Generalized Relative Positional Encoding

Let $h_{1:N}$ be the hidden vectors at any Transformer layer. We add additional *key-relation embeddings* $\text{Emb}_S^K(\cdot)$ of the relations in the query-value product term by

$$e_{ij} = \frac{h_i W^Q \left(h_j W^K + \sum_{a \in \mathcal{S}} \text{Emb}_a^K(r_{ij}^a) \right)^T}{\sqrt{d_z}}, \quad (10)$$

where d_z is the hidden vector size of attention heads, W^Q, W^K are query and key transformations, and e_{ij} is used to compute attention weights.

Similarly, we also add *value-relation embeddings* $\text{Emb}_S^V(\cdot)$ in the attention weight application:

$$z_i = \sum_{j=1}^N \alpha_{ij} \left(h_j W^V + \sum_{a \in \mathcal{S}} \text{Emb}_a^V(r_{ij}^a) \right), \quad (11)$$

where $\alpha_{ij} = \text{softmax}_j(e_{ij})$, W^V is the value transformation, and z_i is fed into the position-wise feed-forward sub-layer to compute the next layer's h_i .

In our implementation, both key-relation embeddings and value-relation embeddings are linear transforms. In addition, for different attention layers and attention heads, we train different embeddings.

We call the above operation *generalized* relative positional encoding (RPE) because the summation of key-relation and value-relation embeddings allows us to hint the MuseBERT from multiple musical perspectives (attribute relations), which is not considered in the original RPE implementation [10].

4. THEORETICAL ANALYSIS

In this section, we first show that \mathcal{R}_{fac} and generalized relative positional encoding are theoretical necessities for a

BERT model to handle unordered data. We further show that MuseBERT is a powerful controllable music generator, giving birth to various fine-tuning tasks with musical merits.

4.1 Well-definedness of MuseBERT

The fundamental assumption made in MuseBERT is the refusal of traditional *sequential* positional encoding. In return, the mathematical property of MuseBERT we gain is *permutation invariance*, i.e., the input order will not change the model output. On the other hand, a natural downside of a permutation invariant model is *ambiguity*: without extra efforts, masked tokens will have the same output distributions and hence not distinguishable.

The problem is solved by factorized data representation and generalized RPE because the former allows us to specify music in more detail. We summarize the discussion with the well-definedness theorem of MuseBERT. Part one of the theorem is maintained by the property of Transformer [25], and part two is self-evident from the discussion so far.

Theorem 1 (Well-definedness) *Pre-training MuseBERT is well-defined: Let (X^*, R_S^*) be a corrupted music segment in \mathcal{R}_{fac} :*

1) (*Permutation invariance*) *Let $\sigma(\cdot)$ be an arbitrary permutation. The reconstruction distribution is unchanged after permutation:*

$$p_{\text{recon}}(\hat{X}|X^*, R_S^*) \equiv p_{\text{recon}}(\sigma(\hat{X})|\sigma(X^*), \sigma(R_S^*))$$

2) (*Unambiguity*) *For arbitrary pair of $x_i, x_j \in X^*$, whose attributes are corrupted but their note relations are not, the reconstruction distributions of the two tokens are not always the same, i.e., there exists a set of model parameters such that*

$$p_{\text{recon}}(\hat{x}_i|X^*, R_S^*) \neq p_{\text{recon}}(\hat{x}_j|X^*, R_S^*).$$

4.2 MuseBERT as a Music Generator

The operations of MuseBERT can be regarded as a single round of a Markov chain transition which alternatively adds noise and denoises (as shown in Figure 1), consisting of four steps: 1) stochastic data conversion to \mathcal{R}_{fac} , 2) data corruption, 3) MuseBERT reconstruction, and 4) deterministic data conversion to $\mathcal{R}_{\text{base}}$.

Benjio et al. [26] showed such transition induced from denoising autoencoder is a special type of Generative Stochastic Network and proved that under mild condition, the transition operations form a Markov Chain whose stationary distribution is the true data distribution. Hence, we conclude the pre-training MuseBERT can be used as the MCMC transition operator, and generates music via iterative sampling.

4.3 Controllability of MuseBERT

Moreover, we show that MuseBERT is a Constraint Optimization Problem (COP) solver [27] by regarding uncorrupted attributes or relations as unary or binary constraints,

respectively. The COP has the form:

$$\begin{aligned}
 & \max_{\hat{X}} p_{\text{recon}}(\hat{X}|X^*) & (12) \\
 & \text{s.t. } \hat{x}_i.a = x_i^*.a, \quad \forall x_i^* \notin C, a \in \mathcal{A}_{\text{fac}} \\
 & \quad \gamma_a(x_i, x_j) = \gamma_a(x_i^*, x_j^*), \\
 & \quad \forall a \in \mathcal{S} \quad \text{if } \gamma_a(x_i^*, x_j^*) \text{ uncorrupted.}
 \end{aligned}$$

Apparently, music continuation and in-painting can be formalized as a special COP problem, and monophony, melodic directions, rhythmic patterns, texture, etc. are all expressible concepts by MuseBERT constraints. Hence, MuseBERT, as a music generator, is also controllable, which can be fine-tuned to solve problem-specific generation tasks and controls.

5. FINE-TUNING MUSEBERT

By defining problem-specific data corrupters, we can fine-tune the model for a wide spectrum of conditioned music generation tasks, such as music continuation, inpainting, rhythm or texture-conditioned music generation. In this section, we in turn show two less obvious applications by fine-tuning, in which we extend the vocabulary of \mathcal{R}_{fac} to represent chord and vocal tracks, and apply MuseBERT for both generation and analysis.

5.1 Case 1: Texture Generation and Chord Analysis

MuseBERT can be fine-tuned for chord and texture controls, which are commonly-studied controlling factors in automatic music generation [28, 29]. The fine-tuning task is set by slight modification from the pre-training stage in two steps: 1) We represent chord progression as \mathcal{R}_{fac} events and prepend it to the music segment. Specifically, we define $p_{\text{hig}} = 7$ for chord event, with $p_{\text{reg}} = 0, p_{\text{reg}} = 1, p_{\text{reg}} = 2$ for chord root, chroma, and bass, respectively. 2) We apply an additional “in-chord” inter-relation to indicate whether a note event is a chord-tone or not. The conditioned generation can be trained by masking all the attributes from the music segment while keeping the chord attributes and all relations. Figure 2 shows an example of music generation conditioned on the texture in Figure 2(a) and the chord F major. The generated segment keeps the textural property and deals with inner voices smoothly (shown by the red boxes).

Similarly, by corrupting the chord events and “in-chord” relation, while maintaining the music segment and the other relations, the model can learn to extract chord progressions from a given piece. The fine-tuning result on POP909 [30] shows the model has a 99.35% accuracy on token-wise (MIDI) chord extraction (99.37% for root, 99.05% for chroma and 99.35% for bass).

5.2 Case 2: Accompaniment Refinement

In the accompaniment refinement task, we want to refine an imperfect accompaniment according to a given lead melody. To achieve this, we represent the melody and the accompaniment both as \mathcal{R}_{fac} and concatenate them

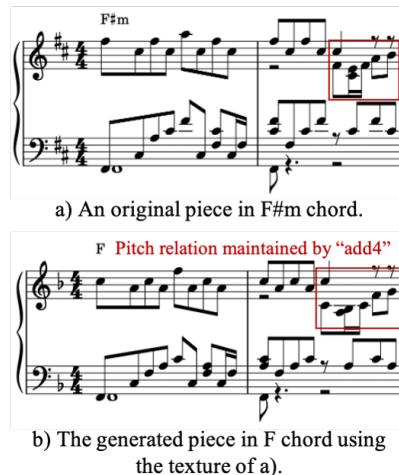


Figure 2. An example of controlled generation by specifying chord and texture.

together. We then randomly corrupt some accompaniment attributes while keeping the melody and relations unchanged during training. Figure 3 shows an example of accompaniment refinement from a corrupted sample Figure 3(a). In (a), the red note heads mark the note being replaced and the masked notes are replaced by rests. The result shows that the generated piece is more consistent with the melody while keeps the original musical content.

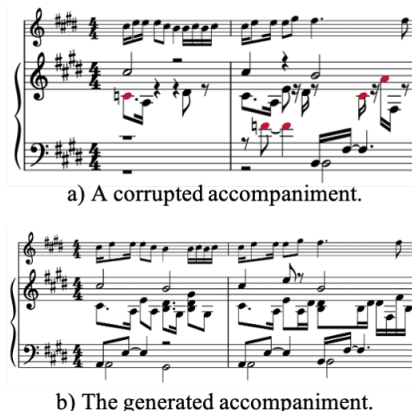


Figure 3. An example of accompaniment refinement conditioned on melody.

6. EXPERIMENTAL RESULT

6.1 Dataset and Training

We collect around 5K classical and popular piano pieces from Musicalion⁴ and the POP909 dataset [30]. We only keep the pieces with 2/4 and 4/4 meters and cut them into 8-beat music segments. In all, we have 19.8K samples. We randomly split the dataset (at song-level) into training set (90%) and test set (10%). All training samples are further augmented by transposing to all 12 keys.

We pre-train our MuseBERT using 12 Transformer layers with 8 attention heads, 0.1 dropout rate, and GELU ac-

⁴ Musicalion: <https://www.musicalion.com>.

tivation [31]. The hidden dimensions of self-attention and position-wise feed-forward layers are $d_{\text{model}} = 256$ and $d_{\text{ff}} = 512$. We train with a batch size of 128 and Adam optimizer with learning rate of $5e-4$ and linear learning rate decay to $5e-6$ after 15K warmup steps. The pre-training takes 20 epochs to converge.

In fine-tuning tasks, we kept the same hyperparameters except the initial learning rate is $2e-4$ without warmup. The epochs to fine-tune the model are task-specific, approximately 4 epochs on average.

6.2 Objective Evaluation

Given that \mathcal{R}_{fac} and data corruption are stochastic, we evaluate the expectation of reconstruction likelihood in terms of token-wise onset, pitch and duration, similar to the algorithm in [4]. We compare among 1) the pre-trained MuseBERT, 2) the fine-tuned models using attribute-specific data corrupters, and three baseline models ablating the use of factorized attributes \mathcal{A}_{fac} and binary relations. Table 1 summarizes the results where we see our pre-trained model outperforms the baselines on all three criteria and the fine-tuned model can perform even better. The use of \mathcal{A}_{fac} and binary relations not only makes the model well-defined, but also are crucial for high-quality reconstruction.

Models	Onset	Pitch	Duration
\mathcal{A}_{fac} w/ rel (ours)	0.902 ± 0.002	0.813 ± 0.003	0.815 ± 0.003
\mathcal{A}_{fac} w/o rel*	0.472 ± 0.002	0.740 ± 0.004	0.743 ± 0.003
$\mathcal{A}_{\text{base}}$ w/ rel	0.856 ± 0.001	0.785 ± 0.004	0.785 ± 0.003
$\mathcal{A}_{\text{base}}$ w/o rel*	0.488 ± 0.003	0.733 ± 0.003	0.733 ± 0.002
\mathcal{A}_{fac} w/ rel (fine-tuned)	0.958 ± 0.001	0.924 ± 0.002	0.927 ± 0.002

Table 1. Objective evaluation of the expectation of reconstruction likelihood. Here, “w/ rel” and “w/o rel” indicates whether generalized RPE is applied, and “*” indicates the ill-defined models.

6.3 Self-attention Visualization

We observe that different attention heads of multiple layers in the pre-trained MuseBERT captures high-level music knowledge. As shown in Figure 4, metrical structure is revealed in (a) & (b), and voice-leading can be seen in (c) & (d). We show the attention matrices on a piano-roll for better interpretability, where an arrow $x_i \rightarrow x_j$ shows the (j, i) entry and the brightness indicates the attention strength.

6.4 Subjective Evaluation of Music Generation

We invite people to subjectively rate the generation quality of the pre-trained model through a double-blind online survey. During the survey, the subjects listen to 8 groups of samples, each containing 4 generated segments together with a ground-truth human composition. The generated segments come from the four models (the same as in Table 1 excluding the fine-tuned model) with the same initial corruption state (60% of the total tokens) from the original piece. Both the order of groups and the sample order within each group are randomized. After listening

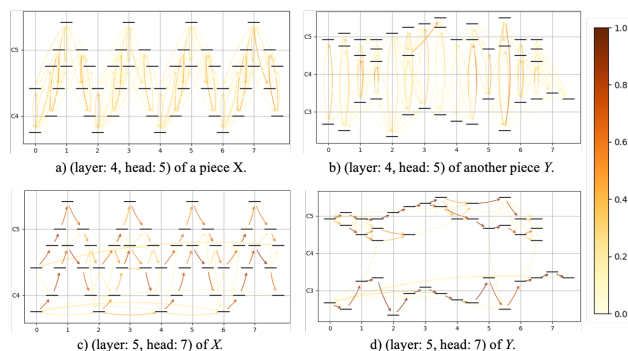


Figure 4. Visualization of attention matrices on selected self-attention layers in pre-trained MuseBERT.

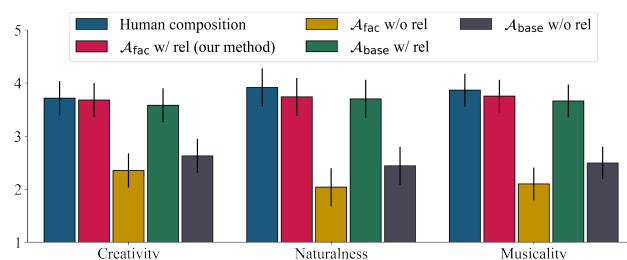


Figure 5. Subjective evaluation results on pre-trained MuseBERT.

to each sample, the subjects rate them based on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria: *creativity*, *naturalness* and *musicality*.

A total of 37 subjects (15 females and 22 males) with different musical backgrounds have completed the survey. The aggregated result (as shown in Figure 5) shows that the pre-trained models considering music relations (the well-defined models) are significantly better than the models without relations (the ill-defined models) (with p-value < 0.005), and are marginal to the human composition (with p-value > 0.05).

7. CONCLUSION

In conclusion, we contributed MuseBERT, a full treatment of BERT in the music domain as a powerful music understanding and generation model. The main novelty lies in the proposed *generalized* relative positional encoding, which successfully reveals the non-sequential, polyphonic structure of music and at the same time turns the masked language model objective into a well-defined controllable generative framework. Also, with the music-tailored factorized data representation, the controls are fine-grained. Furthermore, by fine-tuning MuseBERT, we demonstrate that the model is *general purpose*, capable of various downstream tasks such as chord analysis, accompaniment generation and refinement etc. as long as the constraints can be expressed via binary relations of different music attributes.

8. REFERENCES

- [1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [2] G. Hadjeres, F. Pachet, and F. Nielsen, “Deepbach: a steerable model for bach chorales generation,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 1362–1371.
- [3] Y. Yan, E. Lustig, J. VanderStel, and Z. Duan, “Part-invariant model for music generation and harmonization.” in *19th International Society for Music Information Retrieval*, 2018, pp. 204–210.
- [4] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, “Counterpoint by convolution,” *arXiv preprint arXiv:1903.07227*, 2019.
- [5] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music transformer,” *arXiv preprint arXiv:1809.04281*, 2018.
- [6] Y.-S. Huang and Y.-H. Yang, “Pop music transformer: Generating music with rhythm and harmony,” *arXiv preprint arXiv:2002.00212*, 2020.
- [7] I. Simon and S. Oore, “Performance rnn: Generating music with expressive timing and dynamics,” <https://magenta.tensorflow.org/performance-rnn>, 2017.
- [8] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, “Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs,” *arXiv preprint arXiv:2101.02402*, 2021.
- [9] J. Jiang, G. Xia, and T. Berg-Kirkpatrick, “Discovering music relations with sequential attention,” in *Proceedings of the 1st workshop on NLP for music and audio (NLP4MusA)*, 2020, pp. 1–5.
- [10] P. Shaw, J. Uszkoreit, and A. Vaswani, “Self-attention with relative position representations,” *arXiv preprint arXiv:1803.02155*, 2018.
- [11] C. Raffel and D. P. Ellis, “Intuitive analysis, creation and manipulation of midi data with pretty midi,” in *15th International Society for Music Information Retrieval conference late breaking and demo papers*, 2014, pp. 84–93.
- [12] M. S. Cuthbert and C. Ariza, “music21: A toolkit for computer-aided musicology and symbolic music data,” 2010.
- [13] “Garageband for mac.” [Online]. Available: <https://www.apple.com/mac/garageband/>
- [14] “Logic pro.” [Online]. Available: <https://www.apple.com/logic-pro/>
- [15] “Cubase guides you on your music production journey.” [Online]. Available: <https://new.steinberg.net/cubase/>
- [16] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, “Pianotree vae: Structured representation learning for polyphonic music,” *arXiv preprint arXiv:2008.07118*, 2020.
- [17] D. Jeong, T. Kwon, Y. Kim, and J. Nam, “Graph neural network for music score data and modeling expressive piano performance,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 3060–3070.
- [18] H. H. Mao, T. Shin, and G. Cottrell, “Deepj: Style-specific music generation,” in *IEEE 12th International Conference on Semantic Computing*. IEEE, 2018, pp. 377–382.
- [19] E. S. Koh, S. Dubnov, and D. Wright, “Rethinking recurrent latent variable model for music composition,” in *IEEE 20th International Workshop on Multimedia Signal Processing*. IEEE, 2018, pp. 1–6.
- [20] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *32nd AAAI Conference on Artificial Intelligence*, 2018.
- [21] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press, 1996.
- [22] F. Lerdahl, *Tonal pitch space*. Oxford University Press, 2004.
- [23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *International Conference on Machine Learning*, 2008, pp. 1096–1103.
- [25] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, “Set transformer: A framework for attention-based permutation-invariant neural networks,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 3744–3753.
- [26] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski, “Deep generative stochastic networks trainable by backprop,” in *International Conference on Machine Learning*. PMLR, 2014, pp. 226–234.
- [27] S. Russell and P. Norvig, “Artificial intelligence: a modern approach,” 2002.
- [28] Z. Wang, D. Wang, Y. Zhang, and G. Xia, “Learning interpretable representation for controllable polyphonic music generation,” *arXiv preprint arXiv:2008.07122*, 2020.

- [29] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, “Learning a latent space of multitrack measures,” *arXiv preprint arXiv:1806.00195*, 2018.
- [30] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, “Pop909: A pop-song dataset for music arrangement generation,” *arXiv preprint arXiv:2008.07142*, 2020.
- [31] D. Hendrycks and K. Gimpel, “Gaussian error linear units (gelus),” *arXiv preprint arXiv:1606.08415*, 2016.

SUPERVISED METRIC LEARNING FOR MUSIC STRUCTURE FEATURES

Ju-Chiang Wang Jordan B. L. Smith Wei-Tsung Lu Xuchen Song
ByteDance

{ju-chiang.wang, jordan.smith, weitsung.lu, xuchen.song}@bytedance.com

ABSTRACT

Music structure analysis (MSA) methods traditionally search for musically meaningful patterns in audio: homogeneity, repetition, novelty, and segment-length regularity. Hand-crafted audio features such as MFCCs or chromagrams are often used to elicit these patterns. However, with more annotations of section labels (e.g., verse, chorus, bridge) becoming available, one can use supervised feature learning to make these patterns even clearer and improve MSA performance. To this end, we take a supervised metric learning approach: we train a deep neural network to output embeddings that are near each other for two spectrogram inputs if both have the same section type (according to an annotation), and otherwise far apart. We propose a batch sampling scheme to ensure the labels in a training pair are interpreted meaningfully. The trained model extracts features that can be used in existing MSA algorithms. In evaluations with three datasets (HarmonixSet, SALAMI, and RWC), we demonstrate that using the proposed features can improve a traditional MSA algorithm significantly in both intra- and cross-dataset scenarios.

1. INTRODUCTION

In the field of Music Structure Analysis (MSA), most algorithms, including many recent and cutting-edge ones [1–3], use conventional features such as MFCCs and Pitch Class Profiles (PCPs). Devising a suitable feature for MSA is challenging, since so many aspects of music—including pitch, timbre, rhythm, and dynamics—impact the perception of structure [4]. Some methods have aimed to combine input from multiple features [5], but this requires care: MSA researchers have long been aware that structure at different timescales can be reflected best by different features (see, e.g., [6]).

A common story in MIR in the past decade is that using feature learning can improve performance on a task. Although this wave of work arrived late to MSA, we have already seen the benefits of supervised learning to model, for instance, ‘what boundaries sound like’ [7], or ‘what choruses sound like’ [8]. One drawback of these two meth-

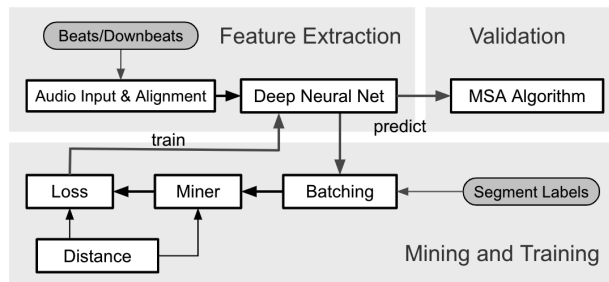


Figure 1. The training pipeline.

ods was that they were not compatible with existing MSA pipelines: new post-processing methods had to be conceived and implemented. Also, each one solved a limited version of MSA: segmentation and chorus detection, respectively. Developing a supervised approach that can explicitly minimize the losses of segmentation and labeling tasks at the same time remains a challenge.

In [9], *unsupervised* training was used to create a deep embedding of audio based on a triplet loss that aimed to reflect within-song similarity and contrast. The embedding vectors, treated as features, can directly replace traditional features in existing MSA pipelines, making it possible to leverage large, unannotated collections for MSA. This approach demonstrates the promise of learning features with a deep neural network (DNN) for MSA.

An unsupervised approach has so far been sensible, given how few annotations exist, and how expensive it is to collect more. However, the appeal of supervised learning has grown with the introduction of Harmonix Set [10], containing 912 annotated pop songs. Although Harmonix Set is smaller than SALAMI [11] (which has 1359 songs), it is much more consistent in terms of genre, which raises our hopes of learning a useful embedding. A model trained on SALAMI alone would have to adapt to the sound and structure of pop music, jazz standards, piano concertos, and more; a model trained on Harmonix Set has only to learn the sound and structure of pop songs. In short, the time is right to pursue a supervised approach.

In this paper, we propose to use supervised metric learning to train a DNN model that, for a given song, will embed audio fragments that lie in different sections far apart, and those from the same section close. (See Fig. 1 for an overview of the training pipeline.) This approach can help the model to capture the homogeneity and repetition characteristics of song structure with respect to the section labels (e.g., verse, chorus, and bridge). We also pro-



pose a batch sampling scheme to ensure the labels in a training pair are interpreted meaningfully. Given several relevant open-source packages that can help achieve this work, we introduce a modular pipeline including various metric learning methods and MSA algorithms, and make clear what parts of the system can be easily changed. By using the embeddings as features for an existing MSA algorithm, our supervised approach can support both segmentation and labeling tasks. In experiments, we leverage Harmonix Set, SALAMI, and RWC [12] to investigate the performance in intra- and cross-dataset scenarios.

2. RELATED WORK

Many MSA approaches (see [13] for an overview) are supervised in the sense of being tuned to a dataset—e.g., by setting a filter size according to the average segment duration in a corpus. An advanced version of this is [14], in which a recurrence matrix is transformed to match the statistics of a training dataset. However, supervised training has only been used in a few instances for MSA.

The first such method used supervision to learn a notion of ‘boundaryness’ directly from audio [7]; the method was refined to use a self-similarity lag matrix computed from audio [15]. Similarly, [8] used supervision to learn what characterizes boundaries as well as “chorusness” in audio, and used it in a system to predict the locations of choruses in songs, which is a subproblem of MSA. Although these 3 systems all have an ‘end-to-end’ flavor, in fact they required the invention of new custom pipelines to obtain estimates of structure, e.g., a peak-picking method to select the likeliest set of boundaries from a boundary probability curve. The post-processing is also complex in [16], in which a boundary fitness estimator similar to [15] is combined in a hybrid model with a trained segment length fitness estimator and a hand-crafted timbral homogeneity estimator. In our work, we aim to arrive at a feature representation that can be used with existing pipelines.

Taking the converse approach, [2] used supervision to model how traditional features (MFCCs, CQT, etc.) relate to music structure, using an LSTM combined with a Hidden semi-Markov Model. Since our approaches are complementary, a combined approach—inputting deep structure features to the LSTM-HSMM—may prove successful, and should be explored in future work.

As noted in the previous section, metric learning was previously applied to improve MSA by [9], but that work took an unsupervised approach: audio fragments in a piece were presumed to belong to the same class if they were near each other in time, and to different classes otherwise. This is a useful heuristic, but by design we expect it to use many false positive and false negative pairs in training. Also, that work did not report any evaluation on whether the learned embeddings could help with the segment labeling task, nor on the impact of many choices made in the system that could affect the results: the model architecture, loss function, and segmentation method. In this work, we conduct evaluations on the segmentation and labeling tasks, and investigate the impact of these design choices.

The supervision strategy in this work differs from prior art, and to our knowledge, this work represents the first attempt to develop supervised feature learning with a goal of improving existing MSA algorithms.

3. SYSTEM OVERVIEW

The training pipeline of our proposed system is illustrated in Fig. 1, and is divided into three stages: (1) feature extraction, (2) mining and training, and (3) validation.

The feature extraction stage consists of two modules. Following most state-of-the-art MSA algorithms [13], we synchronize the audio features with beat or downbeats. We use madmom [17] to estimate the beats and downbeats, and use these to create audio inputs to train a DNN; details of this are explained in Section 4.1. The network outputs the embedding vectors of a song for a subsequent algorithm to complete the task.

The mining and training stage covers four modules: batching, which we define ourselves, followed by miner, loss and distance modules, for which we use PML (pytorch-metric-learning¹), an open-source package with implementation options for each.

Batching: The training data are split into batches with a fixed size. To allow sensible comparisons among the training examples within a batch, we propose a scheme that ensures a batch only contains examples from the same song.

Miner: Given the embeddings and labels of examples in a batch, the miner provides an algorithm to pick *informative* training tuples (e.g., a pair having different labels but a large similarity) to compute the loss. Conventional metric learning methods just use all tuples in a batch (or, sample them uniformly) to train the model. As the batch size grows, using an informative subset can speed up convergence and provide a better model [18].

Loss: PML provides many well-known loss functions developed for deep metric learning, such as contrastive loss [19] and triplet loss [20]. We instead use MultiSimilarity loss [18] (see Section 4.4), a more general framework that unifies aspects of multiple weighting schemes that has not yet been used in an MIR application.

Distance: The distance metric defines the geometrical relationship among the output embeddings. Common metrics include Euclidean distance and cosine similarity.

For the validation stage, an MSA algorithm is adopted to generate the boundary and label outputs and validate the model learning status in terms of music structure analysis. The open-source package MSAF has implemented a representative sample of traditional algorithms [21]. An algorithm for a different task could be inserted here to tie the training to a different objective.

4. TECHNICAL DETAILS

4.1 Deep Neural Network Module

The input to the *DNN module* is defined to be a window (e.g. 8 second) of waveform audio, and the output to be a

¹ <https://github.com/KevinMusgrave/pytorch-metric-learning>

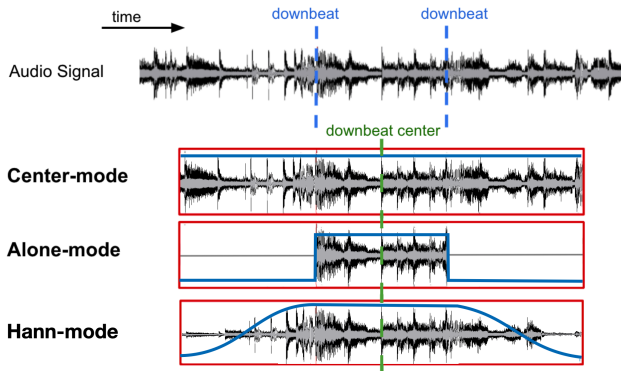


Figure 2. Each red box presents a window mode.

multi-dimensional embedding vector. We use a two-stage architecture in which the audio is transformed to a time-frequency representation before entering the DNN, but a fully end-to-end DNN would be possible.

In this work, we study two existing two-stage model architectures: Harmonic-CNN [22] and ResNet-50 [23]. These open-source architectures have shown good performance in general-purpose music audio classification (e.g., auto-tagging [22]), so we believe they can be trained to characterize useful information related to music sections. We replace their final two layers (which conventionally consist of a dense layer plus a sigmoid layer) with an *embedding module*, which in turn contains a linear layer, a leaky ReLU, a batch normalization, a linear layer, and a L2-normalization at the output. The input and output dimensions of this module are 256 and 100, respectively.

Any model with a similar purpose could be used for the DNN module in the proposed general framework. We have chosen the above architectures for their convenience, but they could be unsuitably complex given the small size of the available MSA training data. Developing a dedicated, optimal architecture is a task for future work.

4.2 Audio Input, Alignment, and Label

In order to synchronize the output embeddings with downbeats, we align the center of an input window to the center of each bar interval. The same procedure applies if aligning to beats. Typically, the input window is much longer than the duration of a bar, so there is additional context audio before and after the downbeat interval. We try three windowing methods that weight this context differently, also as illustrated in Fig. 2: *center-mode*, where the window is unaltered; *alone-mode*, where the context audio is zeroed out; and *Hann-mode*, where a Hann-shaped ramp from 0 to 1 is applied to the context audio. In our pilot studies, Hann-mode performed best, indicating that some context is useful, but the model should still focus on the signal around the center.

Annotations of structure contain, for each section, two timestamps defining the interval, and a label. These labels may be explicit functions (e.g., intro, verse, chorus) or abstract symbols (e.g., A, A', B, and C) indicating repetition. A training example is assigned with a label according to

Algorithm 1: One epoch of learning procedure.

Input: $\{\{s_i^j\}_{i=1}^{m_j}\}_{j=1}^M$, model Θ , and batch size β

Output: Learned model $\hat{\Theta}$

```

1 for  $j = 1$  to  $M$  do
2    $[s_{i'}^j]_{i'=1}^{m_j} \leftarrow$  shuffle sequence  $[s_i^j]_{i=1}^{m_j}$ 
3    $n \leftarrow \lceil m_j / \beta \rceil$  // number of batches
4   if  $n > 1$  then
5      $r \leftarrow n\beta - m_j$  // space in batch
6      $[s_{i'}^j]_{i'=1}^{n\beta} \leftarrow$  concat  $[s_{i'}^j]_{i'=1}^{m_j}$  and  $[s_{i'}^j]_{i'=1}^r$ 
7   for  $k = 1$  to  $n$  do
8      $\mathcal{B} \leftarrow \{s_{i'}^j\}, i' = \beta(k-1) : \min(\beta k, m_j)$ 
9      $\hat{\Theta} \leftarrow$  update  $\Theta$  with loss computed on  $\mathcal{B}$ 
    
```

the label of the exact center in the input audio. We denote a training example aligned with the i^{th} beat/downbeat of the j^{th} song by $s_i^j = (\mathbf{x}_i^j, y_i^j)$, where \mathbf{x} and y are the audio and label, respectively.

4.3 Batch Sampling Scheme

Let a dataset be denoted by $\{\{s_i^j\}_{i=1}^{m_j}\}_{j=1}^M$, where the j^{th} song has m_j examples. The proposed batch sampling scheme ensures that no cross-song examples are sampled in a batch. Therefore, when comparing any examples within a batch, the labels are meaningful for supervision. For example, we do not want a chorus fragment of song A to be compared with a chorus fragment of song B, since we have no *a priori* way to know whether these should be embedded near or far in the space.

Algorithm 1 gives the procedure for one epoch, i.e., one full pass of the dataset. We shuffle the original input sequence (line 2) to ensure that each batch is diverse, containing fragments from throughout the song. Lines 4–6 ensure, when more than one batch is needed for a song, the last batch is full by duplicating examples within the song. Once a batch is sampled (line 8), we can run a miner to select informative pairs from the batch to calculate the loss to update the model.

4.4 Miner and Loss

The MultiSimilarity framework [18] uses three types of similarities to estimate the importance of a potential pair: self-similarity (Sim-S), positive relative similarity (Sim-P), and negative relative similarity (Sim-N). The authors show that many existing deep metric learning methods only consider one of these types when designing a loss function. By considering all three types of similarities, MultiSimilarity offers stronger capability in weighting important pairs, setting new state-of-the-art performance in image retrieval. From our experiments, it also demonstrates better accuracy over other methods.

For an anchor s_i^j , an example s_k^j will lead to a positive pair if they have the same label (i.e., $y_i^j = y_k^j$), and a negative pair otherwise (i.e., $y_i^j \neq y_k^j$). At the minor phase, the algorithm calculates the Sim-P's for each positive/negative pair against an anchor, and selects the chal-

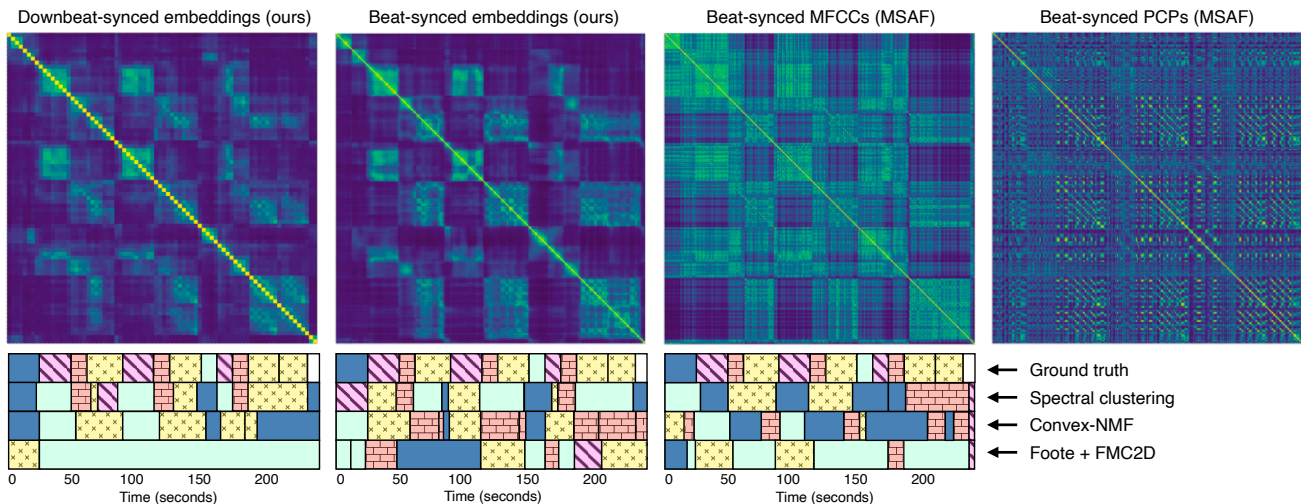


Figure 3. Four SSMs using different features for a test song *Avril Lavigne - Complicated*: two versions of the proposed embeddings (left), and two standard features (right). Below the SSMs are the segments and labels for the ground truth analysis, plus the estimated analyses from three algorithms. The block colors indicate the label clusters within a song.

lenging pairs when certain conditions are satisfied. At the loss phase, it uses the Sim-S’s and Sim-N’s to calculate the weights for the positive and negative pairs, respectively, where the weights are actually the gradients for updating the model. To summarize, MultiSimilarity aims to minimize intra-class dissimilarity at the mining stage, and to simultaneously maximize intra-class similarity and minimize inter-class similarity at the loss stage. In musical terms, the desired result is that fragments with the same section type will be embedded in tight clusters, and that clusters for different sections will be far from one another.

4.5 MSA Algorithms

The typical input to an MSA algorithm [21] is a sequence of feature vectors. Then, the algorithm outputs the predicted timestamps and an abstract label for each segment.

Fig. 3 presents four self-similarity matrices (SSMs) of the same test song using different features. We compute the pairwise Euclidean distance matrix and then apply a Gaussian kernel (see [1] for details) to derive the pairwise similarity. The left two matrices are based on a Harmonic-CNN trained with the MultiSimilarity miner and loss; the right two matrices are based on two traditional features, MFCCs and PCPs. We see that, compared to traditional features, the learned features can enhance the blocks considerably in the images, reducing the complexity faced by the MSA algorithm.

We picked three MSA algorithms to study here: spectral clustering (*scluster*) [1], convex-NMF (*cnmf*) [24], and *foote+fmc2d* (using Foote’s algorithm [25] for segmentation and *fmc2d* [26] for labeling). Note that each is based on analyzing some version of an SSM. As these algorithms were developed using traditional features, we need to adjust their default parameters in MSAF to be more suitable for a SSM with prominent but blurry blocks, rather than a sharp but noisy SSM typically treated with a low-pass filter to enhance the block structure. Also, some MSA

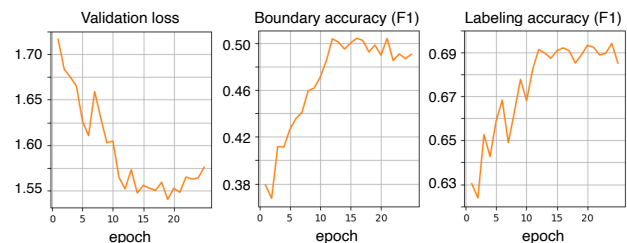


Figure 4. Validation loss vs. MSA (*scluster*) performance.

algorithms can be sensitive to temporal resolution, which prefers beat- to downbeat-synchronized features.

How the model training criterion improves an MSA algorithm can be explained in a theoretical way. For example, in *scluster*, small (e.g., one-bar) fragments of music that have the same labels are considered to be mutually connected in a sub-graph. When the metric learning loss is minimized, *scluster* is more likely to produce a clear sub-graph for each segment in a song, making the graph decomposition more accurate. As Fig. 4 illustrates, the evolution of the validation loss is consistent with the performance of *scluster* when it is fine-tuned to fully exploit the embeddings. This technique provides a guideline to adjust the parameters of an MSA algorithm for most cases.

5. EXPERIMENTS

5.1 Datasets

We use three datasets to study the performance: Harmonix Set [10], SALAMI [11], and RWC (Popular) [12].

The Harmonix Set covers a wide range of western popular music, including pop, electronic, hip-hop, rock, country, and metal. Each track is annotated with segment function labels and timestamps. The original audio to the annotations is not available, but a reference YouTube link is provided. We searched for the audio of the right version

and manually adjusted the annotations to ensure the labels and timestamps were sensible and aligned to the audio.

In SALAMI, some songs are annotated twice; we treat each annotation of a song separately, yielding 2243 annotated songs in total. We also use a subset with 445 annotated songs (from 274 unique songs) in the “popular” genre, called *SALAMI-pop*, for cross-dataset evaluation.

The Popular subset of RWC is composed of 100 tracks. There are two versions of the ground truth: one originally included with the dataset (AIST), and the other provided by INRIA [27]. The INRIA annotations contain boundaries but not segment labels.

Table 1 lists some properties of the datasets. “Num” is the number of annotated songs. The number of unique labels per song (“Uni”) ranges between 5.7 and 7.8, indicating that the segment labels are not too repetitive nor too diverse and thus can offer adequate supervision for metric learning. Additional statistics like the number of segments per song (“Seg”) and the mean duration per segment in second (“Dur”) are all within a proper range. Three of the datasets are employed in MIREX, so we can compare our systems with historical ones.

5.2 Evaluation Metrics

We focus on flat annotations (i.e. non-hierarchical) in our experiments. The evaluation metrics for MSA have been well-defined, and details can be found in [13]. We use the following: (1) *HR.5F*: F-measure of hit rate at 0.5 seconds; (2) *HR3F*: F-measure of hit rate at 3 seconds; (3) *PWF*: F-measure of pair-wise frame clustering; (4) *Sf*: F-measure of normalized entropy score. Hit rate measures how accurate the predicted boundaries are within a tolerance range (e.g., ± 0.5 seconds). Pair-wise frame clustering is related to the accuracy of segment labeling. Normalized entropy score gives an estimate about how much a labeling is over- or under-segmented.

5.3 Implementation Details

PyTorch 1.6 is used. We adopt audio windows of length 8 seconds, which we found better than 5 or 10 seconds. The audio is resampled at 16KHz and converted to log-scaled mel-spectrograms using 512-point FFTs with 50% overlap and 128 mel-components. We follow [28] and [29] to implement Harmonic-CNN and ResNet-50, respectively. For the miner and loss in pytorch-metric-learning, the default parameters suggested by the package are adopted. We employ the Adam optimizer to train the model, and monitor the *MSA summary score*, defined as $\frac{5}{14}(HR.5F) + \frac{2}{14}(HR3F) + \frac{4}{14}(PWF) + \frac{3}{14}(Sf)$, to determine the best model. The weights were chosen intuitively, but could be optimized in future work. We use a scheduled learning rate starting at 0.001, and then reduced by 20% if the score is not increased in two epochs. We train the models on a Tesla-V100-SXM2-32GB GPU with batch sizes of 128 and 240 for Harmonic-CNN and ResNet-50, respectively.

Regarding fine-tuning the parameters for MSAF, we run a simple grid search using a limited set of integer values on the validation set. As mentioned, the parameters are

Dataset	Num	Uni	Seg	Dur	MIREX
Harmonix Set	912	5.7	10.6	21.7	✗
SALAMI	2243	5.9	12.5	24.2	✓
SALAMI-pop	445	6.4	13.2	18.0	✗
RWC-AIST	100	7.8	15.3	15.2	✓
RWC-INRIA	100	-	15.3	15.0	✓

Table 1. Dataset and segment label statistics.

Model	System	HR.5F	HR3F	PWF	Sf
Base	cnmf/B	.183	.453	.498	.566
	ft+fmc2d/B	.242	.584	.536	.592
	scluster/B	.263	.547	.586	.641
Harm	cnmf/B/eu/mul	.352	.679	.647	.681
	ft+fmc2d/B/eu/mul	.395	.713	.580	.630
	scluster/B/eu/mul	.466	.728	.689	.737
ResN	cnmf/B/eu/mul	.339	.637	.618	.661
	ft+fmc2d/B/eu/mul	.373	.685	.572	.634
	scluster/D/eu/mul	.433	.720	.673	.728
Harm	scluster/D/eu/mul	.497	.738	.684	.743
	scluster/D/co/mul	.474	.706	.668	.727
	scluster/D/eu/tri	.454	.713	.669	.722
	scluster/D/co/tri	.448	.693	.659	.713
	scluster/D/eu/con	.435	.682	.635	.698

Table 2. Cross-validation result on the Harmonix Set. Top 9 rows: Comparison of different models { ‘Base’: baseline, ‘Harm’: Harmonic-CNN, ‘ResN’: ResNet-50} and MSA methods at beat-level (‘B’). Bottom 6 rows: comparison of different distances { ‘eu’: Euclidean, ‘co’: cosine} and losses { ‘mul’: MultiSimilarity, ‘tri’: TripletMargin, ‘con’: Contrastive} options, at downbeat-level (‘D’). ‘ft’ stands for Foote [25].

mostly different from the defaults. For instance, in scluster, we set (“evcc_smooth”, “rec_smooth”, “rec_width”) as (5, 3, 2), which were (9, 9, 9) by default. Also, scluster was designed to use separate timbral and harmonic features, but we use the same proposed features for both.

5.4 Result and Discussion

We present three sets of evaluations: (1) a comparison of many versions of our pipeline to establish the impact of the choice of modules; (2) a cross-dataset evaluation; and (3) a comparison of our system with past MIREX submissions.

First, we study the effect of several options for the proposed pipeline: (1) beat or downbeat alignment for input audio; (2) distance metric for the learned features; (3) miner and loss for metric learning. For (3), we use the proposed MultiSimilarity approach and TripletMargin miner and loss [20]; we also test Contrastive loss [19] with a BaseMiner, which samples pairs uniformly. Each version of the feature embedding is trained and tested on the Harmonix Set using 4-fold cross-validation.

We compare the success of three MSA algorithms when using the proposed features and when using conventional features. In all cases, we synchronize the features to the beats/downbeats estimated by madmom; for the proposed features, we use the ground-truth beats/downbeats for training and the estimated ones for testing. For a fair comparison, we fine-tune the algorithm parameters for each algorithm-feature combination (including the conven-

Model	System	HR.5F	HR3F	PWF	Sf
Base	cnmf/B	.259	.506	.485	.521
	ft+fmc2d/B	.319	.593	.521	.551
	scluster/B	.305	.553	.545	.572
Harm	cnmf/B/eu/mul	.301	.573	.588	.601
	ft+fmc2d/B/eu/mul	.358	.599	.538	.581
	scluster/B/eu/mul	.378	.613	.621	.644
	scluster/D/eu/mul	.447	.623	.615	.653

Table 3. Cross-dataset result on SALAMI-pop (trained on Harmonix Set); and “ft” stands for Foote.

System	AIST				INRIA	
	HR.5F	HR3F	PWF	Sf	HR.5F	HR3F
OLDA+fmc2d	.255	.554	.526	.613	.381	.604
SMGA2 (2012)	.246	.700	.688	.733	.252	.759
GS1 (2015)	.506	.715	.542	.692	.697	.793
scluster/D/eu/mul	.438	.653	.704	.739	.563	.677

Table 4. MIREX-RWC (cross-dataset) result.

tional features) by running a grid search and optimizing the MSA summary score on the training set.

Table 2 presents the results. They show that every MSA algorithm is improved by using the learned features instead of the baseline ones, by a wide margin: HR.5F nearly doubles in most cases when switching to learned features. The performance differences for each algorithm (e.g., ‘Base scluster/B’ versus ‘Harm scluster/B/eu/mul’) are significant with p-values $< 10^{-5}$ for every metric. The top MSA algorithm overall is scluster, which performs the best on boundary hit rate when synchronized with downbeats, but performs slightly better on PWF when synchronized to beats. Comparing the two architectures, Harmonic-CNN performs better than ResNet-50 in general, perhaps because the deeper ResNet model requires more data.

Regarding the other training settings, we find that using Euclidean distance was consistently better than using cosine distance, and that the MultiSimilarity loss gave consistently better results than the other loss functions. While running the experiments, we notice that with Euclidean distance, the validation loss evolved in a more stable way.

Second, we study cross-dataset performance by using the best trained model on Harmonix Set to make predictions for the songs in SALAMI-pop. This tests the model ability to avoid overfitting to one style of annotations. In Table 3, we see that the scluster algorithm again performs the best, and again improves significantly when using the learned features (p-value $< 10^{-10}$). However, the improvement margins are smaller for cnmf and foote+fmc2d (e.g., for cnmf, HR.5F increases by 0.042; before, it increased by 0.169). Perhaps the MSA parameters for these two algorithms are over-tuned to the training data; or, it may be that the learned features overfit the style of pop in Harmonix Set, but that scluster is more robust to this.

Finally, we collect previous MIREX results to compare our system to others. For the RWC (popular) task, we use the same model (trained on Harmonix Set) from the previous experiment on SALAMI-pop. The results are shown in Table 4 alongside those of three of the strongest MIREX submissions. We omit some, like SMGA1, that are re-

System	HR.5F	HR3F	PWF	Sf
cnmf (2016)	.228	.427	.527	.543
foote+fmc2d (2016)	.244	.503	.463	.549
scluster (2016)	.255	.420	.472	.608
OLDA+fmc2d [14, 31]	.299	.486	.471	.559
SMGA1 (2012) [32]	.192	.492	.581	.648
Segmentino [33, 34]	.209	.475	.551	.612
GS1 (2015) [15, 35]	.541	.623	.505	.650
cnmf/D/eu/mul	.318	.506	.587	.578
foote+fmc2d/B/eu/mul	.289	.519	.558	.563
scluster/D/eu/mul	.356	.553	.568	.613

Table 5. MIREX-SALAMI result.

lated to or based on the same approach as others listed but perform worse. Our system can outperform the state-of-the-art (SMGA2) in terms of PWF and Sf. Regarding its segmentation performance, it is still competitive, outperforming OLDA (the top-performing segmenter offered by MSAF) by a large margin (HR.5F/3F).

For the SALAMI task, the identity of the songs used in MIREX is private, but 487 songs (with 921 annotations) have been identified [30]. We use this portion as the test set, and the remainder of SALAMI (1322 annotations of 872 songs) as the sole training and validation set. The results are shown in Table 5, along with other MIREX competitors, including OLDA+FMC2D, SMGA1, and GS1 (which uses a CNN trained to directly model the boundaries [15]). As SALAMI is more diverse than Harmonix Set, the model sees fewer examples per style compared to when it was trained on Harmonix Set. Thereby, we can expect the learned features to be less successful. However, we once again see that each model in MSAF is improved on all metrics when using the learned features, particularly in terms of PWF. In fact, our model boosts cnmf—already third-best among the baseline algorithms shown here—to outperform the state-of-the-art (SMGA1).

The MSAF algorithms are improved with the learned features, but they still lag behind GS1. This is reasonable, since the training of that model directly connects to the loss of boundary prediction, and ours does not. Nonetheless, “scluster/D/eu/mul” can outperform all the other systems except GS1 by a large margin on both HR.5F and HR3F.

6. CONCLUSION AND FUTURE WORK

We have presented a modular training pipeline to learn the deep structure features for music audio. The pipeline consists of audio pre-processing, DNN model, metric learning module, and MSA algorithm. We have explained the functionality for each component and demonstrated the effectiveness of different module combinations. In experiments, we have found that using the learned features can improve an MSA algorithm significantly.

However, the model is not yet fully end-to-end: the MSA outputs (boundaries and labels) are not directly back-propagated to the DNN model. We plan to explore ways to change this in future work—e.g., by exploring self-attention models like the Transformer [36, 37] to build a deep model that directly outputs the segment clusters. This would eliminate the need to fine-tune MSA parameters.

7. REFERENCES

- [1] B. McFee and D. Ellis, “Analyzing song structure with spectral clustering,” in *ISMIR*, 2014, pp. 405–410.
- [2] G. Shibata, R. Nishikimi, and K. Yoshii, “Music structure analysis based on an LSTM-HSMM hybrid model,” in *ISMIR*, 2020, pp. 15–22.
- [3] C. Wang and G. J. Mysore, “Structural segmentation with the variable Markov oracle and boundary adjustment,” in *Proc. ICASSP*, 2016, pp. 291–295.
- [4] M. J. Bruderer, M. F. McKinney, and A. Kohlrausch, “The perception of structural boundaries in melody lines of western popular music,” *Musicae Scientiae*, vol. 13, no. 2, pp. 273–313, 2009.
- [5] J. de Berardinis, M. Vamvakaris, A. Cangelosi, and E. Coutinho, “Unveiling the hierarchical structure of music by multi-resolution community detection,” *Trans. ISMIR*, vol. 3, no. 1, pp. 82–97, 2020.
- [6] T. Jehan, “Hierarchical multi-class self similarities,” in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2005, pp. 311–314.
- [7] K. Ullrich, J. Schlüter, and T. Grill, “Boundary detection in music structure analysis using convolutional neural networks,” in *ISMIR*, 2014, pp. 417–422.
- [8] J.-C. Wang, J. B. L. Smith, J. Chen, X. Song, and Y. Wang, “Supervised chorus detection for popular music using convolutional neural network and multi-task learning,” in *Proc. ICASSP*, 2021, pp. 566–570.
- [9] M. C. McCallum, “Unsupervised learning of deep features for music segmentation,” in *Proc. ICASSP*, 2019, pp. 346–350.
- [10] O. Nieto, M. McCallum, M. Davies, A. Robertson, A. Stark, and E. Egozy, “The Harmonix Set: Beats, downbeats, and functional segment annotations of western popular music,” in *ISMIR*, 2019, pp. 565–572.
- [11] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. D. Roure, and J. S. Downie, “Design and creation of a large-scale database of structural annotations,” in *ISMIR*, 2011, pp. 555–560.
- [12] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC Music Database: Popular, classical and jazz music databases,” in *ISMIR*, 2002, pp. 287–288.
- [13] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, “Audio-based music structure analysis: Current trends, open challenges, and applications,” *Trans. ISMIR*, vol. 3, no. 1, 2020.
- [14] B. McFee and D. P. Ellis, “Learning to segment songs with ordinal linear discriminant analysis,” in *Proc. ICASSP*, 2014, pp. 5197–5201.
- [15] T. Grill and J. Schlüter, “Music boundary detection using neural networks on combined features and two-level annotations,” in *ISMIR*, 2015, pp. 531–537.
- [16] A. Maezawa, “Music boundary detection based on a hybrid deep model of novelty, homogeneity, repetition and duration,” in *Proc. ICASSP*, 2019, pp. 206–210.
- [17] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: a new Python Audio and Music Signal Processing Library,” in *Proceedings of the ACM International Conference on Multimedia*, 2016, pp. 1174–1178.
- [18] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, “Multi-similarity loss with general pair weighting for deep metric learning,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5022–5030.
- [19] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1735–1742.
- [20] E. Hoffer and N. Ailon, “Deep metric learning using triplet network,” in *International workshop on similarity-based pattern recognition*. Springer, 2015, pp. 84–92.
- [21] O. Nieto and J. P. Bello, “Systematic exploration of computational music structure research,” in *ISMIR*, 2016, pp. 547–553.
- [22] M. Won, S. Chun, O. Nieto, and X. Serra, “Data-driven harmonic filters for audio representation learning,” in *Proc. ICASSP*, 2020, pp. 536–540.
- [23] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in *European conference on computer vision*, 2016, pp. 630–645.
- [24] O. Nieto and T. Jehan, “Convex non-negative matrix factorization for automatic music structure identification,” in *Proc. ICASSP*, 2013, pp. 236–240.
- [25] J. Foote, “Automatic audio segmentation using a measure of audio novelty,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2000, pp. 452–455.
- [26] O. Nieto and J. P. Bello, “Music segment similarity using 2D-Fourier magnitude coefficients,” in *Proc. ICASSP*, 2014, pp. 664–668.
- [27] F. Bimbot, O. Le Blouch, G. Sargent, and E. Vincent, “Decomposition into autonomous and comparable blocks: a structural description of music pieces,” 2010.
- [28] M. Won, <https://github.com/minzwon/sota-music-tagging-models/blob/master/training/model.py>, Last accessed on May 5, 2021.

- [29] TorchVision, <https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py>, Last accessed on May 5, 2021.
- [30] J. Schlüter, http://www.ofai.at/research/impml/projects/audiostreams/ismir2014/salami_ids.txt, Last accessed on May 5, 2021.
- [31] O. Nieto, “MIREX: MSAF v0.1.0 submission,” *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)*, 2016.
- [32] J. Serra *et al.*, “The importance of detecting boundaries in music structure annotation,” *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)*, 2012.
- [33] M. Mauch, K. C. Noland, and S. Dixon, “Using musical structure to enhance automatic chord transcription,” in *ISMIR*, 2009, pp. 231–236.
- [34] C. Cannam, E. Benetos, M. Mauch, M. E. Davies, S. Dixon, C. Landone, K. Noland, and D. Stowell, “MIREX 2016: Vamp plugins from the Centre for Digital Music,” *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)*, 2016.
- [35] T. Grill and J. Schlüter, “Structural segmentation with convolutional neural networks mirex submission,” *Proceedings of the Music Information Retrieval Evaluation eXchange (MIREX)*, p. 3, 2015.
- [36] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [37] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, “SpecTNT: a time-frequency transformer for music audio,” in *ISMIR*, 2021.

LEARNING LONG-TERM MUSIC REPRESENTATIONS VIA HIERARCHICAL CONTEXTUAL CONSTRAINTS

Shiqi Wei^{1,2} Gus Xia²

¹ School of Data Science, Fudan University

² Music X Lab, Computer Science Department, New York University Shanghai

sqwei19@fudan.edu.cn, gxia@nyu.edu

ABSTRACT

Learning symbolic music representations, especially disentangled representations with probabilistic interpretations, has been shown to benefit both music understanding and generation. However, most models are only applicable to short-term music, while learning long-term music representations remains a challenging task. We have seen several studies attempting to learn hierarchical representations directly in an end-to-end manner, but these models have not been able to achieve the desired results and the training process is not stable. In this paper, we propose a novel approach to learn long-term symbolic music representations through contextual constraints. First, we use contrastive learning to pre-train a long-term representation by constraining its difference from the short-term representation (extracted by an off-the-shelf model). Then, we fine-tune the long-term representation by a hierarchical prediction model such that a good long-term representation (e.g., an 8-bar representation) can reconstruct the corresponding short-term ones (e.g., the 2-bar representations within the 8-bar range). Experiments show that our method stabilizes the training and the fine-tuning steps. In addition, the designed contextual constraints benefit both reconstruction and disentanglement, significantly outperforming the baselines.

1. INTRODUCTION

Deep music representation learning have been proven to be a powerful tool for high-quality symbolic music generation [1]. The learned representations can be directly fed into downstream predictive models such as LSTMs [2] and Transformers [3] to achieve more coherent results than note-based or event-based generation [4–6]. Furthermore, when a representation learning model has a probability interpretation, the representation can then be easily interpolated or resampled to create new music pieces. Recently, several studies further disentangle music representations into interpretable factors (such as pitch, rhythm, chord and texture) to achieve a more controllable and interactive music generation [5, 7, 8]. For example, we can keep the pitch

factor of a melody while resampling its rhythm factor to achieve theme variation. We can also interpolate the pitch factor for a smooth music morphing [7].

Despite the above mentioned progress [9–11], most existing work applies only to short music segments with a length of several beats, while learning long-term representations remains a challenging task. In particular, studies have shown that even for monophonic melodies, "flat" model designs (e.g., using long-range sequential encoders) have difficulty remembering a complete music phrase at once. Some other studies have attempted to solve this problem by building another layer of hierarchy on top of short-range flat models, learning short-term and long-term representations simultaneously in an end-to-end manner [1, 12]. However, as the model expressivity increases with the number of layers, models also become much more difficult to train.

We argue that the main problem with current methods is the lack of proper inductive bias, and in this paper we propose a new method for learning long-term, phrase-level symbolic music representations through contextual constraints. The method consists of two stages pre-training and fine-tuning, with two steps in each stage. In the pre-training stage, we first adopt EC²-VAE [7] to learn bar-level, disentangled latent pitch and rhythm representations. Then, we apply the same model to learn phrase-level representation but with contrastive losses to constrain the difference between phrase-level and bar-level representations. It is indeed difficult to learn phrase-level representations directly using bar-level models, but the additional contrastive constraint can serve as a useful inductive bias to help find a reasonable solution that can subsequently be improved by fine-tuning. During the fine-tuning stage, we replace the pre-trained decoder with a hierarchical prediction model that forces the phrase-level representation to reconstruct the bar-level ones. This is achieved by first tuning only the new hierarchical decoder (while fixing the pre-trained encoder) and then tuning the whole network. During these two steps, structured contrastive loss is applied to stabilize the learning process.

Experiments show that the proposed method significantly outperforms the baselines and successfully learns disentangled pitch and rhythm representations for 8-bar long phrases (32 beats in 4/4 meter) without increasing the latent dimensionality. To our knowledge, this is also the first generative model that achieves phrase-level composition style transfer, latent factor interpolation, and theme



variation. In sum, our contributions are as follows:

- We demonstrate the importance of structured contextual constraints in learning long-term disentangled representations. Our approach only requires reasonable amount of data to train and could learn compact latent representation.
- We show that the proposed Structured InfoNCE loss effectively expresses the contextual constraints, stabilizes the training of long-range models and helps the model converge faster.
- Our model achieves phrase-level music style transfer, latent factor interpolation, and theme variation.

2. RELATED WORK

We review two realms of research related to our work on long-term music-representation learning: contrastive learning, which is the main method to stabilize the training process, and hierarchical music modeling, which is related to our fine-tuning model.

2.1 Contrastive Learning

Contrastive learning (CL) is an efficient method in self-supervised learning [13–15], serving as regularization to latent representations. For example, NCE-based contrastive losses [16, 17] have been widely used and achieved good results in natural language processing. Contrastive predictive coding (CPC) [18] and Deep Infomax (DIM) [19] explore the relation between minimizing a contrastive learning loss and maximizing a lower bound of the mutual information. In DIM, global feature is connected with local feature to learn more abstract and informative representations.

2.2 Hierarchical Music Representation Learning

The hierarchical nature of music has been studied for a long time [20–23]. Recently, we see some efforts on learning long-term music representations using hierarchical modeling [12, 24, 25]. The basic idea is that since a flat model design can only effectively learn short-term representations, we can stack more layers on top of the short-term representations module for long-term representations. Existing works include MusicVAE [1], Music Transformer VAE [12], Jukebox [26], etc. However, experiments show that unless we have a huge amount of data, the model is in general very difficult to train. In this study, we provide a two-stage algorithm with contrastive loss as a better learning strategy. Also, no model so far has achieved disentanglement for long-term representation as done in this study.

3. METHODOLOGY

In this section, we introduce our algorithm in detail. Conceptually, it consists of two stages, each with two steps. The first stage is *pre-training*:

- In step 1, we simply adopt EC²-VAE [7], an existing music representation disentanglement model, to extract short-term pitch and rhythm representations.

- In step 2, we build Long-EC²-VAE, a long-term version of the model and train it with an extra contextual constraint using the proposed *Structured InfoNCE* loss. Intuitively, this loss prevents the learned long-term representations from deviating too far from corresponding well-trained short-term representations.

The second stage is *fine-tuning*, in which we build a hierarchical representation-learning model by combining the encoder of Long-EC²-VAE with a hierarchical decoder. We name this model after Hierarchical-EC²-VAE.

- In step 1, we only train the hierarchical decoder to ensure the predictive power of the long-term representation.
- In step 2, we train the whole hierarchical network for a better long-term pitch-rhythm disentanglement.

3.1 Pre-training by Contrastive Learning

The model of the pre-training stage, Long-EC²-VAE, is shown in Figure 1. It is built upon an off-the-shelf music representation model, EC²-VAE [7], which can effectively disentangle pitch and rhythm factors for short music segments by cutting the latent representation into two parts and pairing one part with a local rhythm decoder. In Fig-

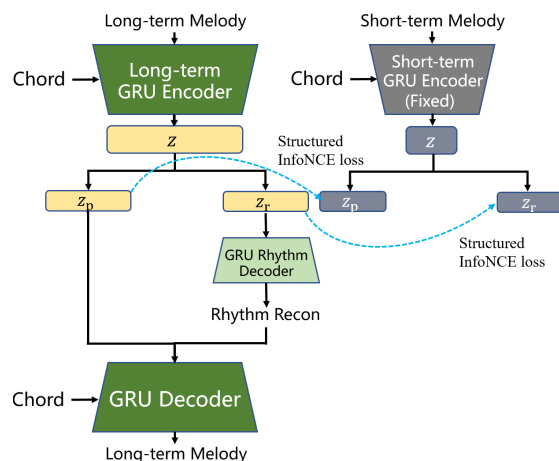


Figure 1: The model architecture of Long-EC²-VAE in the pre-training stage, where the right-hand-side is short-term model with parameter fixed and the left-hand-side is the long-term model. The dotted lines denote contrastive losses, whose weighting matrices are joined optimized with the parameters on the left-hand-side networks.

ure 1, the right-hand-side part is a literal copy of the EC²-VAE encoder (with parameters fixed) to extract short-term representations, while the left-hand-side part is a simple adaptation of EC²-VAE for long-term music by lengthening its temporal receptive field. Note that the left part alone is not able to learn long-term representations, and our goal is to assist it using contrastive learning. Formally, the loss function of Long-EC²-VAE is:

$$\mathcal{L} = \mathcal{L}_{\text{Long-EC}^2\text{-VAE}} + \mathcal{L}_{\text{Structured InfoNCE}}, \quad (1)$$

where $\mathcal{L}_{\text{Long-EC}^2\text{-VAE}}$ is the same as in the original EC²-VAE model (which contains the KL loss, the rhythm loss

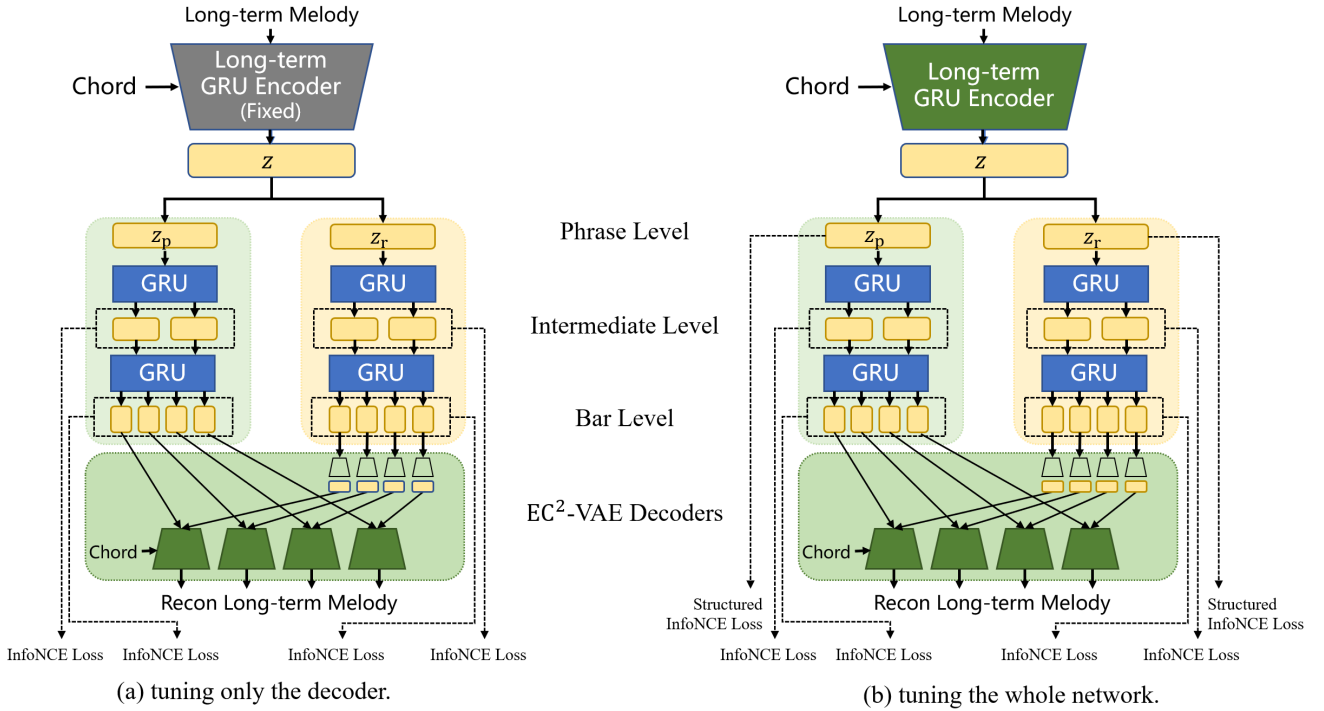


Figure 2: The model architecture of Hierarchical-EC²-VAE in the fine-tuning stage. The training follows two steps.

and the overall reconstruction loss). The Structured InfoNCE loss expresses the contextual constraint. It is developed from InfoNCE [18] loss, and it is *structured* since the compared representation pairs are extracted from music segments of different length, one is long term and the other is short term. Formally:

$$\mathcal{L}_{\text{Structured InfoNCE}} = -\ln \frac{\exp\left(z_{L,f}^T W \hat{z}_{S,f}^+ / \tau\right)}{\exp\left(z_{L,f}^T W \hat{z}_{S,f}^+ / \tau\right) + \sum_{i=1}^K \exp\left(z_{L,f}^T W \hat{z}_{S,f}^- / \tau\right)}, \quad (2)$$

where $z_{L,f}$ and W are the normalized long-term representations and weighing matrix we need to learn. $f = \{p, r\}$ indicates whether it is the pitch or rhythm factor. Likewise, we use $\hat{z}_{S,f}$ to denote the short-term representations extracted by right-hand-side model. K and τ are hyper-parameters. K is the amount negative samples and τ is the temperature parameter.

In specific, the short-term melodies are half as long as long-term ones. The positive samples $\hat{z}_{S,f}^+$ are in the cases that the corresponding short-term melody is a part of the long-term melody and f takes the same value as in $z_{L,f}$, while the negative samples are not in this case. Also, the long-term and short-term representations share the *same* dimensionality.

3.2 Fine-tuning with Hierarchical Generation

Figure 2 shows the architecture of the fine-tuning model, Hierarchical-EC²-VAE, where the two subfigures illustrate the two training steps. Here, the encoder design is the same as in the Long-EC²-VAE model, while the decoder is a hierarchical predictive model with three layers. The first two layers are new designed and the last layer is an aggregation

of several EC²-VAE decoders sharing the same parameters. Given the disentangled long-term (phrase-level) representations, it first decodes intermediate-level representations, then decodes bar-level representations, and finally reconstructs concrete rhythm and music tokens.

Compared to the phrase-level representation, the temporal receptive fields of the intermediate-level representations all shrink to a half, but at the same time their number doubles in order to cover the same range of music. The same relationship holds between intermediate and bar-level representations. In particular, a phrase means 8 bar (in 4/4 meter, 32 beats) in our design, so that the intermediate-level and bar-level mean 4-bar and 2-bar melody segments (a length which the original EC²-VAE model can handle), respectively. All levels of latent representations share the same dimensionality.

In the first step of training (Figure 2(a)), the encoder is a literal copy from the Long-EC²-VAE model and we only train the hierarchical decoder. Formally, the loss function is:

$$\mathcal{L}_{\text{step1}} = \mathcal{L}_{\text{Hierarchical-EC}^2\text{-VAE Decoder}} + \mathcal{L}_{\text{InfoNCE}}, \quad (3)$$

where the first term refers to the reconstruction losses adopted from the EC²-VAE model, and the second term is defined as:

$$\mathcal{L}_{\text{InfoNCE}} = -\ln \frac{\exp\left(z_{l,f}^T W \hat{z}_{l,f}^+ / \tau\right)}{\exp\left(z_{l,f}^T W \hat{z}_{l,f}^+ / \tau\right) + \sum_{i=1}^K \exp\left(z_{l,f}^T W \hat{z}_{l,f}^- / \tau\right)}, \quad (4)$$

where $z_{l,f}$ are the normalized hierarchical representations we need to learn with $l = \{\text{intermediate, bar}\}$ indicating the level of representation and other notations follow the

same meaning as in Eq.(2). Here, both positive $\hat{z}_{l,f}^+$ and negative $\hat{z}_{l,f}^-$ samples are normalized representations computed from a pre-trained EC²-VAE, in which the positive samples are in the cases that the $\hat{z}_{l,f}^+$ and $z_{l,f}$ are computed based on the same music segment and have the same value of l and f , while $\hat{z}_{l,f}^-$ are not in this case.

After the first step achieves a reasonable accuracy, we proceed to step 2 (Figure 2(b)), unfreezing the encoder and training the whole hierarchical representation-learning model with:

$$\mathcal{L}_{\text{step2}} = \mathcal{L}_{\text{step1}} + \mathcal{L}_{\text{Structured InfoNCE}} + \beta \mathcal{L}_{\text{KL phrase}}, \quad (5)$$

where the first two terms are defined in Eq. (3) and Eq. (2) respectively. $L_{\text{KL phrase}}$ is KL divergence to only regularize the phrase-level representations by a normal distribution. The value β controls the degree of KL divergence penalty.

4. EXPERIMENTS

4.1 Dataset and data format

We train our model on Nottingham Database [27] and POP909 database [28]. Our dataset contains 2154 melodies (at song level) in total. We randomly split these pieces into 2 subsets: 90% songs for training and 10% songs for test. The data format is designed as the same as in [7] in which 4 bar or 8 bar melodies are formalized as sequences of 130-dimensional one-hot embedding vectors and 16-beat and 32-beat rhythm pattern is represented by a sequence of 3-dimensional one-hot embedding vectors. Each vector in the melody sequence denotes a $\frac{1}{4}$ -beat unit. The first 128 dimensions of this vector denote 128 MIDI-format pitches from 0 to 127, the 129th dimension is the holding state for longer note duration, and the last dimension is kept for rest. The three dimensions of rhythm pattern vectors represent the onset of any pitch, a holding state, and rest, respectively.

4.2 Implementation Details

All of our models are trained using Adam optimizer [29] with a scheduled learning rate from 1e-3 to 1e-5. The batch size is 128 in the pre-training stage and is 64 in the fine-tuning stage. We do normalization on representations in Eq.(2) and (4) to make the training process more stable. The representations fed into decoders are original representation without normalization.

4.2.1 Pre-training

In the pre-training stage, we simply adopt the structure of EC²-VAE [7] to model 4 bar and 8 bar EC²-VAE. Each model comprises an encoder with a bi-directional GRU layer, a rhythm decoder with a GRU layer, and a global decoder with a GRU layer. We set the hidden dimension of the GRU in the encoder and decoders to 2048. The latent dimension is 128 for disentangled pitch representations and 128 for disentangled rhythm representations for each range model. For $\mathcal{L}_{\text{Structured InfoNCE}}$ depicted in Eq. (2), we set K to 512 and τ to 1. The positive samples for Eq. (2)

and Eq. (4) are the representations of 1-4th, 3-6th and 5-8th bar from well-trained 4 bar EC²-VAE. Actually, even when training the 4-bar EC²-VAE (right-hand side of Figure 1), we use a similar constrastive loss as in Eq. (2) where the positive samples are representations of 1-2th, 2-3th, 3-4th bar from well-trained 2 bar (original) EC²-VAE [7].

4.2.2 Fine-tuning

Hierarchical-EC²-VAE model consists of a long-term (8 bar) EC²-VAE encoder, 4 GRU layers, and an aggregation of 2 bar EC²-VAE decoders. We first train the hierarchical model with fixed 8 bar EC²-VAE encoder from pre-trained stage for around 25 epochs. Then we train the whole model without fixing parameters. We set the hidden dimension of 4 GRU layers to 1024. We set K to 256 and τ to 1 for both Structured InfoNCE loss and InfoNCE loss and set β to 0.1 in Eq. (5).

4.3 Objective Evaluation

We objectively evaluate the model in terms of reconstruction accuracy, training stability, and disentanglement.

4.3.1 Reconstruction Accuracy

Table 1 shows that the reconstruction accuracy of the proposed models (2nd an 3rd rows) significantly outperform the baseline, a vanilla EC²-VAE applied to 8-bar melody (first row). The last two rows show the results of two ablation settings of Hierarchical-EC²-VAE: one without the contrastive loss and the other without first fixing the parameters of encoder and directly train the model end-to-end. We see that the proposed Structured InfoNCE or InfoNCE losses play a vital role for an accurate reconstruction and the two-step training strategy improves the result marginally.

Method	Recon.Acc	Rhythm Recon.Acc
Baseline	0.772	0.847
Long-EC ² -VAE	0.992	0.995
H-EC ² -VAE(ours)	0.997	0.995
H-EC ² -VAE(w-o CL)	0.584	0.599
H-EC ² -VAE(w-o fixed)	0.991	0.989

Table 1: A comparison on reconstruction accuracy of different models.

4.3.2 Training stability

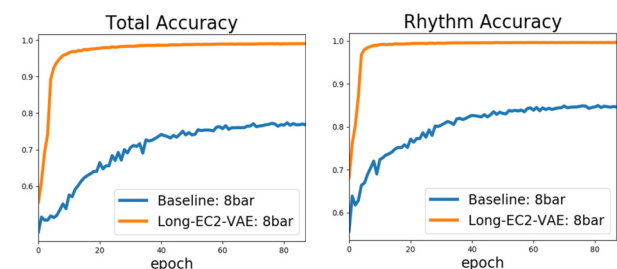
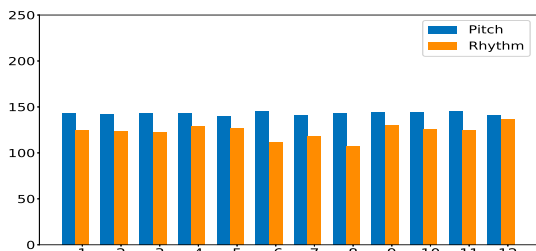


Figure 3: Experimental results of overall reconstruction and rhythm accuracy on the test set.

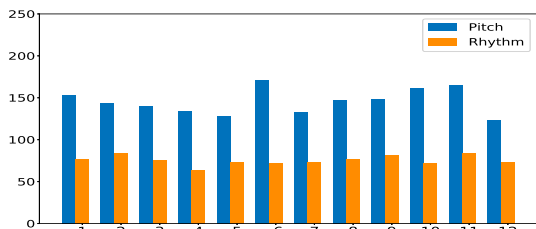
Comparing the accuracy curves of the proposed Long-EC²-VAE with the baseline as illustrated in Figure 3, we find that the proposed Long-EC²-VAE converges more quickly during training. This indicates that the proposed training strategy leads to a better initialization and makes the performance of the model fluctuate less during training.

4.3.3 Disentanglement Evaluation

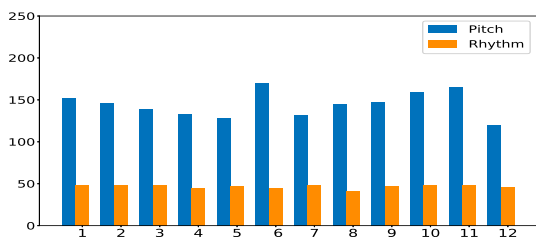
We evaluate the disentanglement performance of models using a disentanglement evaluation method adopted in [7] and [30]. The method randomly transposes all the notes of the input data by $i(i \in [1, 12])$ semitones while keeping the rhythm and underlying chord unchanged and then measures the variation of disentangled representations. We denote $\Sigma|\Delta z_p|$ and $\Sigma|\Delta z_r|$ as the variation of z_p and z_r .



(a) Baseline model (8 bar)



(b) Long-EC²-VAE model (8 bar)



(c) Hierarchical-EC²-VAE model (8 bar)

Figure 4: The comparison between $\Sigma|\Delta z_p|$ and $\Sigma|\Delta z_r|$ after transposition. The numbers show the pitch augmented by 12 semitones in each sub-figure from left to right.

As shown in Figure 4, values of $\Sigma|\Delta z_p|$ of the proposed Hierarchical-EC²-VAE are relatively high while $\Sigma|\Delta z_r|$ maintains in a significantly low level. This indicates that the pitch and rhythm representations of the proposed Hierarchical-EC²-VAE are well-disentangled as the change of notes has a tiny impact on z_r . Similarly, we can intuitively find in the figure that the disentanglement performance of the proposed Hierarchical-EC²-VAE is much better than the baseline and also outperforms the proposed Long-EC²-VAE.

4.4 Music generative examples

In this section, we show some music generation results by manipulating the disentangled phrase-level pitch and rhythm representations in three different ways: style transfer via swapping the representation, rhythm morphing via interpolating the representation, and theme variation via representation posterior sampling.

4.4.1 Phrase-level composition style transfer

We cross-swap the disentangled pitch and rhythm factors z_p and z_r of two 8-bar melodies A and B and then obtain generative pieces C and D. The results are shown in Figure 5, in which we see that both of the two generative pieces perfectly inherit target rhythm patterns. Besides, these generative melodies vary slightly from the source melody and these variations tend to sound creative, i.e. the appearance of embellished notes.



Figure 5: Style transfer examples by hierarchical-EC²-VAE model.

4.4.2 Latent z_r interpolation

We interpolate rhythm representations z_r of two phrases using SLERP [31] while keeping the pitch and chord unchanged. The interpolated latent representations can then be “re-synthesized” using Hierarchical-EC²-VAE.

As shown in Figure 6, we interpolate z_r of the piece A and B with different SLERP weights. The results exhibit a surprising sense of coherence of pitch and rhythm in generative melodies, even in the transition between consecutive bars. This implies that a longer-term representation is also adept at modeling short-term generation and even contains more global harmonic information than a short-term representation.

4.4.3 Theme variation

We can also achieve theme variation by adding a Gaussian noise to z_r while keeping z_p unchanged. As a sample shown in Figure 7, we find that as the variance of the noise grows larger, the pitch and rhythm of the generative melody are still reasonable smooth, implying that the long-term representations contain the coherence of contextual information and can “control” the generation process.



Figure 6: Interpolation examples.

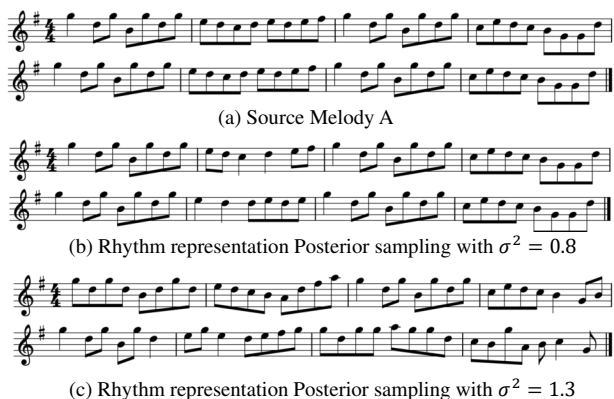


Figure 7: Rhythm representation posterior sampling examples.

4.5 Subjective Evaluation

One may wonder what are the advantages of learning long-term representations since we can always generate the music bar by bar using short-term models and just concatenate the generated samples together. One merit lies in the coherency in controlled music generation. For example, when sampling the long-term rhythm representation, the overall rhythm pattern of a phrase is controlled as an organic whole, while individually sampling the rhythm of different bar may easily lose the rhythm coherency. To better illustrate this idea, we conduct a survey on theme variation (as in Section 4.4.3) to compare the performance of the proposed 8-bar Hierarchical-EC²-VAE and baseline 2-bar EC²-VAE.

4.5.1 Survey Configuration

In our survey, each subject is given 5 groups of pieces. Each group contains three 8 bar pieces: a human-composed piece from Nottingham dataset and 2 theme variations generated by a 2-bar EC²-VAE and Hierarchical-EC²-VAE, respectively. In each group, the generated pieces use z_p of the human-composed piece and the sampled z_r .

Each subject listens to five randomly arranged groups in turn and is required to rate each melody ranging from 1 (very low) to 5 (very high) according to three aspects: *creativity*, *naturalness* (how human-like the composition is) and overall *musicality*.

4.5.2 Results

A total of 29 subjects (18 females and 11 males) participated in the survey. Experimental results depicted in Figure 8 demonstrate that people prefer melodies generated by the proposed Hierarchical-EC²-VAE to those generated by the 2 bar EC²-VAE [7], implying the effects of a long-term coherence learned by our model. The heights of bars represent means of the ratings and the error bars represent the MSEs computed via within-subject ANOVA [32]. The results show that our model performs significantly better than the 2 bar EC²-VAE in terms of all three dimensions ($p < 0.05$). Besides, the qualities of melodies generated by the proposed Hierarchical-EC²-VAE reach a competitive standard compared to the human-composed pieces, especially in *creativity*.

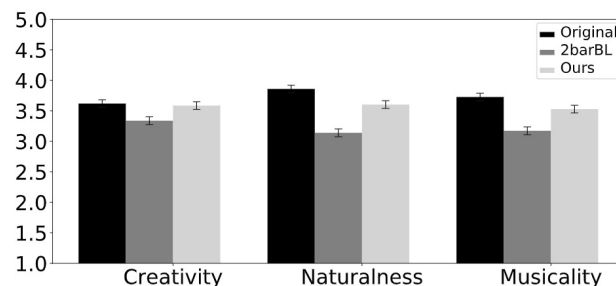


Figure 8: The results of the subjective evaluation.

5. CONCLUSION

In conclusion, we contribute a pipeline of algorithms to learn long-term and disentangled music representations. The main novelty lies in the proposed two inductive biases which constrain the long-term representations using contextual information. The first one requires long-term representation to be not too different from the short-term ones which represent a part of the long-term sequence, and we demonstrate contrastive loss is well-suited for such rough constraint. The second inductive bias is that a good long-term representation should be able to reconstruct the corresponding short-term ones, and we use a hierarchical predictive model to realize this constraint. Unlike most hierarchical models, our purpose is not prediction for its own sake, but rather to leverage the prediction power to learn a well-disentangled long-term representation. Experimental results show that our approach is quite successful, capable of disentangling pitch and rhythmic factors for phrase-level (32 beats) melody without increasing the dimensionality of latent representation compared to bar-level models. Moreover, the learned representations enable high-quality phrase-level style transfer via representation swapping and theme variation by representation interpolation and posterior sampling.

6. REFERENCES

- [1] A. Roberts, J. H. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden*, 2018, pp. 4361–4370.
- [2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, Long Beach, CA, USA*, 2017, pp. 5998–6008.
- [4] K. Chen, G. Xia, and S. Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," in *14th International Conference on Semantic Computing, San Diego, CA, USA*, 2020, pp. 128–135.
- [5] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, Montreal, Canada*, 2020, pp. 662–669.
- [6] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands*, 2019, pp. 343–351.
- [7] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands*, 2019, pp. 596–603.
- [8] Y. Huang and Y. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA*, 2020, pp. 1180–1188.
- [9] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, Paris, France*, 2018, pp. 747–754.
- [10] T. Akama, "Controlling symbolic music generation based on concept learning from domain knowledge," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands*, 2019, pp. 816–823.
- [11] Y. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, Delft, The Netherlands*, 2019, pp. 746–753.
- [12] J. Jiang, G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer VAE: A hierarchical model for structure-aware and interpretable music representation learning," in *International Conference on Acoustics, Speech and Signal Processing, Barcelona, Spain*, 2020, pp. 516–520.
- [13] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 1597–1607.
- [14] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA*, 2020, pp. 9726–9735.
- [15] X. Chen, H. Fan, R. B. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *CoRR*, vol. abs/2003.04297, 2020.
- [16] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. JMLR Proceedings, vol. 9, 2010, pp. 297–304.
- [17] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, United States*, 2013, pp. 2265–2273.
- [18] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *CoRR*, vol. abs/1807.03748, 2018.
- [19] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *7th International Conference on Learning Representations, New Orleans, LA*, 2019.
- [20] F. Lerdahl and R. Jackendoff, "A generative theory of tonal music," *Journal of Aesthetics and Art Criticism*, vol. 9, no. 1, pp. 72–73, 1996.
- [21] W. Rothstein, O. Jonas, and J. Rothgeb, "Introduction to the theory of heinrich schenker: The nature of the musical work of art," *Journal of Music Theory*, vol. 27, no. 2, 1983.

- [22] M. Hamanaka, K. Hirata, and S. Tojo, "Implementing "a generative theory of tonal music"," *Journal of New Music Research*, vol. 35, no. 4, pp. pp. 249–277, 2006.
- [23] A. Marsden, "Schenkerian analysis by computer: A proof of concept," *Journal of New Music Research*, vol. 39, no. 3, pp. 269–289, 2010.
- [24] H. H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, Montreal, Canada, 2020*, pp. 109–116.
- [25] G. Hadjeres and L. Crestel, "Vector quantized contrastive predictive coding for template-based music generation," *CoRR*, vol. abs/2004.10120, 2020.
- [26] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *CoRR*, vol. abs/2005.00341, 2020.
- [27] E. Foxley, "Nottingham database," 2011.
- [28] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, Montreal, Canada, 2020*, pp. 38–45.
- [29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2015.
- [30] H. Kim and A. Mnih, "Disentangling by factorising," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 2654–2663.
- [31] S. Hollasch, "Advanced animation and rendering techniques: By alan watt and mark watt, acm press," *Computers & Graphics*, vol. 18, no. 2, p. 249, 1994.
- [32] H. Scheffé, "The analysis of variance," in *Architectural Institute of Japan*, 1999.

LEARNING PITCH-CLASS REPRESENTATIONS FROM SCORE-AUDIO PAIRS OF CLASSICAL MUSIC

Christof Weiß^{1,2}, Johannes Zeitler¹, Tim Zunner¹, Florian Schuberth¹, Meinard Müller¹

¹ International Audio Laboratories Erlangen, ² LTCI, Télécom Paris, Institut Polytechnique de Paris

{christof.weiss,meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Chroma or pitch-class representations of audio recordings are an essential tool in music information retrieval. Traditional chroma features relying on signal processing are often influenced by timbral properties such as overtones or vibrato and, thus, only roughly correspond to the pitch classes indicated by a score. Deep learning provides a promising possibility to overcome such problems but requires large annotated datasets. Previous approaches therefore use either synthetic audio, MIDI-piano recordings, or chord annotations for training. Since these strategies have different limitations, we propose to learn transcription-like pitch-class representations using pre-synchronized score-audio pairs of classical music. We train several CNNs with musically inspired architectures and evaluate their pitch-class estimates for various instrumentations including orchestra, piano, chamber music, and singing. Moreover, we illustrate the learned features' behavior when used as input to a chord recognition system. In all our experiments, we compare cross-validation with cross-dataset evaluation. Obtaining promising results, our strategy shows how to leverage the power of deep learning for constructing robust but interpretable tonal representations.

1. INTRODUCTION AND RELATED WORK

In the field of music information retrieval (MIR), many algorithms rely on pitch-class or chroma representations for analyzing audio recordings. Such representations capture the signal's energy distribution over the twelve chromatic pitch classes (ignoring octave information) and, thus, allow for a direct musical interpretation. Chroma features have been successfully used for different MIR tasks such as chord recognition [1–4], key estimation [5], structure analysis [6], or audio retrieval [7, 8] especially for Western music. While traditional chroma features were designed in a handcrafted fashion based on signal processing techniques [9–12], such features exhibit several drawbacks caused by audio-related artifacts such as timbral characteristics, overtones, vibrato, or transients. Moreover, the relative loudness of a note directly influences the feature representation.

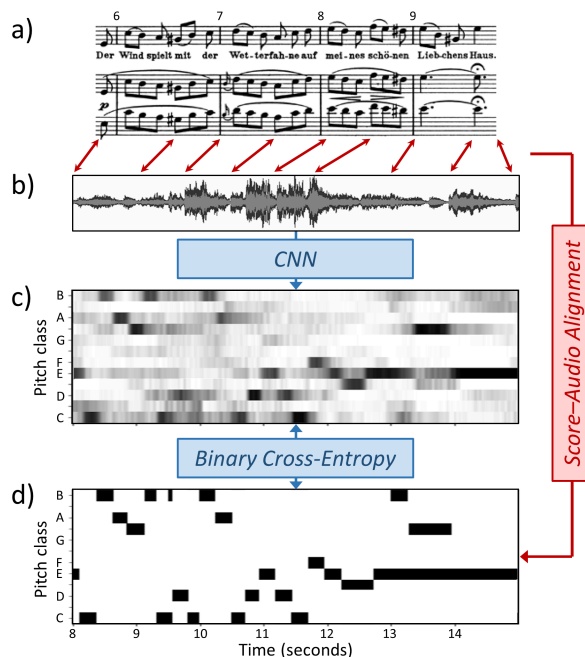


Figure 1. Illustration of the pitch-class training strategy with an example from Schubert’s *Winterreise* [13]. (a) Score. (b) Audio recording. (c) Pitch-class estimates of the CNN. (d) Pitch-class labels derived from aligned score.

Over the years, a number of solutions for these problems were proposed involving spectral whitening [14], peak picking [12], overtone removal [3, 15], or timbre homogenization [16]. Most of these techniques led to improved results for tasks such as chord recognition [1–4] or audio retrieval [7]. However, the problem remains challenging since improvements for one task may deteriorate another—a good chroma for music synchronization [16] might be worse for chord recognition [2], or removal of harmonics might introduce sub-harmonic artifacts [3]. Finally, chroma features are often noisy compared to the pitch classes in the score, limiting their interpretability by musicologists as well as their potential for visualization and cross-modal retrieval and analysis applications.

To overcome such problems, more recent strategies make use of deep neural networks for learning chroma representations from data [17–21]. For the successful training of high-capacity networks, large amounts of annotated recordings are necessary. Since manual creation of pitch-class annotations is tedious and requires expert knowledge, there are several alternative strategies, all of which have their benefits and limitations. Early approaches to a “deep chroma” make use of chord labels to derive pitch-



Table 1. Datasets and annotations used in this work (“Perf.”: number of performances per piece).

ID	Dataset Name	Instrumentation	Pitch annotation type	Chords	Tracks	Pieces	Perf. ¹	hh:mm
SWD	Schubert Winterreise [13]	Piano, voice	Aligned scores	yes	216	24	9	10:50
BSD	Beethoven Piano Sonatas [25] ²	Piano	Aligned scores	yes	192	32	6	62:30
WaR	Wagner Ring [26] ²	Orchestra, voice	Aligned scores	no	33	11	3	43:13
MuN	MusicNet [27]	Piano, strings, winds	Aligned scores	no	330	330	1	34:08
SMD	Saarland Music Data [22]	Piano	MIDI piano / Disklavier	no	50	50	1	4:43

class annotations [17, 18]. As shown in [17], this leads to a chroma extractor that has a strong bias towards the chords’ pitch classes (in [17], these are triads) and does not actually detect the pitch classes notated in the score, thus limiting interpretability, generalization to other chord vocabularies and genres, and applicability to other tasks. As an alternative strategy for obtaining training data, symbolic music representations were used to render synthetic audio recordings together with the corresponding annotations [19]. While this is pragmatic, systems trained on synthetic data often show limited generalization to recorded audio. Another strategy makes use of MIDI-fied instruments (e. g., Disklaviers) for capturing pitch information, which led to a number of comprehensive piano transcription datasets [22–24]. However, these approaches are limited to piano or MIDI-fied instruments, and the use of the sustain pedal constitutes a problem for determining the perceptually relevant duration of a note.

In this paper, we target a pitch-class representation that relates to the task of multi-pitch estimation (MPE) or framewise transcription [28]. Concretely spoken, we aim for *detecting the framewise activity of all pitch classes indicated by the score* (multi-pitch-class estimation, see Figure 1c). In an ideal scenario, such a representation helps to close the gap between audio- and symbolic-based MIR and, as a consequence, is well-interpretable and capable of generalizing to different music genres, instrumentations, and MIR tasks. For this purpose, we propose an alternative training strategy using score–audio pairs of classical music that are pre-aligned using music synchronization techniques [29]. As an alternative to this, weakly-annotated score–audio pairs were recently used for training using an attention mechanism [30], the CTC loss [20], or a multi-label CTC variant that can deal with polyphonic pitch-class representations [21]. A (strong) alignment strategy similar to ours was successfully applied for multi-instrument music transcription with the MusicNet dataset [27], which we include in this paper. We prepare three further classical music datasets comprising several performances of Schubert’s song cycle *Winterreise* [13], the first movements of Beethoven’s piano sonatas [25], and Wagner’s four-opera cycle *Der Ring des Nibelungen* [26]. We generate pitch-class annotations for these datasets using symbolic scores, manual measure annotations [26], and music synchronization techniques [29]. The data comprises various styles and instrumentations including piano, orchestra, chamber music, as well as singing voice.

As our first contribution, we use this data for supervised learning of a transcription-like pitch-class representation with a medium-sized, musically motivated convolutional

neural network (CNN) inspired by [20, 31, 32]. We test the network’s pitch-class estimates using evaluation measures from music transcription. Second, we compare this CNN with other architectures such as wider and deeper networks, inception blocks, and residual connections. Third, we test the benefit of the learned features for harmony analysis, specifically chord recognition for classical music. To systematically assess the role of the input features, we employ a controlled and well-understood chord recognition approach based on hidden Markov models (HMMs) [2, 4]. We compare the novel features with traditional chroma features and idealized pitch-class representations derived from the score. In all stages, we compare cross-validation results on individual datasets with cross-dataset results to systematically test generalization [33].

The remainder of paper is organized as follows. In Section 2, we introduce the datasets used for our experiments. Section 3 describes our CNN-based feature learning. In Section 4, we evaluate the learned pitch-class features. Section 5 discusses chord recognition results using the learned features. Section 6 concludes the paper.

2. DATASETS

As mentioned in Section 1, the limited availability of annotated data is a major issue for multi-pitch and pitch-class estimation—a “key challenge” of music transcription [34]. Since manual annotation is tedious and requires expert annotators, several workarounds were proposed [35]. A common approach involves the use of MIDI-fied pianos (Disklaviers) for simultaneously generating audio and annotations, leading to piano transcription datasets such as SMD [22], MAPS [23], or MAESTRO [24].

Beyond the solo piano scenario, there are only few and small datasets with pitch annotations such as Bach10 [36], TRIOS [37], or PHENICX-Anechoic [38] (all ≤ 10 pieces), which often involve multi-track recordings to simplify the manual annotation process [36–38] or to automatically generate annotations using a monophonic F0-tracker as done for MedleyDB [39]. Since this leads to F0 annotations following the performed frequencies rather than the pitches in the score, we do not use MedleyDB here.

As a further strategy, score–audio pairs of classical music can be exploited to generate pitch (class) annotations. This requires score–audio synchronization methods [29]. A dataset created with this strategy is MusicNet (MuN) [27], which comprises pitch annotations for 330 audio recordings of piano and chamber music. For our experiments, we reduce the pitch annotations to the pitch-class level. As another score–audio dataset, we make use of

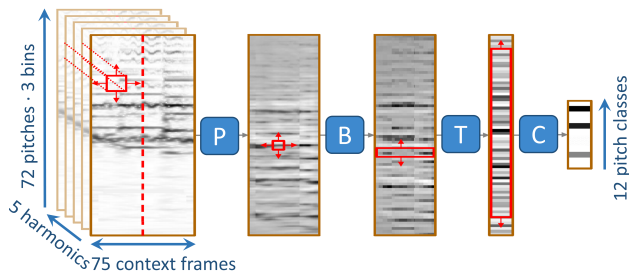


Figure 2. Illustration of the CNN architecture (Basic).

the Schubert Winterreise Dataset (SWD) [13], which comprises recorded performances (two of nine freely available), scores, measure positions, and chord annotations. We use the scores (MIDI files) and measure annotations together with a synchronization algorithm [29] based on dynamic time warping (DTW) to generate pitch-class annotations.¹ Using the same strategy, we create two further private² datasets: The Beethoven Sonatas Dataset (BSD) comprises the 32 first movements of Beethoven’s piano sonatas in six versions. Using DTW-based alignment [29], we generate pitch-class annotations from corresponding scores and chord labels based on the annotations by Chen and Su [25]. In a similar fashion, we create pitch-class annotations for Wagner’s four-opera cycle *Der Ring des Nibelungen* (WaR) based on manual measure annotations [26] and a full score (first act of *Die Walküre*) or a piano-reduced score (remaining acts), respectively. Table 1 gives an overview of the datasets used in this paper.

3. DEEP-LEARNING METHODS

In this section, we describe our CNN-based approach for extracting pitch-class representations and discuss our design choices, motivated by related work. Previous deep-learning approaches for pitch-class representations use a variety of architectures including fully-connected [17, 40] and convolutional neural networks (CNNs) [17, 19, 20], where the latter often exhibit large kernels in the last layers to aggregate harmonic information. Due to the lower number of parameters, we pursue a CNN-based approach inspired by [20, 32], summarized in Figure 2 and Table 2.

Input representation. As network input, spectral representations are used most frequently, either generated by a short-time Fourier transform [17] or a constant-Q transform (CQT) [40]. The CQT can be extended to a harmonic CQT (HCQT) with CQTs in harmonic frequency ratios stacked on top of each other, thus allowing for convolutions across harmonics (overtones) along the channel axis [32]. As our input representation, we use such a HCQT with five harmonics (no sub-harmonic). Based on audio sampled at 22050 Hz, we use a CQT hopsize of 384 samples resulting in a feature rate of roughly 57.4 Hz.³ Our HCQT spans 72 semitones (6 octaves) starting at C1 and a resolution of three bins per semitone. We choose a

¹ With this paper, we publish pitch and pitch-class annotations for the SWD, to be found at <https://zenodo.org/record/5139893/>.

² These datasets cannot be published due to copyright issues.

³ As the only parameter, the CQT is determined by the hopsize, which must be an integer multiple of powers of two.

Table 2. Musically informed CNN architecture (Basic).

Function	Kernel size, #	Stride	Output Shape	Activ.
Prefiltering (P):				
LayerNorm			$216 \times 75 \times 5$	
Conv2D	$15 \times 15, N_0$	(1, 1)	$216 \times 75 \times N_0$	LReLU
MaxPool		(1, 2)	$216 \times 37 \times N_0$	
Dropout				
Binning to MIDI pitches (B):				
Conv2D	$3 \times 3, N_1$	(3, 3)	$72 \times 12 \times N_1$	LReLU
MaxPool		(1, 2)	$72 \times 6 \times N_1$	
Dropout				
Time reduction (T):				
Conv2D	$1 \times 6, N_2$	(1, 1)	$72 \times 1 \times N_2$	LReLU
Dropout				
Chroma reduction (C):				
Conv2D	$1 \times 1, N_3$	(1, 1)	$72 \times 1 \times N_3$	LReLU
Dropout				
Conv2D	$61 \times 1, 1$	(1, 1)	$12 \times 1 \times 1$	Sigmoid

centering strategy with bins corresponding to integer MIDI pitches placed between the two surrounding bins.

Context frames. To accurately predict a frame, a network needs information about the context surrounding the target frame. When using a single-stage system, this can be done by feeding multiple time frames of the spectral representation to the network [17, 19, 32]. For our network, we feed the network with 75 context frames (37 to each side of the target frame), corresponding to 1.3 sec at a frame rate of 57.4 Hz. Thus, we feed the network with an input tensor of shape $216 \times 75 \times 5$ to predict a pitch-class activation vector of size 12 (see Table 2).

Basic CNN architecture. Our proposed CNN filters the input data in a musically meaningful way. Table 2 gives detailed information about the proposed model; Figure 2 provides a schematic illustration. First, we perform layer normalization to ensure zero mean and unit variance for each input sample followed by a (trainable) linear transformation of the normalized input tensor. Next, N_0 (default 20) feature maps are extracted in the **Prefiltering** layer (P). Using a kernel size of 15×15 allows the network to detect, e.g., vibrato for singing. The second convolutional layer performs a **Binning to MIDI pitches** (B) by moving a 3×3 kernel with stride 3 and no padding along the pitch axis, so that each output bin corresponds to an integer MIDI pitch. We learn N_1 (default 20) feature maps. Third, a convolution across time performs a **Time reduction** (T), resulting in N_2 (default 10) feature maps with 72 bins each. Fourth, we perform pitch-class or **Chroma reduction** (C): After reducing the representation to N_3 (default 1) channels with a 1×1 convolution, we move a kernel with length $72 - 11 = 61$ along the pitch axis. In all convolutional layers, we use LeakyReLU activation (negative slope 0.3) to prevent vanishing gradients. MaxPooling along time reduces the number of parameters and forces generalization. Dropout (rate 0.2) hampers overfitting while retaining a large amount of information. We use sigmoid activation in the final layer and train with binary cross-entropy loss between predicted pitch-class vectors $\mathbf{p} \in [0, 1]^{12}$ and bi-

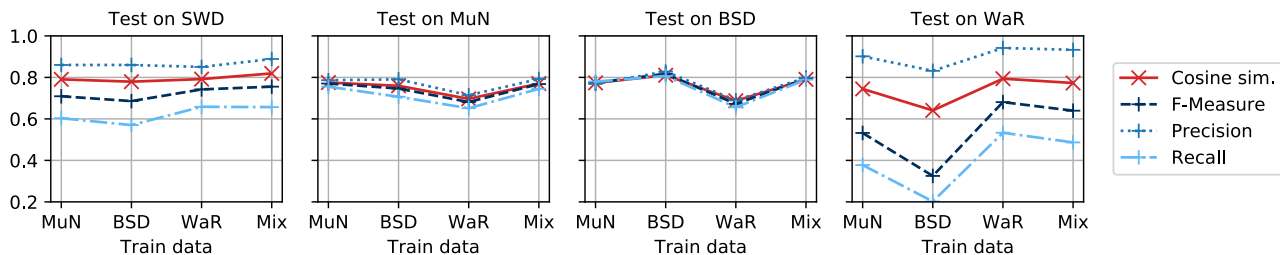


Figure 3. Pitch-class estimation results (*Basic* model) for different datasets (train/test subsets of each dataset are disjoint).

nary, multi-hot target vectors $\mathbf{t} \in \{0, 1\}^{12}$, obtained from the score’s note occurrences without weighting (Figure 1).

Larger CNN architectures. In addition to this basic CNN architecture (denoted as *Basic* in the following) with roughly 27k convolutional parameters (plus the parameters of layer normalization), we test a number of extended network architectures. A simple strategy is to increase the number of learned feature maps ($N_0 \dots N_3$). For the architecture *BasicLast10*, we increase the last layer to $N_3 = 10$ (28k conv. params.). For the architecture *Wide*, we increase the number of channels in all layers by a factor of five so that $N_0 = N_1 = 100$, $N_2 = 50$, $N_3 = 5$ (233k conv. params.). Since the choice of the prefiltering kernel size is difficult, we adopt the concept of inception blocks [41], where the input is filtered by kernels with different sizes in parallel. For this *WideInception* architecture, we use kernel sizes 3×3 , 9×9 , 15×15 , and 27×27 , leaving the total number of kernels as in *Wide*. As an alternative, we test a *Deep* architecture with more hidden layers, replicating the first layer (P) five times. All remaining layers and parameters are identical to *Basic*. Since training deep architectures is difficult due to vanishing gradients, we test the *DeepResNet* architecture with residual connections [42]. We add shortcut connections to the five P layers, leaving the remainder identical to *Deep*.⁴

4. EVALUATING PITCH-CLASS ESTIMATION

In the following, we evaluate the pitch-class estimates of different networks, trained and tested on various datasets. We measure the frame-wise precision, recall, and F-measure (F) using a threshold of 0.5 (motivated by the sigmoid activation) as well as the cosine similarity (CS) between targets and non-thresholded predictions.

Evaluating general settings. We start with several experiments in a cross-validation on SWD (train on seven, test on two performances i. e., a version split [33]), which serves as our development set to decide on general settings. We train all networks with Adam [43] on mini-batches of size 25 using learning rate scheduling and early stopping. For the *Basic* architecture (as described in Section 3), we obtain $F=0.832$ and $CS=0.836$ on the test versions of SWD, which is already a promising result. Precision (0.850) is slightly higher than recall (0.814). Since the choice of the input HCQT’s frame rate is important, we compare this result (with a frame rate of 57.4 Hz) to the use of a smaller

rate (10.1 Hz) while holding the (physical) amount of context constant by adjusting CNN kernel shapes accordingly. With this smaller frame rate, we obtain slightly worse results of $F=0.820$ and $CS=0.833$. Thus, we use the finer resolution of 57.4 Hz in the following. Next, we test the influence of context frames: Reducing the original context of 75 frames (roughly 1.3 sec) to 51 frames (0.9 sec) leads to decreased results of $F=0.827$, with 25 frames to a further decrease of $F=0.823$. We thus opt for the larger context of 75 frames. Finally, we test different kernel sizes in the first layer (prefiltering P). Compared to *Basic* with 15×15 kernels, 9×9 kernels lead to $F=0.835$, and 5×5 kernels to $F=0.824$. Though the 9×9 kernels perform slightly better than the 15×15 kernels, we choose the larger kernel size since it spans a larger pitch range and may be better capable of, e. g., detecting vibrato.

Evaluating different datasets. With these parameter choices, we now perform a cross-dataset experiment. In addition to the SWD dataset (two versions for test), we use MuN (50 pieces test, 280 pieces train/val), BSD (two versions test, four versions train/val), and WaR (test on *Die Walküre*, 1st act, train/val on all other acts). We further compile a *Mix* train set, which encompasses the train subsets from MuN, BSD, and WaR to equal parts (using subsampling). Note that train subsets of a dataset are never used for testing (and vice versa) even for cross-dataset splits. Figure 3 shows the pitch-class estimation results for all train/test combinations. Using the same source for train and test set (MuN–MuN, BSD–BSD, WaR–WaR), the respective combination yields best results. In those cases, the *Mix* train set always achieves second-best results. In the case of a completely unknown test set (SWD), the diverse train data in *Mix* yields best results, slightly worse than the cross-validation results in the previous paragraph.

Evaluating CNN architectures. We now compare the *Basic* model to the larger architectures introduced in Section 3. Figure 4 illustrates the respective results using the same test datasets as for the previous experiment and the *Mix* training set. Compared to the *Basic* architecture, *BasicLast10* has an increased number of channels in the final layer ($N_3=10$), which yields slightly better results than *Basic* with only about 600 more parameters. In comparison, the *Wide*, *WideInception*, *Deep*, and *DeepResNet* architectures increase the number of parameters by a factor of roughly ten. All of them yield better results than *Basic*, except for the WaR test set. Although we can achieve minor improvements by the use of inception blocks and skip connections, the performance metrics

⁴ Our source code (Keras) and pre-trained models are available under https://github.com/christofw/pitchclass_cnn/.

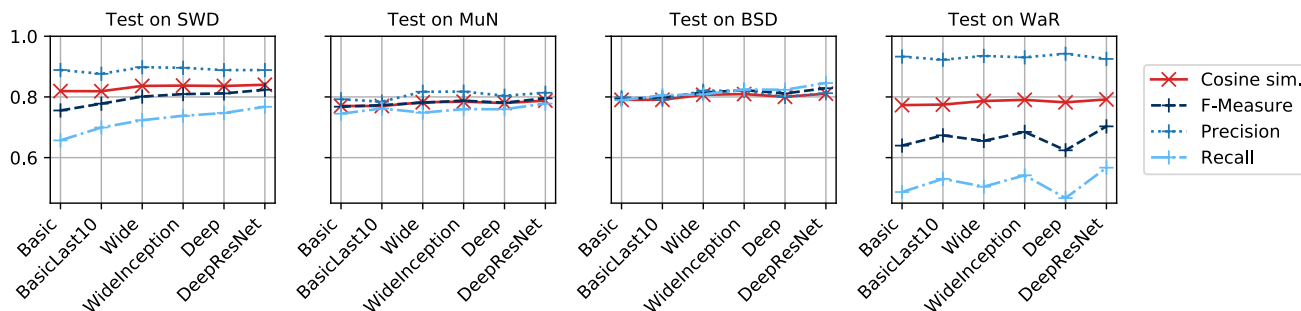


Figure 4. Pitch-class estimation results for different architectures trained on *Mix* dataset (subsets of BSD, MuN, WaR).

of the four most complex architectures are quite similar. Comparing these networks’ results on SWD (cross-dataset) with our first experiment—a cross-validation on SWD with $F=0.832$ and $CS=0.836$ —, we notice almost identical results. From this, we draw the important conclusion that a larger, diverse training set (e.g., *Mix*) together with a high-capacity network (e.g., *Wide*) can compensate for not “knowing” the particular dataset (here the style of *Winterreise* and the combination of piano and singing). We therefore use the *Wide* network trained on mixed datasets for the following chord recognition experiment.⁵

5. APPLICATION FOR CHORD RECOGNITION

Besides visualization purposes, pitch-class representations serve as front-end features for various MIR applications. To examine the effectiveness of our learned features for the important task of chord recognition, we present systematic experiments using the chord annotations of SWD and BSD. Rather than optimizing the chord recognition performance, we want to analyze the features’ influence and test the hypothesis that our learned features behave similar to features derived from the score (the training targets for our CNNs). To gain these insights, we do not use an end-to-end chord-recognition approach but opt for a traditional yet effective method based on HMMs and Gaussian chord models [4]. For train/test of the HMM, we again compare cross-dataset results with cross-validation results on each dataset, making sure that neither a specific song nor a specific performance are seen during training (neither split) to avoid the kind of “musical overfitting” observed in [33].

Chord recognition method. On the training set, we learn multivariate Gaussian chord models in the pitch-class space \mathbb{R}^{12} . We cyclically shift and average the models of each chord type in order to obtain transposition-invariant models, which we use for generating the HMM’s emission probabilities. Inspired by [4], we apply a uniform transition matrix with a high self-transition probability, which we optimize on the validation set together with other hyperparameters (log compression strength and pre-filtering length for the input features). We simplify the chord annotations of SWD and BSD to three common chord vocabu-

larities: *MajMin* comprises the 24 major and minor triads, *Triads* adds the 12 diminished and 4 augmented triads resulting in 40 chords, and *Sevenths* further adds five types of seventh chords (dom7, maj7, min7, half-dim7, dim7) amounting to 91 chords.⁶

Evaluating feature variants. First, we assess the effectiveness of our pitch-class features (denoted as P_{CNN}) and compare those with other feature variants. To obtain P_{CNN} , we train the *Wide* model in a cross-dataset split (train data similar to *Mix* but leaving out the target dataset): For SWD, we train on BSD, MuN, and WaR; for BSD, we train on SMD, MuN, and WaR (we replace BSD with SMD to include a piano dataset). The resulting features are re-sampled to 10Hz. For comparison, we consider three traditional chroma variants based on a CQT (P_{CQT}), an STFT (P_{STFT}), and an IIR filterbank (P_{IIR}), respectively.⁷ As baseline, we use an idealized binary feature (P_{Score}) derived from the aligned score (the CNN’s training targets). We train and validate the HMM in a cross-validation setting, making sure that neither test performances nor pieces are seen during training [33].

Figure 5 shows the results, reporting chord-symbol recall (which equals the F-measure and accuracy when ignoring no-chord frames). Looking at the traditional feature variants (P_{CQT} , P_{STFT} , P_{IIR}), we observe varying performance, with best results for P_{CQT} on SWD and for P_{STFT} on BSD. Over all datasets, our CNN-based feature P_{CNN} systematically outperforms the traditional variants with substantial improvements for the more complex vocabularies *Triads* and *Sevenths*. Most remarkably, P_{CNN} almost reaches the performance of the idealized chroma P_{Score} . This is a promising result, indicating that for the task of chord recognition, the *signal-processing* challenge of extracting pitch-class information from audio recordings can be approached in a suitable way using deep learning, while the remaining challenge mainly lies in the mapping of pitch-class information to chord labels.

Evaluating train/test splits. Next, we test the generalization behavior of the chord recognition system (Figure 6). To this end, we compare the cross-validation of the previous experiment with cross-dataset evaluation where we train and test on the other dataset, respectively (note that we speak of training chord recognition—the fea-

⁵ For training large networks, a sufficient amount of data is necessary to prevent overfitting. Our largest model has roughly 550k parameters (including layer normalization). As a comparison, 550k frames of training data sampled at 50 Hz give a dataset of about three hours. The amount of data in MuN (34 hours), for example, is large compared to the number of parameters, which is even more the case for the larger *Mix* dataset.

⁶ We do not discriminate between chords that are identical on the pitch-class level, e.g., C aug and E aug or C dim7 and Eb dim7.

⁷ For implementation details, please see <https://librosa.org/>.

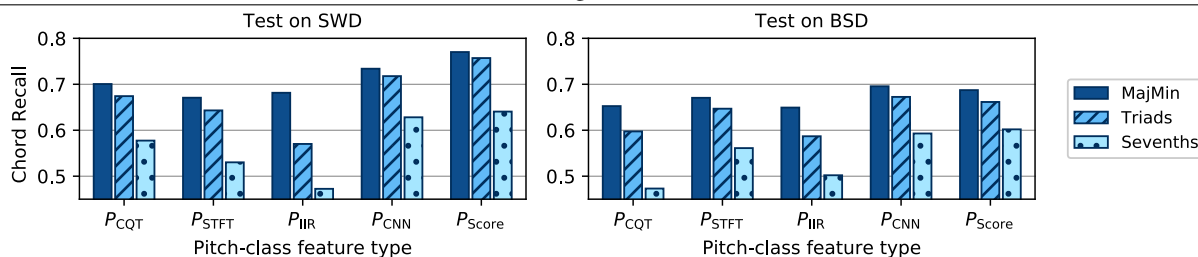


Figure 5. Chord recognition results based on different pitch-class features for SWD (left) and BSD (right), trained/tested with cross-validation using different chord vocabularies.

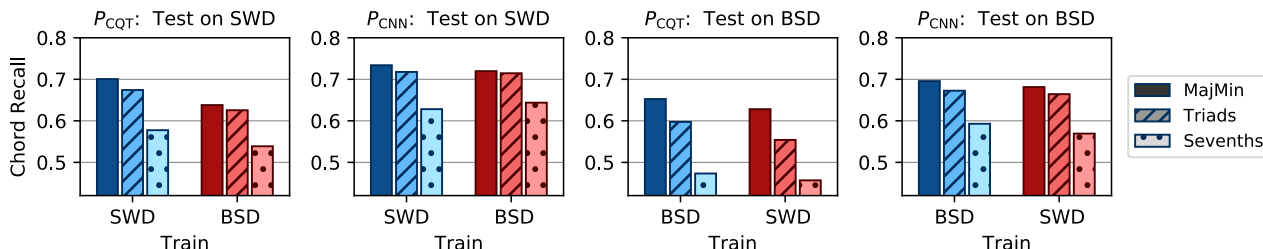


Figure 6. Chord recognition results using features P_{CQT} and P_{CNN} for SWD (left) and BSD (right), trained/tested with cross-validation (blue) and cross-dataset evaluation (red).

tures P_{CNN} are always trained in a cross-dataset split). Using the traditional feature P_{CQT} (left), we observe a clearly worse performance for the cross-dataset experiment (red), which is musically more challenging. When using P_{CNN} (second plot), this drop does not occur—we observe almost identical results for cross-validation and cross-dataset. Testing on BSD (right plots), this tendency is weaker. Still, we conclude that our score-based feature P_{CNN} not only leads to better but also to more robust chord recognition systems, which are widely capable of generalizing to unseen music.

Comparing score- and chord-based pitch classes.

Overall, our chord recognition results are not as high compared to, e. g., recent results for pop music [17–19]—even for our baseline feature P_{Score} . On the one hand, this might be due to the simpler system (HMM) we use compared to recent approaches. As a main difference, however, the methods of [17, 18] directly use the chord labels to train a *chord-related* pitch-class representation (instead of a *score-oriented* one). To test the potential of this strategy, we use another baseline feature (P_{Chord}) derived from the chord annotations (without any reduction to a smaller vocabulary), thus capturing idealized, binary activities of the chords’ pitch classes. As Figure 7 indicates, we observe a large increase for both datasets. This is of course expected (confirming a similar baseline experiment for pop music in [17]). The comparison of P_{Chord} with P_{Score} and P_{CNN} tells us that the main challenge of chord recognition (at least for our datasets) is a *musical* one: Even when knowing the pitches from the score, it is difficult to decide on which pitches are relevant for the annotated chords. This is of course not trivial and touches questions of musical style, music theory concepts, and annotator subjectivity [33, 44, 45]. Therefore, deep-learning approaches for mapping score information to chord labels such as [17, 18] are promising. We think that such methods could benefit from using our score-based features as input, thus helping to improve generalization. Beyond that, we want to again

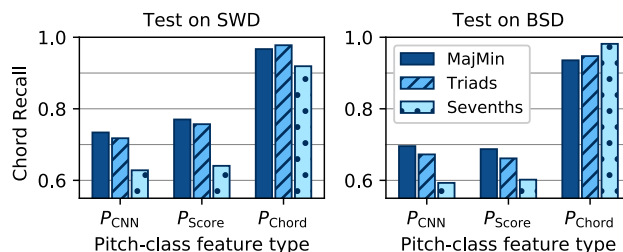


Figure 7. Chord recognition based on P_{CNN} compared with score (P_{Score}) and chord-label (P_{Chord}) baselines.

emphasize that our aim is not to improve chord recognition itself but to obtain a pitch-class representation that helps to close the gap between audio- and symbolic-based approaches to, e. g., harmony analysis and generalizes to unseen recordings. As our experiments indicate, deep learning allows us to take a crucial step towards this goal.

6. CONCLUSIONS

We presented a CNN-based approach for extracting transcription-like pitch-class representations from music audio recordings. As our main contribution, we proposed a novel strategy for training CNNs with pre-aligned score–audio pairs of classical music. We tested the effectiveness of this approach for pitch-class estimation by comparing different CNN architectures and dataset splits. Using the features as input to a traditional chord recognition system led to improved results and generalization compared to traditional features and is almost on par with symbolic pitch-class features. We conclude that the signal processing challenge of extracting pitch-class information from audio recordings can be successfully approached with deep learning, thus serving as an excellent basis to approach the musical challenge of finding the relevant pitch classes for chords and other harmonic structures—an interesting observation that should be verified for genres beyond Western classical music in future work.

7. ACKNOWLEDGEMENTS

This work is supported by the German Research Foundation (DFG WE 6611/1-1, DFG MU 2686/7-2). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institute for Integrated Circuits IIS. We thank Leo Brütting and Michael Kohl for help with dataset preparation and Geoffroy Peeters for fruitful discussions.

8. REFERENCES

- [1] M. Stein, B. M. Schubert, M. Gruhne, G. Gatzsche, and M. Mehnert, "Evaluation and comparison of audio chroma feature extraction methods," in *Proceedings of the AES Convention*, Ilmenau, Germany, 2009.
- [2] N. Jiang, P. Grosche, V. Konz, and M. Müller, "Analyzing chroma feature types for automated chord recognition," in *Proceedings of the AES Conference on Semantic Audio*, Ilmenau, Germany, 2011.
- [3] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 135–140.
- [4] T. Cho and J. P. Bello, "On the relative importance of individual components of chord recognition systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 2, pp. 477–492, 2014.
- [5] E. Gómez and P. Herrera, "Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Barcelona, Spain, 2004, pp. 92–95.
- [6] R. B. Dannenberg and M. Goto, "Music structure analysis from acoustic signals," in *Handbook of Signal Processing in Acoustics*, D. Havelock, S. Kuwano, and M. Vorländer, Eds. New York, NY, USA: Springer, 2008, vol. 1, pp. 305–331.
- [7] M. Müller and S. Ewert, "Towards timbre-invariant audio features for harmony-based music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 649–662, 2010.
- [8] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, pp. 1138–1151, 2008.
- [9] T. Fujishima, "Realtime chord recognition of musical sound: A system using common lisp music," in *Proceedings of the International Computer Music Conference (ICMC)*, Beijing, China, 1999, pp. 464–467.
- [10] M. A. Bartsch and G. H. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, USA, 2001, pp. 15–18.
- [11] M. Müller, F. Kurth, and M. Clausen, "Chroma-based statistical audio features for audio matching," in *Proceedings of the IEEE Workshop on Applications of Signal Processing (WASPAA)*, New Paltz, USA, 2005, pp. 275–278.
- [12] E. Gómez, "Tonal description of music audio signals," PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.
- [13] C. Weiß, F. Zalkow, V. Arifi-Müller, H. Grohgan, H. V. Koops, A. Volk, and M. Müller, "Schubert Winterreise dataset: A multimodal scenario for music analysis," *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 14, no. 2, pp. 25:1–18, 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5139893>
- [14] A. P. Klapuri, "Multipitch analysis of polyphonic music and speech signals using an auditory model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 255–266, 2008.
- [15] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *Proceedings of the International Computer Music Conference (ICMC)*, New Orleans, USA, 2006, pp. 306–311.
- [16] M. Müller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.
- [17] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016, pp. 37–43.
- [18] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 188–194.
- [19] Y. Wu and W. Li, "Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model," *IEEE/ACM Transactions on Audio, Speech & Language Processing*, vol. 27, no. 2, pp. 355–366, 2019.
- [20] F. Zalkow and M. Müller, "Using weakly aligned score-audio pairs to train deep chroma models for cross-modal music retrieval," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020, pp. 184–191.
- [21] C. Weiß and G. Peeters, "Training deep pitch-class representations with a multi-label CTC loss," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2021.
- [22] M. Müller, V. Konz, W. Bogler, and V. Arifi-Müller, "Saarland music data (SMD)," in *Demos and Late Breaking News of the International Society for Music Information Retrieval Conference (ISMIR)*, Miami, USA, 2011.

- [23] V. Emiya, R. Badeau, and B. David, “Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.
- [24] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, USA, 2019.
- [25] T. Chen and L. Su, “Functional harmony recognition of symbolic music data with multi-task recurrent neural networks,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 90–97.
- [26] C. Weiß, V. Arifi-Müller, T. Prätzlich, R. Kleinertz, and M. Müller, “Analyzing measure annotations for Western classical music recordings,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, 2016, pp. 517–523.
- [27] J. Thickstun, Z. Harchaoui, and S. M. Kakade, “Learning features of music from scratch,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.
- [28] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, “Automatic music transcription: An overview,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [29] S. Ewert, M. Müller, and P. Grosche, “High resolution audio synchronization using chroma onset features,” in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.
- [30] R. Nishikimi, E. Nakamura, M. Goto, and K. Yoshii, “End-to-end melody note transcription based on a beat-synchronous attention mechanism,” in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, USA, 2019, pp. 26–30.
- [31] A. Elowsson and A. Friberg, “Modeling music modality with a key-class invariant pitch chroma CNN,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 541–548.
- [32] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, “Deep salience representations for F0 tracking in polyphonic music,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 63–70.
- [33] C. Weiß, H. Schreiber, and M. Müller, “Local key estimation in music recordings: A case study across songs, versions, and annotators,” *IEEE/ACM Transactions on Audio, Speech & Language Processing*, vol. 28, pp. 2919–2932, 2020.
- [34] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, “Automatic music transcription: An overview,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [35] L. Su and Y. Yang, “Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription,” in *Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, Plymouth, UK, 2015, pp. 309–321.
- [36] Z. Duan, B. Pardo, and C. Zhang, “Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions,” *IEEE Transactions on Audio, Speech & Language Processing*, vol. 18, no. 8, pp. 2121–2133, 2010.
- [37] J. Fritsch and M. D. Plumbley, “Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis,” in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 888–891.
- [38] M. Miron, J. Carabias-Orti, J. Bosch, E. Gómez, and J. Janer, “Score-informed source separation for multichannel orchestral recordings,” *Journal of Electrical and Computer Engineering*, 2016.
- [39] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, “MedleyDB: A multitrack dataset for annotation-intensive MIR research,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 155–160.
- [40] Y. Wu and W. Li, “Music chord recognition based on MIDI-trained deep feature and BLSTM-CRF hybrid decoding,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 376–380.
- [41] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, 2015.
- [42] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [43] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the International Conference for Learning Representations (ICLR)*, San Diego, USA, 2015.
- [44] Y. Ni, M. McVicar, R. Santos-Rodríguez, and T. D. Bie, “Understanding effects of subjectivity in measuring chord estimation accuracy,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 12, pp. 2607–2615, 2013.
- [45] H. V. Koops, W. B. de Haas, J. A. Burgoyne, J. Bransen, A. Kent-Muller, and A. Volk, “Annotator subjectivity in harmony annotations of popular music,” *Journal of New Music Research*, vol. 48, no. 3, pp. 232–252, 2019.

TRAINING DEEP PITCH-CLASS REPRESENTATIONS WITH A MULTI-LABEL CTC LOSS

Christof Weiß, Geoffroy Peeters

LTCI, Télécom Paris, Institut Polytechnique de Paris, France

ABSTRACT

Despite the success of end-to-end approaches, chroma (or pitch-class) features remain a useful mid-level representation of music audio recordings due to their direct interpretability. Since traditional chroma variants obtained with signal processing suffer from timbral artifacts such as overtones or vibrato, they do not directly reflect the pitch classes notated in the score. For this reason, training a chroma representation using deep learning (“deep chroma”) has become an interesting strategy. Existing approaches involve the use of supervised learning with strongly aligned labels for which, however, only few datasets are available. Recently, the Connectionist Temporal Classification (CTC) loss, initially proposed for speech, has been adopted to learn monophonic (single-label) pitch-class features using weakly aligned labels based on corresponding score–audio segment pairs. To exploit this strategy for the polyphonic case, we propose the use of a multi-label variant of this CTC loss, the MCTC, and formalize this loss for the pitch-class scenario. Our experiments demonstrate that the weakly aligned approach achieves almost equivalent pitch-class estimates than training with strongly aligned annotations. We then study the sensitivity of our approach to segment duration and mismatch. Finally, we compare the learned features with other pitch-class representations and demonstrate their use for chord and local key recognition on classical music datasets.

1. INTRODUCTION AND RELATED WORK

The Pitch Class Profile (PCP) or chroma is one of the most frequently used audio feature in Music Information Retrieval (MIR). Chroma features are typical for MIR for several reasons: First, they were developed specifically for music [1] as opposed to other features which were inherited from speech processing (such as MFCCs). Second, despite the success of end-to-end-systems, PCP or chroma features are still used today due to their semantic mid-level nature, being musically interpretable as energy distribution over the twelve chromatic pitch classes in an audio signal (see Figure 1). Because of this, PCPs directly relate to musical harmony, therefore being used for chord and key

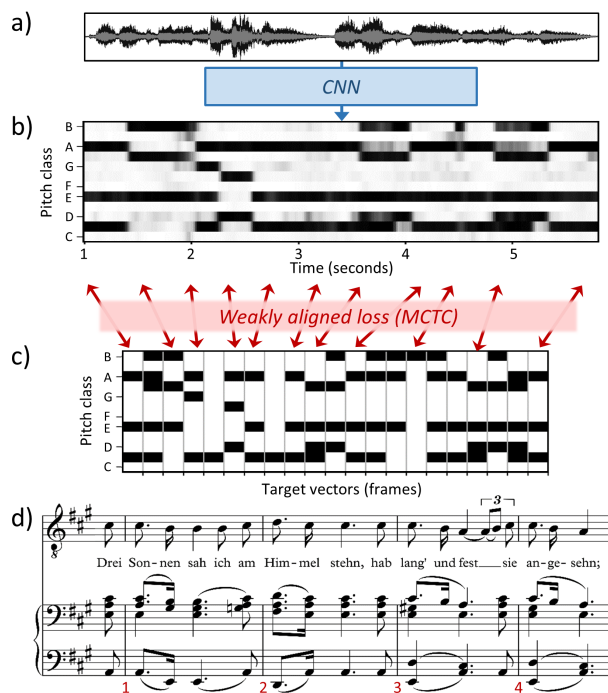


Figure 1. Training a CNN with weakly aligned targets (schematic). Song No.23 from Schubert’s *Winterreise* sung by R. Trekel. (a) Waveform. (b) Pitch-class estimates. (c) Non-aligned Targets derived from the score. (d) Score.

estimation or audio matching (cover song retrieval) tasks.

Chroma features based on signal processing. Early approaches [2, 3] to chroma are based on signal processing and map a time–frequency representation such as the Short Time Fourier Transform (STFT) [2] or the Constant-Q-Transform (CQT) [3] to the twelve pitch classes. However, due to timbral characteristics such as overtones (which correspond to different pitch classes), transient note onsets, or vibrato, these chroma features do not directly reflect the pitch classes notated in the score, thus limiting their interpretability. This motivated sophisticated while still hand-crafted features, which aim at reducing the influence of timbre [4–6], at making this influence equal for all instruments [7], or at equalizing loudness variation and transient components [8]. To study the effect of these improvements for chord recognition, Cho et al. [9] present an in-depth comparison, concluding that suitable chroma features largely redeem the benefits of complex chord models.

Pitch-class representations based on deep learning. Recently, deep learning of pitch-class features from data has become a promising direction. Yet, prior works on “deep chroma” have a concrete application task in mind



and do not directly evaluate the obtained pitch-class representations: Humphrey et al. [10] trained a Convolutional Neural Network (CNN) to estimate a Tonnetz representation and use this representation to estimate chords. Later works [11, 12] extend this to end-to-end (CQT-based) chord recognition. In a similar fashion, more recent approaches [13, 14] train pitch-class extractors using annotations derived from chord labels. While this led to promising results for chord recognition and other tasks [15], Korzeniowski et al. [13] showed that the learned representations strongly focus on chord-like structures and do not actually represent the pitch classes notated in the score, thus limiting their interpretability and generalization capability. As an alternative, Wu et al. [16] estimate pitch classes by training with audio and annotations synthesized from MIDI files (which are available in large quantity). The trained features are used for chord recognition with good results, improved in a follow-up work [17]. While this strategy is interesting, systems trained on synthetic data show limited generalization to real audio.

Training with aligned scores. To overcome this problem, large amounts of real audio recordings with pitch-class annotations are needed. Annotation can be done with MIDI-fied instruments [18], which led to several transcription datasets such as MAPS [19], SMD [20], or MAE-STRO [21]—all limited to the piano. For other instruments, there are only few pitch-annotated datasets such as Bach10 [22], TRIOS [23], or PHENICX-Anechoic [24] (all ≤ 10 pieces), which often involve multi-track recordings to simplify annotation. As an alternative, symbolic scores can be used to semi-automatically generate pitch-class annotations. While such scores are considered only “weak labels” for popular music [25], the correspondence between score and audio is clearly higher for professional recordings of classical music. For exploiting such score–audio pairs (as done for the MusicNet dataset [26]), automated music synchronization technology such as [27] is required in order to generate a so-called *strong alignment*. While this training strategy leads to effective pitch-class representations [28], the necessary synchronization constitutes a costly and challenging pre-processing step.

Motivation of our work. To simplify and improve this procedure, synchronisation between audio and labels can be done either within the *network* using attention models [29] or transformers [30], or within the *loss computation* using the Connectionist Temporal Classification (CTC) loss [31] for sequence-to-sequence training. CTC was successfully applied by Zalkow et al. [32] to train a monophonic deep-chroma representation from weakly aligned data, which they use for cross-modal retrieval. Yet, since CTC applies to single-label outputs, only monophonic pitch-class representations can be trained this way.

Proposal and paper organization. To overcome this limitation, we propose to use a multi-label variant of CTC (denoted MCTC), recently introduced for handwritten text and optical music recognition [33]. Based on our previous work on MCTC for multi-pitch estimation [34], we apply this loss to train a polyphonic pitch-class representations

from score–audio pairs of general correspondence (*weak alignment*) without the need for pre-computing *strong alignments*. Using MCTC, we train a network to detect the framewise activity of pitch classes as indicated by the score (*multi-pitch-class estimation*, see Figure 1).

Our main contributions are as follows: First, we re-formalize the MCTC loss to be applicable for PCP. Second, we use this loss to train a musically-motivated CNN inspired by [28] for extracting pitch-class representations. Using several public datasets, we perform experiments to analyze their efficacy and robustness against input modifications. Third, we propose a set of performance measures to directly evaluate the PCP quality without a side-task. Fourth, we demonstrate the potential of our MCTC-based features for visualization and for two downstream tasks: local key and chord estimation. We compare our features to several baselines including features trained with strongly aligned scores. All results indicate that MCTC is a promising tool for training efficient pitch-class representations with weakly aligned score–audio pairs.

2. MCTC LOSS FOR PITCH CLASS ESTIMATION

In the following, we describe and formalize the MCTC loss for training deep pitch-class features, closely following the descriptions in [33, 34] for comparability.

CTC. We consider a Neural Network (NN) which maps an input sequence $\mathbf{x} := (\mathbf{x}^1, \dots, \mathbf{x}^U)$ to an output sequence $\mathbf{y} := (\mathbf{y}^1, \dots, \mathbf{y}^T)$ with U possibly larger than T due to additional context frames. The CTC loss, initially proposed for speech recognition [31], allows to map the output sequence \mathbf{y} to a target sequence (or *label*) l of length $S \ll T$, $l := (l^1, \dots, l^S)$. l consists of *characters* $l^s \in L$ where L is an *alphabet*. A path π is a possible alignment between the two sequences \mathbf{y} and l . To compute the probability of l given \mathbf{x} , $p(l|\mathbf{x})$, we need to consider all possible (valid) paths π between \mathbf{y} and l . CTC requires an extra character *blank* (or “-”), which stands for either no symbol being active or a repetition of the previously active symbol. This results in an extended alphabet $L' = L \cup \{\text{blank}\}$. We then define a mapping function $\mathcal{B} : L'^T \rightarrow L^{S \leq T}$, which transforms a path $\pi = (\pi^1 \dots \pi^t \dots \pi^T) \in L'^T$ to a label $l = (l^1, \dots, l^S) \in L^S$ by removing repeated and then blank symbols.¹ Given a target sequence l , the inverse of \mathcal{B} or *pre-image* $\mathcal{B}^{-1}(l)$ provides the set of all valid paths π that collapse to l . In practice, \mathbf{y}^t is the output of a NN at time t with a softmax activation giving the likelihood of each character $k \in L'$ at time t . The probability of a given path π is the product of the relevant probabilities over time: $\prod_{t=1}^T y_{\pi^t}^t$. The probability of observing the label l is the sum over all its valid paths $\pi \in \mathcal{B}^{-1}(l)$:

$$p(l|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(l)} \prod_{t=1}^T y_{\pi^t}^t \tag{1}$$

CTC allows to compute this efficiently using dynamic programming and was used for MIR tasks such as lyrics alignment [35] or monophonic pitch-class representations [32].

¹ For example, if $T=5$ and $L = \{a, \dots, z\}$, $\mathcal{B}(a - ab -) = \mathcal{B}(aa - ab) = \mathcal{B}(-a - ab) = aab$.

Multi-label CTC (MCTC). The CTC loss is a useful tool for single-label problems. However, polyphonic pitch-class estimation is a multi-label problem, where an input \mathbf{x} needs to be mapped to several (non-mutually exclusive) target labels y_k . An extension of the CTC loss to the multi-label case has been proposed by Wigington et al. [33] for handwritten text recognition (letters with accents) and applied to multi-pitch estimation in our previous work [34].

In the case of PCP, any combination of pitch-classes is allowed to be simultaneously active (see Figure 1). Modeling all combinations as individual symbols would be possible in theory but leads to a large network output ($2^{12} = 4096$), thereby not accounting for the high interdependency of similar combinations, and is therefore not adequate for the problem. Following [34], we thus consider the 12 pitch classes $q_i \in \{C, C\#, \dots, B\}$ as different categories C_i , $i \in \{1, \dots, 12\}$, each of which comprises the same set of components: 0 (absence of this pitch-class), 1 (presence) together with the *blank* symbol. This leads to the alphabet $\{\text{blank}, 0, 1\}$. A character k is then the union (a tuple) of components from different categories, i. e. in our case, a multi-hot target vector denoting pitch-class activities $k \in \{0, 1\}^{12}$ denoting the co-occurrence of several pitch classes. The label l is a sequence of characters with vocabulary L (all binary pitch-class vectors). It can be decomposed into component-level label sequences l_i with vocabulary L_i . In the same spirit, at the path-level, we can define a character-level path π with vocabulary L' and a component-level path π_i with vocabulary L'_i . We now describe the realization of these sets for different multi-label variants of CTC proposed in [33].

Separable CTC Loss (SCTC). Assuming that pitch-classes occur independently of each other (which is of course a wrong assumption since e. g., the pitch classes of a chord are tied together), there is a trivial approach in which each category C_i is considered by an individual CTC loss. The individual losses are then multiplied:

$$p(l|\mathbf{x}) = \prod_{i=1}^{12} \sum_{\pi_i \in \mathcal{B}^{-1}(l_i)} \prod_{t=1}^T y_{i,\pi_i^t}^t. \quad (2)$$

For applying SCTC to pitch classes, $C_i = L'_i = \{\text{blank}, 0, 1\}$ and $L_i = \{0, 1\}$, resulting in $|C| = 12$ distinct categories. The input to the loss, which is the output of the network, is a tensor $y_{i \in \{1 \dots 12\}, k_i \in \{\text{blank}, 0, 1\}}^t \in [0, 1]^{12 \times 3}$ with softmax activation along the second dimension. It represents the probability of observing *blank*, not observing pitch-class q_i , or observing q_i at time t . Treating each pitch class as an independent sequence of components $\in \{\text{blank}, 0, 1\}$ makes their alignment difficult (no explicit modelling of pitch-class co-occurrence). We thus do not expect SCTC to work well in accordance with [33, 34].

MCTC Loss Without Epsilon (MCTC:NE). For correctly modelling the joint occurrence of pitch classes, we introduce the MCTC loss in its simplest form, the “no epsilon” variant MCTC:NE (details in the following). In this case, we have an individual blank_i symbol for each category so that $C_i = L'_i = \{\text{blank}_i, 0, 1\}$. Then, the set of all possible characters is $L' = L'_1 \times L'_2 \times \dots \times L'_{12}$. The

Table 1. CNN architecture. Depending on the loss used, we choose $Q \in \{1, 2, 3\}$ and $P \in \{0, 1\}$ appropriately.

Layer	Kernel size	Output shape	# Parameters
Layer norm.		$(T+74, 216, 6)$	2592
Conv2D, MaxPool	15×15	$(T+74, 216, 20)$	27020
Conv2D, MaxPool	3×3	$(T+74, 72, 20)$	3620
Conv2D	75×1	$(T, 72, 10)$	15010
Conv2D	1×1	$(T, 72, 1)$	11
Conv2D	1×61	$(T, 12+P, Q)$	$Q(62+73 \cdot P)$
Total			48253 + $Q(62+73 \cdot P)$

overall *blank* character is the combination of the blank components: $\text{blank}_{\text{MCTC}} = (\text{blank}_1, \dots, \text{blank}_{12})$. We compute the probability y_k^t of a character k at time t as the product of all its component probabilities $y_k^t = \prod_{i=1}^{12} y_{i,k_i}^t$:

$$p(l|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(l)} \prod_{t=1}^T \prod_{i=1}^{12} y_{i,\pi_i^t}^t. \quad (3)$$

In practice, we do this only for the characters k within the training batch. The input to the loss is a $\mathbf{y}^t \in [0, 1]^{12 \times 3}$ with softmax activation over the second dimension.

MCTC Loss With Epsilon (MCTC:WE). In the previous variant, the network has to simultaneously predict *blank* for all components individually in order to predict the $\text{blank}_{\text{MCTC}}$ character. As proposed in [33], there is a more elegant way of dealing with repetitions of the complete character (pitch-class vector): using an extra category. We therefore define the new category $C_1 = \{\text{blank}, \text{not blank}\}$. The remaining categories $C_2 \dots C_{13}$ correspond to the 12 pitch classes, as before. To ignore these categories when computing the $\text{blank}_{\text{MCTC}}$ probability, we introduce for them an additional ε symbol. This leads to $L'_i = \{0, 1, \varepsilon\}$ for $C_2 \dots C_{13}$. Please note that ε does not correspond to a network output but is only defined for mathematical convenience. In this scenario, the $\text{blank}_{\text{MCTC}}$ character is defined as $\text{blank}_{\text{MCTC}} = (\text{blank}, \varepsilon, \dots, \varepsilon)$ and all other characters are of the form $k = (\text{not blank}, 0/1, \dots, 0/1)$. Using this variant, it is also possible to explicitly model silence using the character $k = (\text{not blank}, 0, \dots, 0)$. We therefore compute

$$y_k^t = \begin{cases} y_{1,\text{blank}}^t & k = \text{blank}_{\text{MCTC}} \\ y_{1,\text{not blank}}^t \cdot \prod_{i=2}^{13} y_{i,k_i}^t & \text{otherwise.} \end{cases} \quad (4)$$

In this variant, $\text{blank}_{\text{MCTC}}$ can be used to repeat the whole character. Its probability is computed ignoring the other categories’ probabilities (ε). Here, the input to the loss is a tensor $\mathbf{y}^t \in [0, 1]^{13 \times 2}$ with softmax activation along the second dimension,² where the first category corresponds to *blank* and the other 12 categories to the pitch classes.

3. DEEP-LEARNING METHOD

To investigate the benefit of MCTC for training, we do neither use complex architectures such as CRNNs [36] or U-Nets [37] nor data augmentation strategies [38] but instead

² A 73×2 tensor with sigmoid activation is equivalent; softmax allows for using the numerically stable *logsoftmax* implementation of Pytorch.

Table 2. Overview of the datasets used in this work.

ID	Name	Instrum.	Annot. Strategy	hh:mm
Mae	MAESTRO [21] v3.0.0	Piano	MIDI piano	198:39
B10	Bach10 [22]	Violin, wind	Multitrack	0:06
Tri	TRIOS [23]	Chamber m.	Multitrack	0:03
PhA	PHENICX-Anechoic [24]	Orchestra	Multitrack	0:10
MuN	MusicNet [26]	Chamber m.	Aligned scores	34:08
SWD	Schubert Winterreise [41]	Piano, voice	Aligned scores	10:50

use a simple 5-layer CNN (Table 1). Inspired by [32, 39], we use a Harmonic Constant-Q Transform (HCQT) as input representations, with five harmonics and one sub-harmonic (six input channels). The audio sample rate is 22050 Hz, the CQT has a hopsize of 512 samples (roughly 43.07 Hz), and three bins per semitone over six octaves ($3 \cdot 6 \cdot 12 = 216$ bins). We use the librosa implementation of the CQT, which includes tuning estimation.³ The input is a HQCT tensor \mathbf{x} of shape $(T+74, 216, 6)$ (with 74 context frames, $U = T+74$) processed with log compression ($\gamma = 10$) and layer normalization [40].

To simulate the weakly aligned target label sequences $l := (l^1, \dots, l^S)$, we use strongly aligned target vectors $(\mathbf{y}^1, \dots, \mathbf{y}^T)$ and remove repeated vectors (see Figure 1c).

Following [28], we use a musically motivated architecture where the first layer performs pre-filtering using small rectangular kernels, followed by binning to the 72 pitches and temporal reduction (removing context frames). Next, we merge the channels with 1×1 convolutions. The final convolution reduces the 72 pitches to 12 pitch classes using a kernel of length 61. The exact output size depends on the loss used and is parameterized by P and Q : For our baseline (strongly aligned targets), we use the same architecture with binary cross entropy (BCE), then $P = 0$, $Q = 1$ and sigmoid activation for the output (see Table 1). For SCTC and MCTC:NE, we use $P = 0$, $Q = 3$ and softmax activation over the last dimension. For MCTC:WE, we need a further output dimension for the $blank_{MCTC}$, thus using $P = 1$ and $Q = 2$. Resulting from this, our network has roughly 48k parameters, slightly varying according to the loss used. We use LeakyReLU activations, max pooling, stochastic gradient descent with momentum (as in [38]), and learning rate scheduling. For strongly aligned training, we use mini-batches of size 25 and length $T=1$ or $U=75$. For MCTC, we use only one example (\mathbf{x}, l) per batch but of a considerably higher length (default $T = 500$ frames).⁴

4. DATASETS

For our experiments, we consider several datasets (Table 2) representing the annotation strategies introduced in Section 1. As a MIDI-piano dataset, we consider a subset (1/6) of MAESTRO (Mae). Moreover, we include the multi-pitch datasets Bach10 (B10), TRIOS (Tri), and PHENICX-Anechoic (PhA), whose pitch annotations are reduced to the pitch-class level. Furthermore, we use two datasets based on aligned scores: MusicNet (MuN), which comprises pitch annotations for 330 chamber music

³ <https://librosa.org/>.

⁴ Code: https://github.com/christofw/pitchclass_mctc/.

Table 3. Comparison of MCTC variants, trained on the dataset SWD in a performance (or version) split.

Model/Loss	P	R	F	CS	AP
All-Zero	0	0	0	0.486	0.211
CQT-Chroma	0.512	0.681	0.579	0.701	0.594
CNN – SCTC	0.850	0.048	0.090	0.520	0.416
CNN – MCTC:NE	0.747	0.775	0.758	0.802	0.798
CNN – MCTC:WE	0.762	0.853	0.802	0.830	0.851
CNN – Strong alignment	0.850	0.790	0.818	0.860	0.886

recordings, and the Schubert Winterreise Dataset (SWD), which comprises several performances, scores, and annotations of Franz Schubert’s song cycle *Winterreise*. For SWD, we use MIDI files and measure annotations together with a synchronization algorithm [27] to generate pitch-class annotations.⁵ For MCTC-based training, we do not need these strongly aligned pitch-class annotations but reduce the pre-aligned targets to a sequence of non-repeating vectors (see Figure 1c). For baseline experiments and for evaluation, we use the strongly aligned targets.

5. EXPERIMENTS

In the following, we present several experiments to assess the effectiveness of MCTC for pitch class estimation.

5.1 Evaluation Measures

All models output frame-wise probabilities \hat{y}_k^t (frame rate 43.07 Hz) for the activity of the twelve pitch classes. To directly measure the similarity between targets \mathbf{y}^t and predictions $\hat{\mathbf{y}}^t$ without a threshold (continuous-valued), we took inspiration from other music transcription tasks by using their Cosine Similarity (CS) and their Average Precision (AP)⁶ as in [38]. We also binarize $\hat{\mathbf{y}}^t$ using a threshold of 0.5 (motivated by the sigmoid/softmax outputs) to compute precision (P), recall (R), and F-measure (F).

5.2 Comparing MCTC Variants

First, we run a number of experiments to compare the different variants of MCTC (Table 3). To test generalization to new acoustic conditions, we consider a “version split” of the SWD [42, 43] with seven performances used for training and validation and two (HU33, SC06) for testing. To assess the effectiveness of MCTC, we consider several baselines:

All-zero: Since the majority of pitch classes are inactive more often than active, we compute our evaluation measures for an all-zero output. We obtain a cosine similarity of CS=0.486 and an average precision of AP=0.211.

CQT-Chroma: Next, we evaluate a CQT-based chroma as implemented in librosa (1 bin per semitone), followed by max-normalization and thresholding. This already leads to CS=0.701 and AP=0.594 as well as F=0.579.

CNN – Strong alignment: This is our central baseline, relying on the supervised training with pre-aligned annota-

⁵ <https://zenodo.org/record/5139893/>.

⁶ Average precision corresponds to the area under the precision–recall curve—a concept similar to Receiver-Operator-Characteristics (ROC).

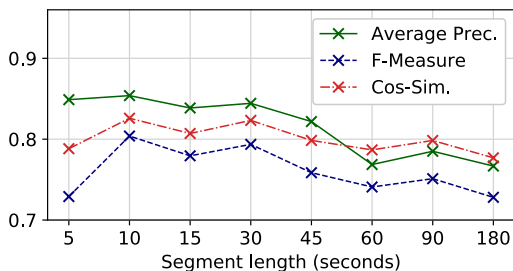


Figure 2. Pitch-class estimation results for different durations of MCTC training segments.

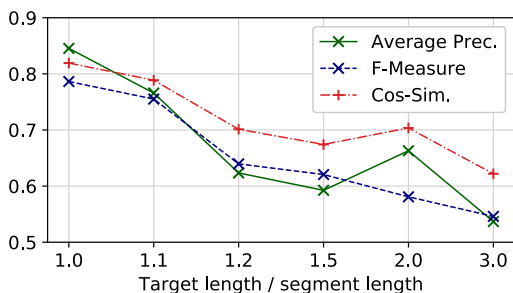


Figure 3. Estimation results for weakly corresponding target segments (until three times the input length).

tions and BCE loss. For this approach, we obtain $F=0.818$, $CS=0.860$ and $AP=0.886$, which are promising results.

Now, we train our CNN described in Section 3 with the different MCTC losses, feeding segments of length $T = 500$ frames (roughly 12 sec) plus context to the network, together with the unaligned pitch activity vectors of the segment as targets l . Similar to [33, 34], SCTC leads to poor performance—in our case with a very low recall and $CS=0.520$, only slightly above the all-zero baseline, which means that the network mostly predicts zero. For MCTC:NE, the results are better with $CS=0.802$ and $AP=0.798$. As in [33, 34], the MCTC:WE variant with an explicit $blank_{MCTC}$ produces the best results with $F=0.802$, $CS=0.830$, and $AP=0.851$. Though all results with MCTC:WE are below the strongly aligned baseline, the gap between the two approaches is small. We thus consider the MCTC:WE as a promising tool, which only requires weakly aligned data for training (and allows to scale up data more easily). For the following experiments, we only use the MCTC:WE variant (from now on: MCTC). Training time (per epoch) is longer for MCTC (by a factor of roughly 20) compared to strongly aligned training while convergence was faster with MCTC.

5.3 Sensitivity of MCTC-based Training

To investigate the behavior of MCTC-based training, we conduct two further systematic experiments.

Sensitivity to segment duration. First, we test the influence of the input segment length T (previously 500 frames or 12 sec). Since boundaries of input and target segments are musically corresponding, we expect shorter segments to result in a simpler alignment task for the loss, but longer segments to give more freedom for alignment. The results in Figure 2 confirms this assumption—a segment length of 10 sec is beneficial compared to 5 sec, and

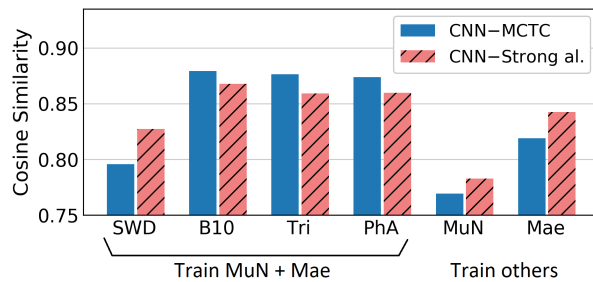


Figure 4. Results for the cross-dataset experiment.

the training behavior is quite stable until 30 sec length. For longer segments, the scores slowly drop. However, even a segment of 3 minutes length leads to a meaningful model, still outperforming the CQT baseline in Table 3. This encouraging result suggests that long score–audio segments of general correspondence can be used with MCTC, i. e. for classical music, short pieces or sections of long ones.

Sensitivity to segment mismatch. Next, we investigate the sensitivity of training when the boundaries of input x and target l segment are not perfectly corresponding (Figure 3). To this end, we use a target segment that corresponds to a longer input segment while the actual input segment is kept at a constant length of $T = 500$ frames. For a factor of 1.1 (target length 550 frames), performance only slightly decreases. In absolute time, this means that the target has one second more “pitch class information” than the network’s input. This scenario can be handled successfully by MCTC, which means that “even more weakly” aligned pairs are possible. For longer targets, performance drops—though training does not break down until a target context of three times the input segment length. We conclude that MCTC allows for quite some imprecision in the correspondence of the segment boundaries (up to one second).

5.4 Cross-Datasets Evaluation

Next, we test our MCTC training procedure on all datasets described in Section 4, covering various instrumentations and annotation strategies. Figure 4 shows the corresponding results. For the first four datasets (SWD, B10, Tri, PhA), we train on MuN and Mae. For SWD, these cross-dataset results are slightly worse than the results reported for cross-validation in Table 3. Evaluating on MuN (training on all others) leads to slightly worse CS; evaluating on Mae works better. For the larger datasets (SWD, MuN, Mae), strongly aligned training is superior to MCTC-based training. For the smaller transcription datasets (B10, Tri, PhA), the MCTC-based strategy obtains slightly better results. However, all differences are small. This is encouraging since with MCTC, larger training datasets can be easily achieved so that using larger networks is promising. In preliminary experiments with a larger CNN (more channels, 600k parameters), we already observed improved results.

5.5 Application: Visualization

As said, we aim at training a transcription-like representations capturing the pitch classes as indicated by the score. To illustrate this, we provide a visual example without

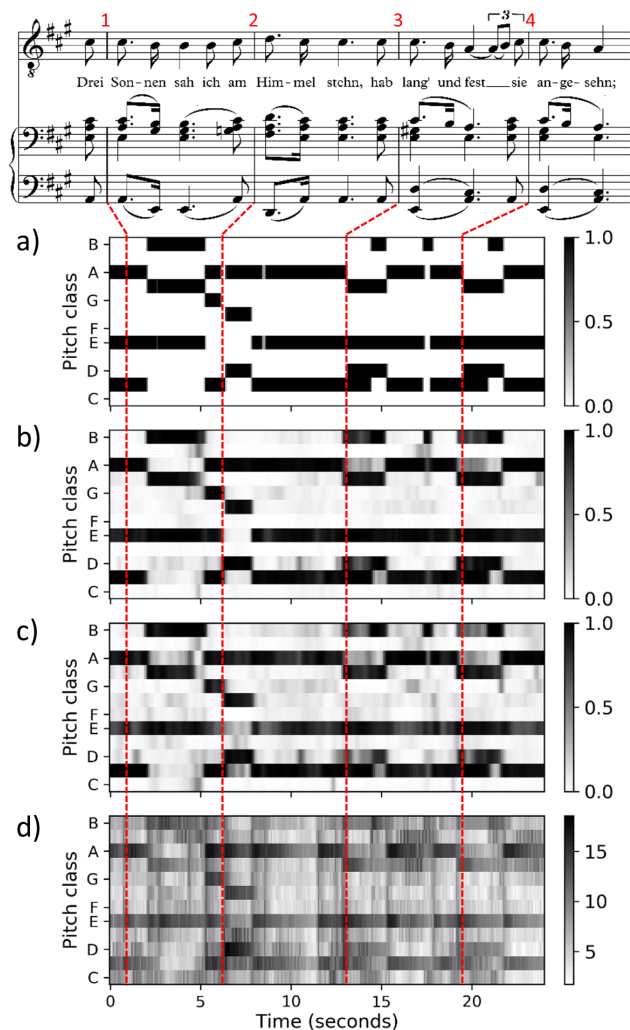


Figure 5. Example pitch-class features for an excerpt of Schubert’s *Winterreise* (Song No.23 sung by R.Trekel). (a) Pitch-class annotations from aligned score. (b) CNN’s pitch-class predictions trained with strongly aligned targets and (c) trained with MCTC. (d) CQT chroma features.

thresholding the outputs (Figure 5). Both strongly and weakly (MCTC) aligned training (on other tracks of *SWD*) lead to visualizations (Figure 5b+c) close to the score-based one (a), with the strongly aligned representation (cosine similarity 0.948 with the score) marginally “cleaner” than the MCTC one (CS=0.930). In contrast, the CQT chroma (Figure 5d) is less clear (CS=0.714) and exhibits the typical artifacts: First, a singer vibrato (e. g., for $C\sharp$ at 10 sec); second, overtones (e. g., B as overtone of E at 9 sec), and third, transient piano onsets (e. g., at 8 sec). All of these artifacts are suppressed by the trained CNNs.

5.6 Application: Chord and Local Key Estimation

Finally, we test the usefulness of our learned features for two harmony analysis tasks: chord recognition (for the 24 major and minor chords) [9] and local key estimation [42] using the respective annotations of *SWD* [41]. Having a score-like and interpretable feature at hand, we opt for a traditional system based on simple templates (thus allowing for defining chord or key templates only through music theory knowledge) and a Hidden Markov Model (HMM)

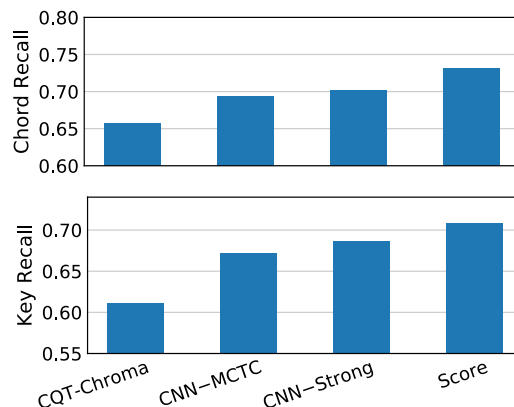


Figure 6. Chord (upper) and local key (lower) estimation results on *SWD* using different pitch-class features.

for context-sensitive smoothing (with uniform, diagonal-enhanced transition matrix, see [9]). This system does not require any pretraining. For both tasks, we compute the HMMs emission probabilities using the cosine similarity between PCP and templates. We downsample all pitch-class features to roughly 10 Hz.

Chord recognition. For this task, we set the HMM self-transition probability to $a_{i,i} = 0.1$ (i. e. $a_{i,j \neq i} = 0.9/23$ for all other transitions) and use binary chord templates (1 at the triad’s pitch classes, 0 otherwise). The results in Figure 6 (upper plot) are promising for such a simple system: The learned features (CNN–MCTC and CNN–Strong) outperform the CQT-based chroma and show a promising improvement towards score-based pitch classes (which are the targets y^t of our CNN training). Also, MCTC-based results are close to strongly aligned ones.

Local key estimation. For this task, we use log compression and median filtering (filter length 10 seconds) for pre-processing the PCPs, together with a higher self-transition probability $a_{i,i} = 0.5$. The key templates are simply based on music-theory (1 for scale pitch classes, 2 for the tonic triad, 0 otherwise). Again, learned features (CNN–MCTC and CNN–Strong) outperform CQT, now almost closing the gap towards the score-based features, and MCTC-based results are close to strongly aligned ones.

While these results do not reach the state of the art for both tasks (e. g. [14, 42]), they are promising for a purely hand-crafted system. Most remarkably, this strategy allows for an “objective” analysis since the systems’ parameters are specified in an explicit, musically motivated way.

6. CONCLUSION

In this paper, we presented a novel strategy for training pitch-class representations with weakly aligned score–audio pairs. To this end, we adapted a multi-label CTC loss, which led to a successful training close to the training with strongly aligned scores. Though being computationally more expensive, MCTC-based feature learning is a very promising direction since weakly aligned annotations for long segments of music can be created with much less effort, thus enabling an easier scalability to larger datasets, which allows for training more complex networks.

7. ACKNOWLEDGEMENTS

C. W. is funded by a research fellowship of the German Research Foundation (DFG WE 6611/1-1). We thank Curtis Wigington for advice on implementation and Meinard Müller and Frank Zalkow for fruitful discussions.

8. REFERENCES

- [1] M. A. Bartsch and G. H. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, USA, 2001, pp. 15–18.
- [2] T. Fujishima, "Realtime chord recognition of musical sound: A system using common lisp music," in *Proc. Int. Computer Music Conf. (ICMC)*, Beijing, China, 1999, pp. 464–467.
- [3] G. H. Wakefield, "Mathematical representation of joint time-chroma distributions," in *Proc. SPIE Conf. on Advanced Signal Processing Algorithms, Architecture and Implementations*, Denver, USA, 1999, pp. 637–645.
- [4] E. Gómez, "Tonal description of music audio signals," PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.
- [5] K. Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *Proc. Int. Computer Music Conf. (ICMC)*, New Orleans, USA, 2006, pp. 306–311.
- [6] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 135–140.
- [7] M. Müller, S. Ewert, and S. Kreuzer, "Making chroma features more robust to timbre changes," in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.
- [8] M. Müller, F. Kurth, and M. Clausen, "Chroma-based statistical audio features for audio matching," in *Proc. IEEE Workshop on Applications of Signal Processing (WASPAA)*, New Paltz, USA, 2005, pp. 275–278.
- [9] T. Cho and J. P. Bello, "On the relative importance of individual components of chord recognition systems," *IEEE/ACM Trans. Audio, Speech, and Language Processing*, vol. 22, no. 2, pp. 477–492, 2014.
- [10] E. J. Humphrey, T. Cho, and J. P. Bello, "Learning a robust tonnetz-space transform for automatic chord recognition," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012, pp. 453–456.
- [11] E. J. Humphrey and J. P. Bello, "Rethinking automatic chord recognition with convolutional neural networks," in *Proc. IEEE Int. Conf. on Machine Learning and Applications (ICMLA)*, Boca Raton, USA, 2012, pp. 357–362.
- [12] —, "From music audio to chord tablature: Teaching deep convolutional networks to play guitar," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014, pp. 6974–6978.
- [13] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, New York City, USA, 2016, pp. 37–43.
- [14] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, 2017, pp. 188–194.
- [15] G. Doras, F. Yesiler, J. Serrà, E. Gómez, and G. Peeters, "Combining musical features for cover detection," in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Montréal, Canada, 2020, pp. 279–286.
- [16] Y. Wu and W. Li, "Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model," *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 27, no. 2, pp. 355–366, 2019.
- [17] Y. Wu, T. Carsault, and K. Yoshii, "Automatic chord estimation based on a frame-wise convolutional recurrent neural network with non-aligned annotations," in *Proc. European Signal Processing Conf. (EUSIPCO)*, A Coruña, Spain, 2019, pp. 1–5.
- [18] L. Su and Y. Yang, "Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription," in *Proc. 11th Int. Symposium on Computer Music Multidisciplinary Research (CMMR)*, ser. Lecture Notes in Computer Science, Plymouth, UK, 2015, pp. 309–321.
- [19] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.
- [20] M. Müller, V. Konz, W. Bogler, and V. Arifi-Müller, "Saarland music data (SMD)," in *Demos and Late Breaking News of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Miami, USA, 2011.
- [21] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proc. Int. Conf.*

- on *Learning Representations (ICLR)*, New Orleans, USA, 2019.
- [22] Z. Duan, B. Pardo, and C. Zhang, “Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions,” *IEEE Trans. Audio, Speech & Language Processing*, vol. 18, no. 8, pp. 2121–2133, 2010.
- [23] J. Fritsch and M. D. Plumbley, “Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis,” in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013, pp. 888–891.
- [24] M. Miron, J. Carabias-Orti, J. Bosch, E. Gómez, and J. Janer, “Score-informed source separation for multichannel orchestral recordings,” *Journal of Electrical and Computer Engineering*, vol. 2016, 2016.
- [25] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, “Automatic music transcription: An overview,” *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.
- [26] J. Thickstun, Z. Harchaoui, and S. M. Kakade, “Learning features of music from scratch,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, Toulon, France, 2017.
- [27] S. Ewert, M. Müller, and P. Grosche, “High resolution audio synchronization using chroma onset features,” in *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2009, pp. 1869–1872.
- [28] C. Weiß, J. Zeitler, T. Zunner, F. Schuberth, and M. Müller, “Learning pitch-class representations from score–audio pairs of classical music,” in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Online, 2021.
- [29] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. Int. Conf. on Learning Representations (ICLR)*, San Diego, USA, 2015.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems 30: Annual Conf. on Neural Information Processing Systems (NeurIPS)*, Long Beach, USA, 2017.
- [31] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” in *Proc. Int. Conf. on Machine Learning (ICML)*, Pittsburgh, USA, 2006, pp. 369–376.
- [32] F. Zalkow and M. Müller, “Using weakly aligned score–audio pairs to train deep chroma models for cross-modal music retrieval,” in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Montréal, Canada, 2020, pp. 184–191.
- [33] C. Wigington, B. L. Price, and S. Cohen, “Multi-label connectionist temporal classification,” in *Proc. Int. Conf. on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, 2019, pp. 979–986.
- [34] C. Weiß and G. Peeters, “Learning multi-pitch estimation from weakly aligned score–audio pairs using a multi-label CTC loss,” in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, USA, 2021.
- [35] D. Stoller, S. Durand, and S. Ewert, “End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model,” in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 181–185.
- [36] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Paris, France, 2018, pp. 50–57.
- [37] J. Abeßer and M. Müller, “Jazz bass transcription using a U-net architecture,” *Electronics*, vol. 10, no. 6, pp. 670:1–11, 2021.
- [38] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, “Invariances and data augmentation for supervised music transcription,” in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 2241–2245.
- [39] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, “Deep salience representations for F0 tracking in polyphonic music,” in *Proc. Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, Suzhou, China, 2017, pp. 63–70.
- [40] L. J. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *CoRR*, vol. abs/1607.06450, 2016.
- [41] C. Weiß, F. Zalkow, V. Arifi-Müller, M. Müller, H. V. Koops, A. Volk, and H. G. Grohgan, “Schubert Winterreise dataset: A multimodal scenario for music analysis,” *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 14, no. 2, pp. 25:1–18, 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5139893>
- [42] H. Schreiber, C. Weiß, and M. Müller, “Local key estimation in classical music recordings: A cross-version study on Schubert’s Winterreise,” in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, Barcelona, Spain, 2020, pp. 501–505.
- [43] C. Weiß, H. Schreiber, and M. Müller, “Local key estimation in music recordings: A case study across songs, versions, and annotators,” *IEEE/ACM Trans. Audio, Speech & Language Processing*, vol. 28, pp. 2919–2932, 2020.

AUDIO DEFECT DETECTION IN MUSIC WITH DEEP NETWORKS

Daniel Wolff, Rémi Mignot and Axel Roebel

Analysis-Synthesis Team, UMR 9912 STMS - IRCAM, CNRS, Sorbonne Université, Paris, France

{daniel.wolff, remi.mignot, axel.roebel}@ircam.fr

ABSTRACT

With increasing amounts of music being digitally transferred from production to distribution, automatic means of determining media quality are needed. Protection mechanisms in digital audio processing tools have not eliminated the need of production entities located downstream the distribution chain to assess audio quality and detect defects inserted further upstream. Such analysis often relies on the received audio and scarce meta-data alone. Deliberate use of artefacts such as clicks in popular music as well as more recent defects stemming from corruption in modern audio encodings call for data-centric and context-sensitive solutions for detection. We present a convolutional network architecture following end-to-end encoder-decoder configuration to develop detectors for two exemplary audio defects. A click detector is trained and compared to a traditional signal processing method, with a discussion on context sensitivity. Additional post-processing is used for data augmentation and workflow simulation. The ability of our models to capture variance is explored in a detector for artefacts from decompression of corrupted MP3 compressed audio. For both tasks we describe the synthetic generation of artefacts for controlled detector training and evaluation. We evaluate our detectors on the large open-source Free Music Archive (FMA) and genre-specific datasets.

1. INTRODUCTION

In recent decades, digital means of media delivery have become increasingly popular, not least with the advent of high-speed internet and the ubiquity of digital playback and recording devices ranging from studio digital signal processing hardware to MP3 players and mobile phones. The greater availability of technology required to produce digital media now allows for small-scale studios to create high quality content and instantly transfer it to distributors such as music labels.

Digital media files can suffer from various degradations that occur during transport and processing of the media. Automatic means exist for detection and correction of certain data errors in uncompressed audio, but many complex

defects remain untackled. In this paper, we explore the efficacy of applying a data-driven machine learning approach using Deep Neural Networks to two audio defects, which music labels are confronted with in quality assurance of incoming media.

For our first scenario, we define clicks as discontinuities affecting a few signal samples, resulting in very short broadband impulses. A prominent source of similar artefacts are buffer under-runs, where, due to synchronisation issues during digital audio processing, a few samples of an old or zeroed signal are transmitted instead of the current signal. Although existing mastering software offers methods to remedy similar artefacts such as clicks, crackle and clipping, restoration often requires manual selection of noise profile, target segments or thresholds. This is due to ambiguities introduced by e.g. signal quality or, depending on the genre, degradations voluntarily applied to audio as effects, rendering current methods costly in large-scale application.

Our hypothesis is that, using a data-driven approach, our network can distinguish deliberate clicks (such as electronic snare drums) from defects, and thereby enable automatic processing of large electronic music corpora. This is an essential challenge in our scenario, where manual inspection may not be feasible because of operational constraints.

In our second scenario, we aim to detect corruption of binary MP3¹ data via the artefacts audible after decoding such files. The MP3 audio encoding family uses a psychoacoustic model to guide data reduction. Due to the transformations applied to data during MP3 decoding, a large variation of effects is possible, ranging from added whistling noise over various missing frequency bands to broadband noise. In contrast to noise normally added during the process of lossy encoding itself, the artefacts stemming from unnoticed corruption have not been approached for detection yet.

Such MP3 corruption may happen within a production chain where manual transcoding is performed by different production agents, as illustrated with the following use case: A distributor receives a sound file in lossless format (e.g. pulse-code modulation format .wav) as a studio quality delivery from a production entity. A degradation is detected at this point by listening to the audio. During discussion of this defect, it is found that the production entity sent a transcoded file they prepared for checking an



© Daniel Wolff, Rémi Mignot and Axel Roebel. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Daniel Wolff, Rémi Mignot and Axel Roebel, "Audio Defect Detection in Music with Deep Networks", in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ See standards <https://www.iso.org/standard/19180.html> and <https://www.iso.org/standard/31537.html>

earlier, defect MP3 delivery. The decoded MP3 was transmitted rather than the original .wav file intended for the delivery. Having no dependable meta-data on intermediary processing, the distributor can only rely on the audio itself for quality assurance. We found the above case to have practical relevance in the music industry, and designed our method to aid error detection in similar circumstances.

Our two detection scenarios are chosen to benefit from our data-driven approach: while the click detection may make use of signal context to determine the “musicality” of a click candidate, the MP3 glitch detector can benefit from the network’s ability to capture the variation of artefacts during training. In contrast to the click artefacts, the artistic use of the MP3 glitch artefacts is currently limited, reducing the chance to confuse such intentional use as degradation. The resulting models are designed for robust scanning of large media libraries in an unsupervised batch processing scenario.

In the following we present: an adaptation of the Wave-U-Net deep architecture to the detection and localisation of audio defects, two separate detectors built upon this to respectively detect clicks and MP3 glitches, methods for simulation of these artefacts assuring significance of inserted defects, and a large-scale evaluation against several music datasets.

2. RELATED WORK

Traditional methods for detecting clicks and similar non-stationary noise in audio aim at detecting discontinuities, using autoregressive modelling in the raw audio or sparse optimisation in the spectral domain [1, 2]. For restoration of analog media, where issues like clicks, overload and high frequency noise are common, wavelet-based implementations are used in commercial noise removal software [3,4]. Recently, Deep Neural Networks have been used for noise and obstruction removal mostly in images but also for audio and other time-based signals [5–7]. Removal of obstructions from images is a task particularly close to our task as it deals with less stationary noise [8, 9]. Matsui et al. [10] use a convolutional neural network similar to our architecture to remove fences from images. Restoration methods differ from our detection task in that explicit detection reporting and evaluation of false positives are not needed when applied as an audio de-noising effect.

For lossy audio encodings such as MP3, the artefacts arising during encoding, mainly as a trade-off between quality and bitrate, vary with methods and encoders and have been extensively discussed for the MP3 format [11]. Nevertheless, corruption of files introduces new, different artefacts that may remain unnoticed during decoding and thus are the subject of our detector.

Research in the related field of Computational Auditory Scene Analysis (CASA) concerns itself with the detection and labelling of events in an audio stream. Deep Neural Networks are now being increasingly used in this field, often with a spectral feature extraction pre-processing step. Mesaros et al. [12] report the detection of “clicks” as one of 61 classes in a detection task on their private dataset,

with a recognition rate of around 65 percent. It is not clear how these clicks compare to the clicks stemming from digitally signal failures which we tackle in this paper. The focus in CASA is to detect recorded sounds while being robust to noise in the recording, thus clicks to be detected would relate to physical events (see “mouse click” in the CASA dataset [13]). From a CASA perspective, intentional clicks, as frequently found in electronic music, would not necessarily be distinguished from those stemming from defects.

This difficulty of ambiguity in audio defect annotation is noted by Alonso-Jiménez et al. [14] who describe their implementation and evaluation of established audio-defect detection algorithms. Their optimised algorithms detect a significant number of audio defects in a database from a commercial streaming service, noting that there may be a long tail of likely, but undetected degradations. In our experiments (Section 6.1) we evaluate their click detector implementation after Vaseghi [15], complementing their results with accuracy metrics on a large dataset containing synthesized defects.

Research on error concealing scenarios, where defects are already identified, e.g. due to missing packets at the network layer during transport of an audio stream [16], can inform us of the complexity and artefacts resulting from such failures. Deep convolutional networks have been recently introduced into the field of audio inpainting - filling a gap in audio in such a way that the error is concealed - to great effect [17, 18]. Their ability to encode and model variation in large datasets results in more intelligible speech reconstruction, when compared to conventional concealment approaches.

We aim to exploit gains from training with large datasets for our detectors. Although large datasets of music, such as the FMA dataset, are available, we are not aware of any open datasets with audio defect labels. This may be due to the fact that defects are mainly corrected in production, and are - in terms of playback time - very rare. The task of audio anomaly detection deals with this issue using unsupervised learning to model usual/common signals on unlabeled data. Autoregressive networks [19] and autoencoders [20] have recently been used to detect unusual acoustic events via their high reconstruction loss after such training. Our architecture is similar to the above, in that it shares an information bottleneck to learn representations, but we use a supervised learning approach with synthetic examples similar to the work in [21] to better control the type of artefacts detected, avoiding the detection of e.g. new instruments or audio samples as anomalies.

Ronneberger et al. [22] originally presented the U-Net as a deep convolutional neural network for the task of segmenting biomedical images. The network structure allows for efficient learning of spatial, or - in the case of audio - temporal and frequency patterns. It has since been used for various end-to-end audio transformation tasks [23]. Stoller et al. [24] modified this structure to work directly on the one-dimensional audio signal as input, resulting in the

Wave-U-Net. In their source-separation task, they employ the network on overlapping excerpts of the original signal at a low 8kHz sample rate.

3. MODEL ARCHITECTURE

In this paper we introduce the Hook-Net as a novel adaptation of the Wave-U-Net for the task of detecting artefacts in audio signals. We apply this model as an end-to-end approach, feeding raw waveform segments into the network to receive a time series of classification results. After initial experiments with a U-Net on spectrogram features, we found training to be more effective when using raw audio input, which may be due to the extreme brevity of our defects. The time-aligning horizontal connections of input and output promise to help capture the context of distortions in the input waveform.

Our network takes as input segments 16384 samples of audio (at 44100Hz sample rate) and outputs a time series of 128 output samples that are individually quantised to the binary decision on original vs. degraded. The original Wave-U-Net implies that input and output share the same sampling rate. To reduce computational cost while operating at source input sample rates, the Hook-Net introduces an imbalance: the output time resolution is reduced by a factor of 128 with regards to the input sample rate, resulting in a classification sample rate of 344.5Hz.²

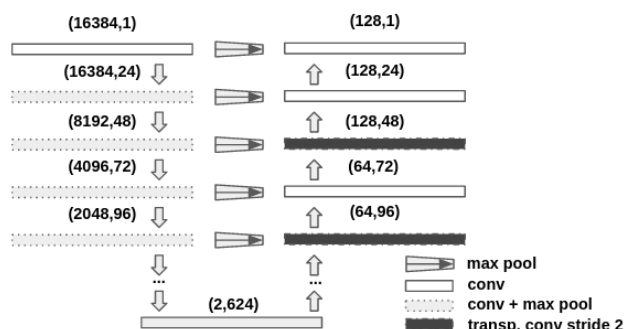


Figure 1. Architecture of the Hook-Net

This is reflected in the network structure as displayed in Figure 1. Our contracting (left side) path consists in blocks comprising two sequences of (zero-padded convolution – batch-norm – activation) layers followed by max-pooling. Here, the temporal resolution is halved every block, while the number of filters increases. An expanding block consists of an upscaling operation, followed by concatenation of the skip connection from the contracting path and a regular convolution layer. In our model, resolution of the axis mapped to time in our output is only doubled every second block - resulting in a reduced temporal resolution at the network output.

For the upscaling operation we follow the original U-Net [22], but employ transposed convolution of stride 2 only every second layer, and stride 1 otherwise. Thereby some steadiness remains in the growth of resolution across

² This also allows comparisons with spectrogram-based models, which did not perform as well and are omitted for brevity.

the expanding path, despite the reduced final output resolution. For the horizontal skip-connections, connecting the contracting and expansive path, we use max-pooling to adapt the time resolutions between the corresponding layers. We furthermore add vertical skip connections on the contracting path, bridging every block of two convolutions. This strategy is motivated by the training benefits reported in residual networks [25].

In the following we describe the generation of click and MP3 glitch artefacts.

4. CLICK ARTEFACT GENERATION

Within the scope of this paper, a click degradation corresponds to a fault in the already digitised signal: we define a click as a discontinuity, where the signal changes sharply to a random value for 1-3 samples but then continues unchanged. With this definition we aim to cover and simulate defects from digital signal transport, commonly resulting from buffer under-run during playback or mixing in a DAW or timing errors during digital transport over wire.

Clicks are inserted on-the-fly into the network input audio segments for training, validation and test scenarios. The position of the click is randomised, and one click is inserted with a probability of 0.1 per audio segment, with a small variation in length. The amplitude value of the click is calculated as a random offset of the current signal, from a uniform distribution within [0.3, 1). Each initial offset sign is chosen randomly, then signs that would create clipping are inverted.

The minimum amplitude offset (0.3) of the inserted clicks’ amplitudes is introduced to assure that the signal change and resulting degradation is significant. In absence of perceptual data on the acoustic salience of inserted artefacts, this heuristic should create clicks that are likely to be audible, where they are not perceptually masked by close preceding transients or loud broad-band noise.

4.1 Audio post-processing via SoX

In order to simulate potential post-processing effects that may have occurred on audio signals with previously undetected clicks, we use the SoX³ sound processor to slightly alter the audio segments after the click-insertion steps. Segments are post-processed regardless of whether a click has actually been inserted. A random combination of reverb, two-band EQ and compression is applied, and strength and filter parameters are chosen randomly within ranges that apply only mild changes to the signal. In Section 6.1 we report results for click detector training and detection with and without post-processing.

4.2 Click target vector

The target vector for training and testing of the detector is a 128-component floating point vector that is set to 1 on the (resampled) position corresponding to the location of

³ <http://sox.sourceforge.net/>

an inserted click, and 0 otherwise. In our experiments we simulate the problem of rare clicks that may be overlooked during production. There is at most 1 click per input segment.

5. MP3 GLITCH ARTEFACT GENERATION

This use-case tackles degradations that result from data corruption in the commonly used MPEG-1/2 Audio Layer III (MP3) lossy audio compression format. We will refer to these as *glitch* defects. This degradation is interesting as it can easily be “overlooked” in quality assurance. Moreover, the generation approach described below can be generalised to other audio codecs.

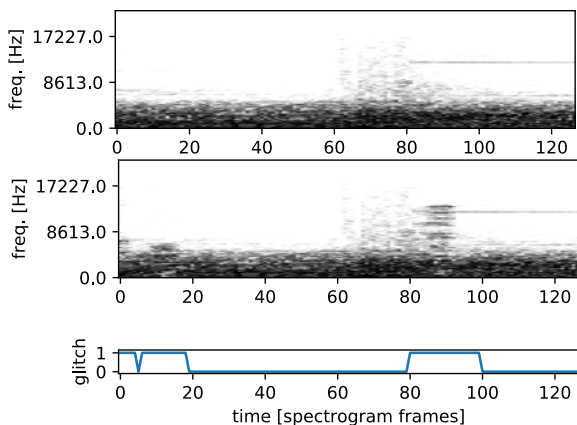


Figure 2. Typical example of glitch degradation. Top: linear frequency spectrogram of original signal. Bottom: same for signal with two degradations at frame 0 (intermittent) and frame 80 with binary ground truth annotations from spectrogram comparison.

The acoustic shape of the degradation varies strongly and often depends on the content and amplitude of the surrounding audio data. Fig.2 displays a typical glitch artefact: the degradation consists in added high-frequency content. This effect appears when comparing the original (top) and degraded (bottom) spectrogram contents between 80-100 frames. The same example features another glitch effect at the start of the frame, affecting a lower part of the spectrum. Acoustically, the effect is similar to a short whistle. An audio example demonstrating glitch artefacts is available online⁴. Note that the overall signal energy does not always change. The often well-embedded and adaptive nature of these glitches render automatic detection difficult.

5.1 Simulation of data corruption

Unlike the click degradations, which can be inserted on-the-fly as in Section 4, MP3 glitch degradations are calculated on a per-piece basis before training due to the less easily indexable file format of MP3 which hinders exact seeking.

In order to simulate data corruption in the compressed format, we modify MP3 encoded data during a decoding process which we survey on a frame by frame basis: the MP3 format encodes an audio stream into a series of MP3 frames. Each of these frames contains a header, containing format information and parameters of the MP3 encoding process itself. An optional integrity check for the header is often omitted to save bitrate. The `lame`⁵ encoder used in the present study does not include CRC checking. The header is followed by the encoded audio data, the size of this block being determined by the bitrate used for the frame during encoding. If a corrupted file remains undetected before and during decoding, introduced errors may be audible as glitches in playback, but become (from a data integrity point of view) undistinguishable from the original signal. This may easily happen if a command-line decoding tool only issues a warning in case of a data corruption which the decoding process can recover from in successive frames.

We generate glitches as follows: First, the input audio is transcoded to 128kbps mono MP3 files. During the following decoding, frames are randomly selected for glitch insertion given a probability of $P_{glitch} = 0.05$. In contrast to click generation we control glitch likelihood per MP3 frame. The data in these frames is then partially overwritten with random data of a randomised length. We found an average overwritten range of 120 bytes (an average frame contains 418 bytes) with a standard deviation of 60 bytes to give realistic results. No distinction was made between the header and data sections of the MP3 frame. In the rare case that the decoding of the degraded frame is not possible, the original MP3 frame is decoded and treated as non-degenerated. The same method is used where the decoded data is of a different length than the original undegraded signal. Frames not selected for glitch insertion are decoded inbetween such that the degraded decoding of the MP3 data results in a file with the same length as the original.

5.2 MP3 glitch target vector

MP3 glitches are inserted during pre-processing. To identify the actual parts of the decoded wave signal affected by the glitch artefact after decoding and segmentation, we employ a spectral distance measure comparing the degradedly decoded audio to a clean decoding of the audio. Using a frame wise thresholded difference of a power spectrogram at the frame rate of the network output (128 spectrogram frames), we determine whether significant degradation has taken place for each of the 128 output values. Fig. 2 (bottom) shows a resulting glitch classification target.

6. EXPERIMENTS

We compare our detectors on the Creative-Commons licensed FMA-Large⁶ dataset, containing roughly 30-second snippets of 106,574 tracks from 16341 artists

⁴ https://osf.io/uqner/?view_only=042774933537440299dd48a4083305b1

⁵ <https://linux.die.net/man/1/lame>

⁶ Online at <https://github.com/mdeff/fma/>.

within 161 genres, the most frequent being "Experimental", "Electronic", and "Rock" [26]. The music is stored in 320kbps stereo MP3 format.

Depending on the scenario, the data is either decompressed and degraded on-the-fly, or, as in the MP3 glitch scenario, data is already pre-processed with degradations added, and loaded as raw waveform alongside corresponding ground truth data. The Hook-Net models take as input audio segments of 16384 samples at a sampling rate of 44.1kHz, corresponding to 0.37 seconds.

Degraded (as well as non-degraded) audio and target data are then used for network training, using a batch size of 200 segments. During configuration of the Hook-Net, we tested variations on general parameters such as numbers of filters and found the models with 13 contracting/expanding blocks (see Section 3), 15 filters per contraction and 5 filters per expansion, totalling at 27,307,633 trainable parameters, performing well for our tasks at hand. Initial experiments consistently showed reduced precision in smaller models. Training is performed in Tensorflow, using the ADAM [27] optimiser, on single Nvidia V100 GPU.

6.1 Click detection

We selected a subset of 57,928 pieces from FMA-Large for training and evaluation of our click detector Hook-Net. Pieces were separated into non-overlapping training (40,000 pieces), validation (8964 pieces), and test (8964 pieces) sets. For each piece, 50 consecutive audio segments (18.6s in total) were extracted with no overlap, resulting in 2,000,000 training segments and 448,200 validation and 448,200 test segments. In the above datasets, 0.078% of the individual target values is set to 1 (on average, in every 10th segment, one value in the target vector is marked as containing a click), which is reflected in the initial setting of the network outputs' activation biases. Evaluation is performed on the target values as smallest units, not summarised at the segment level. The model was trained using a root mean square loss weighted towards the "click" class with a learning rate of 0.001. We report results from models of the epoch with best validation set accuracy.

As a baseline for the click detection task we apply a generic click detector (SigClick) as implemented⁷ in the open source Essentia library [28] to the data segments. To apply this to our segments of 16384 samples, we add an additional sub-segmentation step, using sub-segments of 4096 samples length, with an overlap of 2048 samples. Click positions returned by SigClick are transformed into a binary output vector of 128 samples.

Table 1 compares the test set performance of the Hook-Net to that of the best performing (by highest validation set accuracy) configuration (threshold 35) of the SigClick detector between tested thresholds of 30(default), 33, 35, 40, and 50. The Hook-Net models were selected by highest validation set accuracy, which was achieved after 13

	test data	acc_t	pr_t	rec_t	f1_t
Hook-Net	FMA	99.9993	99.77	99.24	99.47
SigClick	FMA	99.95	84.39	90.52	84.60
Hook-Net	FMA _{post}	99.9991	99.70	99.02	99.32
Hook-Net _{post}	FMA _{post}	99.9995	99.86	99.11	99.46
SigClick	FMA _{post}	99.97	85.52	81.04	80.62

Table 1. Click detector performances for plain (top) and post-processed (bottom) click-degraded data (higher is better, best per dataset in bold): test-set accuracy (acc_t), test precision(pr_t), recall(rec_t) and f-measure (f1_t) for click detection. In percent.

epochs (36 total) for the click data (Hook-Net) and 10 epochs (14 total) for training with post-processed click data (Hook-Net_{post}).

Due to the bias of the dataset and target vectors towards 0 (no click), we concentrate on precision and recall which is reported for the click class. For the application of finding defects in large commercial databases, the precision of click detection is of great importance. Top of Table 1 shows the Hook-Net achieves significantly better precision and recall of clicks. The bottom of the table confirms this using post-processed click data as test set. Added variation from post-processing results in lower recall values, particularly with SigClick. The difference in training with (Hook-Net_{post}) and without (Hook-Net) post-processing shows good generalisation from the generated click artefacts to the diverse post-processed artefacts.

These results highlight a critical difference in the data-driven paradigm applied in network training versus the more generalistic detection in SigClick: while the assumption for a general click detector is to detect any existing click (given other preconditions e.g. sufficient distance to the preceding one), the Hook-Net has been trained to detect our inserted clicks while ignoring other click instances in the music. Note that we do only compare the performances on our clicks simulating digital defects such as buffer under-run and timing errors. The SigClick detector may detect a wider range of clicks, but for this initial experiment we refrained from training more generic models due to the lack of datasets with clicks stemming from defects in music production, as we aim to minimise false positives.

Manual verification showed that the FMA dataset does feature many examples of electronic "glitch" music with intentional or expected clicks. The goal of our detector is to differentiate such clicks from the ones added due to signal failures.

We validate this relation to musical genre in Table 2 by applying the above model and SigClick on three smaller genre-consistent datasets of 11400 (electronic), 10000 (pop rock) and 31600 (classical) segments not included in the former training. Following the hypothesis that the electronic genre features more intentional click samples than the classical genre, we see the precision of SigClick very high for the classical data but significantly dropping in the rock/pop and electronic genres, while the Hook-Net maintains high precision with only little difference. The values

⁷ https://essentia.upf.edu/reference/std_ClickDetector.html

	dataset	acc_t	pr_t	rec_t	f1_t
Hook-Net	electronic	99.9997	99.91	99.74	99.82
SigClick	electronic	99.94	80.34	98.06	84.94
Hook-Net	rock pop	99.99992	100.0	99.90	99.95
SigClick	rock pop	99.99	92.08	94.37	92.05
Hook-Net	classical	100.0	100.0	100.0	100.0
SigClick	classical	99.992	92.47	99.11	95.34

Table 2. Click performance in control datasets. Test-set accuracy (acc_t), test precision(pr_t), recall(rec_t) and f-measure (f1_t).

	ep.	acc_v	acc_t	pr_t	rec_t	f1_t
Hook-Net	31	98.52	98.46	92.83	88.04	90.37

Table 3. Glitch detection performance. Epoch with greatest validation-set accuracy, validation accuracy (acc_v), test-set accuracy (acc_t), test precision(pr_t), recall(rec_t) and f-measure (f1_t) for glitch detection. In percent.

for post-processed data not reported here due to space limitations confirm this effect.

6.2 MP3 glitch detection

For this task, pre-caching of degraded audio allows us to use larger subsets of FMA for training (66476 pieces), validation (14244 pieces) and test (14244 pieces).

For each piece, 50 consecutive audio segments were extracted as above, resulting in 3,323,800 training segments and 712,200 validation and test segments. Segments were randomised within each of the above datasets. While a click only results in 1 target value to be set to 1, glitches affect multiple target values per segment due to whole MP3 frames being affected by each corruption. This results in train and validation datasets containing 8.04% of the target values marked as glitched (test set: 8.19%).

Training is performed using root mean square loss, with initial learning rate of 0.001 and reduction-on-plateau (factor 0.1, patience 10 epochs) of the learning rate. Across 40 training epochs, the best model was selected based on its validation accuracy (acc_v) measure. Table 3 shows the performance of this best model. Given the bias towards the non-degraded class we report f-measure and precision regarding the glitched class due to their relevance for our application scenario. The Hook-Net glitch model generalises well from the validation to the test set, with an f1-measure of 0.9037.

7. CONCLUSIONS AND FUTURE WORK

We presented the Hook-Net convolutional neural network architecture with two novel applications to detect click and data corruption errors in digital audio recordings. The design goal of the architecture is to capture the typical shape and variation of artefacts in the direct audio signal, with respect to their acoustic context. The detectors localise errors with a time resolution of less than 10 milliseconds.

For click detection, we demonstrated an end-to-end simulation, post-processing and detector training method. Our evaluation shows the resulting detector outperforms a

state-of-the-art baseline on the large FMA popular music dataset, using synthetically generated defects. A genre-specific evaluation experiment shows the practical relevance of inclusion of context and the capability of our model to capture this: discontinuities resembling clicks in audio may represent intentional music content depending on their context. The achieved precision (> 99.77%) renders the detector suitable for the testing of large commercial music databases.

Our second proposed application is aimed to filter degradations in modern compressed audio from persisting unnoticed in subsequent use of the defect audio. We describe the MP3 decoding glitch as a relatively novel type of audio degradation for which detection is difficult due to its variation. Our evaluation shows that our detector generalises well on common glitch artefacts. The proposed training tasks come with a bias towards non-defective audio, which we assume to be strong in real-world applications. This is tackled using large training and validation datasets with synthesised artefact insertion. The simulation of data corruption with subsequential MP3 decoding promises a stronger realism of the artefacts synthesized in this task. This process is applicable to other audio codecs, depending on decoder robustness and consistency checks.

In future work we plan to extend the range of lossy compression defects simulated and apply our architecture to further and more generalised local audio degradations. We also aim to reduce model complexity without negatively affecting performance.

8. ACKNOWLEDGEMENTS

This work was performed using HPC resources from GENCI-IDRIS (Grant 2020-AD011011379).

9. REFERENCES

- [1] S. Godsill, P. Rayner, and O. Cappé, *Digital Audio Restoration*. Springer, 2006.
- [2] A. Adler, V. Emiya, M. Jafari, M. Elad, R. Gribonval, and M. Plumbley, “A constrained matching pursuit approach to audio declipping,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, 2011.
- [3] J. Berger, R. Coifman, and M. Goldberg, “Removing noise from music using local trigonometric bases and wavelet packets,” *Journal of the Audio Engineering Society*, vol. 42, no. 10, pp. 808–818, 1994.
- [4] R. Laney, “Automatic detection of flaws in recorded music using wavelet fingerprinting,” Ph.D. dissertation, College of William and Mary, USA, 2011.
- [5] J.-M. Valin, “A hybrid dsp/deep learning approach to real-time full-band speech enhancement,” in *IEEE 20th International Workshop on Multimedia Signal Processing*, Vancouver, 2018.

- [6] A. Maas, Q. V. Le, T. M. O’Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, “Recurrent neural networks for noise reduction in robust ASR,” in *13th Annual Conf. of the International Speech Communication Association*, Portland, 2012.
- [7] H.-T. Chiang, Y.-Y. Hsieh, S.-W. Fu, K.-H. Hung, Y. Tsao, and S.-Y. Chien, “Noise reduction in ecg signals using fully convolutional denoising autoencoders,” *IEEE Access*, vol. 7, pp. 60 806–60 813, 2019.
- [8] P. Xiang, L. Wang, F. Wu, J. Cheng, and M. Zhou, “Single-image de-raining with feature-supervised generative adversarial network,” *IEEE Signal Processing Letters*, vol. 26, no. 5, pp. 650–654, 2019.
- [9] C. Wang, C. Xu, C. Wang, and D. Tao, “Perceptual adversarial networks for image-to-image transformation,” *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4066–4079, 2018.
- [10] T. Matsui and M. Ikehara, “Single-image fence removal using deep convolutional neural network,” *IEEE Access*, vol. 8, pp. 38 846–38 854, 2020.
- [11] K. Brandenburg, “MP3 and AAC explained,” in *Audio Engineering Society Conference: 17th International Conference: High-Quality Audio Coding*, Signa, 1999.
- [12] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen, “Acoustic event detection in real-life recordings,” in *Proc. of European Signal Processing Conference*, Aalborg, 2010.
- [13] D. Giannoulis, D. Stowell, E. Benetos, M. Rossignol, M. Lagrange, and M. D. Plumbley, “A database and challenge for acoustic scene classification and event detection,” in *Proc. of European Signal Processing Conference*, Marrakech, 2013.
- [14] P. Alonso-Jiménez, L. Joglar-Ongay, X. Serra, and D. Bogdanov, “Automatic detection of audio problems for quality control in digital music distribution,” in *Audio Engineering Society Convention 146*, Dublin, 2019.
- [15] S. V. Vaseghi, *Impulsive Noise: Modelling, Detection and Removal*. John Wiley & Sons, Ltd, 2008.
- [16] C. Perkins, O. Hodson, and V. Hardman, “A survey of packet loss recovery techniques for streaming audio,” *IEEE Network*, vol. 12, no. 5, pp. 40–48, 1998.
- [17] B.-K. Lee and J.-H. Chang, “Packet loss concealment based on deep neural networks for digital speech transmission,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, 2015.
- [18] A. Marafioti, N. Perraudin, N. Holighaus, and P. Majdak, “A context encoder for audio inpainting,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019.
- [19] E. Rushe and B. M. Namee, “Anomaly detection in raw audio using deep autoregressive networks,” in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brighton, 2019.
- [20] E. Marchi, F. Vesperini, S. Squartini, and B. Schuller, “Deep recurrent neural network-based autoencoders for acoustic novelty detection,” *Computational Intelligence and Neuroscience*, 2017.
- [21] R. Mühlbauer, “Automatic audio defect detection,” Bachelor’s Thesis, 2010.
- [22] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention*, Munich, 2015.
- [23] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, “Singing voice separation with deep U-Net convolutional networks,” in *18th International Society for Music Information Retrieval Conference*, Suzhou, 2017.
- [24] D. Stoller, S. Ewert, and S. Dixon, “Wave-U-Net: A multi-scale neural network for end-to-end audio source separation,” in *Proc. of the 19th International Society for Music Information Retrieval Conference*, Paris, 2018.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 2016.
- [26] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, “FMA: A dataset for music analysis,” in *18th International Society for Music Information Retrieval Conference*, Suzhou, 2017.
- [27] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference for Learning Representations*, San Diego, 2015.
- [28] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, “Essentia: an open-source library for sound and music analysis,” in *ACM International Conference on Multimedia*, Barcelona, 2013.

SEMI-SUPERVISED MUSIC TAGGING TRANSFORMER

Minz Won^{1,2} Keunwoo Choi² Xavier Serra¹

¹ Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain

² ByteDance, Mountain View, California, United States

minz.won@upf.edu

ABSTRACT

We present Music Tagging Transformer that is trained with a semi-supervised approach. The proposed model captures local acoustic characteristics in shallow convolutional layers, then temporally summarizes the sequence of the extracted features using stacked self-attention layers. Through a careful model assessment, we first show that the proposed architecture outperforms the previous state-of-the-art music tagging models that are based on convolutional neural networks under a supervised scheme.

The Music Tagging Transformer is further improved by noisy student training, a semi-supervised approach that leverages both labeled and unlabeled data combined with data augmentation. To our best knowledge, this is the first attempt to utilize the entire audio of the million song dataset.

1. INTRODUCTION

Automatic music tagging is a classification task whose objective is computational understanding of music semantics. From a given audio excerpt, a trained music tagging model predicts relevant tags (e.g., genre, mood, instrument, decade, region) based on its acoustic characteristics. The task has attracted music information retrieval (MIR) researchers due to its wide pragmatic usages in many applications. Especially, there is a strong demand from industries that have music recommendation services from large-scale music libraries. Thanks to the recent advances in deep learning, mostly convolutional neural networks (CNNs) [1], the performances of music tagging models have been significantly enhanced by leveraging large-scale data with various deep architectures [2–5]. However, there still are two limitations in the current music tagging research: i) chunk-based prediction and ii) a limited amount of labeled data for supervised learning.

Music signals are in the form of sequential data. In this sequence, regarding typical tags, some acoustic characteristics may appear locally (e.g., instruments) while some others may span over the sequence (e.g., mood, genre). This means a successful music tagging model needs to be

able to extract both local and global features. Fully convolutional network [2], one of the very early deep learning models for music tagging, was designed to capture both local and global features by increasing the size of the overall receptive fields with max-pooling. More recently, however, it is shown that training with a smaller hop size with shorter audio chunks is beneficial for music tagging [6]. This approach has been adopted in many CNN-based models [3–5], where the models are trained with short audio chunks (3 to 5 second long), densely striding max-pooling, and a global pooling layer. To predict music tags of a 3-minute song, for example, the audio is split into multiple short audio chunks, and the model makes predictions on each chunk. Then, the predictions are aggregated through majority vote or global average-/max-pooling. This means that on a track level, the current music tagging models are performing like a bag-of-features model [7] instead of modeling music representation as a sequence.

Another limitation of the current music tagging research is a limited amount of labeled data. The Modern deep learning models are data-hungry. However, manually labeling music tags is time-consuming and requires domain expertise. In pursuit of large-scale research, the million song dataset (MSD) [8], which literally includes a million songs in it, became popular in music tagging research. Among the million songs, however, only about 24% are labeled with at least one of the top-50 music tags. Most of the previous music tagging research has only utilized the labeled data while discarding 76% of the songs in the dataset. This type of setup (i.e., a small labeled dataset along with a large unlabeled dataset) is not limited to the MSD but can be found often in the real world regardless of the domain. To leverage the unlabeled data, self-supervised [9–13] and semi-supervised [14–16] learning have been actively explored in computer vision and natural language processing.

In this paper, we present Music Tagging Transformer that is trained with a semi-supervised approach. Our main contribution is three-fold: (i) through a careful model assessment, we show that our Music Tagging Transformer outperforms the previous works, (ii) we show that we can use unlabeled data to improve music tagging performances via semi-supervised learning, (iii) we provide a new split of the MSD to solve known issues of the previous one. Reproducible code is available online. ¹

¹ <https://github.com/minzwon/semi-supervised-music-tagging-transformer>



2. RELATED WORKS

2.1 Music Tagging

2.1.1 CNN-based models

The research using CNNs accelerated the improvement of music tagging models. Choi et al. proposed to adopt CNNs for music tagging [2]. Pons et al. intended to inject domain knowledge in their architecture by designing vertically and horizontally long filters so that the network can capture timbral and temporal features, respectively [3]. Lee et al. formalized music tagging as a fully end-to-end process by using a 1D CNN and raw audio inputs [4]. Won et al. proposed to use a learnable harmonic front end for a 2D CNN [5]. More recently, the authors of [6] evaluated all these CNN-based music tagging models under the same experimental environment. Two of the main conclusions of [6] are to recommend (i) using mel spectrogram inputs, and (ii) using the most granular 2D filters (i.e., 3×3 convolution) instead of manual design choices. That means, a simple 2D CNN with mel spectrogram inputs, which is prevalent and sometimes referred to as *vgg-ish* model, is still outperforming the other music tagging models.

2.1.2 Multiple instance learning

Music tagging can be seen as a multiple instance learning (MIL) [17] problem. A given music signal (a bag of multiple instances) will be labeled with a tag if a part (an instance) of the signal has a certain, relevant acoustic characteristic. The acoustic characteristic may span across the entire sequence (e.g., mood) or appear partially (e.g., instruments). There are two approaches of handling this aspect in music tagging. One is to train an instance-level model then aggregate the instance-level predictions; the other is to handle multiple instances in a single model.

The first approach has been used in many systems [3–5]. This approach is justified by our intuition – humans can predict music tags within just a few seconds. For example, people would not spend 3 minutes to determine whether a track is *rock*. In this approach, during the evaluation phase, the instance-level predictions are aggregated with a method such as majority vote, global max pooling, global average pooling, or adaptive pooling [18].

The second approach, the single model one, tackles the MIL problem in an end-to-end fashion. Fully convolutional network [2], an early music tagging model using CNN, models the entire 30-second mel spectrogram inputs. However, this model has to stick to a fixed input size and shows a relatively lower performance compared to other models that are trained with short audio chunks. To take the global structure into account while not losing local features, sequential modeling was added to the deep learning architectures. Convolutional recurrent neural network (CRNN) [19] is designed to capture local acoustic characteristics in a CNN front end and to summarize the sequence of the extracted features using an RNN back end. Similarly, another previous work [20] adopted a sequence model from the natural language processing field, the Transformer [21].

2.2 Transformers

2.2.1 Overview

Transformer [21, 22] has shown its ability in sequence modeling, establishing itself *de facto* state-of-the-art in natural language processing. The structure of Transformer is a deep stack of self-attention layers. In each layer, by self-attention mechanism, the pairwise attention scores between every time step are calculated to output another sequence which includes a better context in each time step. In detail, the self-attention score is calculated as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (1)$$

where d_k is the dimension of the *key* and Q , K , and V are *query*, *key*, and *value*, respectively. Based on the relevance of *key* and *query* (i.e., relationship between two items), how much of *value* to be passed to the next layer is decided. Since Transformer has direct paths between each time step, it does not suffer from *vanishing gradient*, which is a critical problem of RNN families when modeling long sequences. However, Transformer may suffer from its memory complexity which is quadratic to the length of the sequence.

Due to its receptive field, which is unlimited within the input sequence, Transformer has become extremely popular in sequential data modeling such as text [21], symbolic music [23], and video [24]. It even demonstrated its representation power in non-sequential data such as image [25].

2.2.2 Self-attention in music tagging

Hereinafter, we refer to the previous work [20] as CNNSA (acronym of convolutional neural network with self-attention) so that we can distinguish it from the proposed Music Tagging Transformer. The CNNSA model is the first music tagging model that uses Transformer. The model consists of a CNN front end and a Transformer back end. The CNN front end captures local (0.03 to 2.6 seconds) acoustic characteristics and the Transformer back end temporally summarizes the sequence of the features. Based on previous works, the CNNSA investigated two types of CNN front end which have two very different motivations. One is based on hand-crafted vertically and horizontally long filters [3] to capture timbral and temporal characteristics, respectively. The other is a more flexible, data-driven approach by using 1D CNN [4] with raw audio inputs. The CNNSA showed comparable results in music tagging but did not outperform other previous CNN approaches. Nevertheless, it demonstrated the Transformer’s sequence modeling ability and better temporal interpretability.

2.3 Semi-supervised Learning

With the advances of scalable hardware and training algorithms, the demand for labeled data has outpaced the progress of the size of datasets in many fields. As a solution, researchers started to develop methods that can take advantage of unlabeled data. Self-supervised [9–12] and

semi-supervised learning [14–16] aim at leveraging the abundant unlabeled data and have shown strong performances in various domains including computer vision and natural language processing.

In many self- and semi-supervised learning approaches, the models are trained to return noise-invariant predictions [11, 12, 15]. When there is an apple on the table, for example, it is always an apple even if we take a look at it from a different angle or a different distance, under different lights, or through glass. This ‘noise’ is usually realized in a form of data augmentation. In detail, with a self-supervised learning scheme, models are trained to optimize the agreement between different views of the same input [11, 12].

On the contrary, semi-supervised learning takes advantage of *both* existing labeled data and unlabeled data. One effective way of handling the data is to formalize the problem as teacher-student learning [15, 26, 27]. In teacher-student learning, a teacher model is first trained with labeled data in a supervised scheme and then, a student model is trained to mimic the teacher’s behavior by predicting pseudo-labels [28] that are generated by the teacher model. The teacher-student training has been actively explored with the purpose of domain adaptation [26], knowledge distillation [29], and knowledge expansion [15]. Especially, noisy student training [15] successfully takes advantage of the teacher-student training with the aforementioned noise invariance.

3. MODELS

3.1 Short-chunk ResNet

As a simple 2D CNN with mel spectrogram inputs outperforms other music tagging models [6], we use a short-chunk ResNet model as our baseline. The model is trained with 3.69-second short audio chunks (instance-level), then the predictions are later aggregated by averaging them in the evaluation phase. It is a seven-layer CNN and each layer comprises of 3×3 convolution with residual connection [30], batch normalization [31], rectified linear unit (ReLU) non linearity, and 2×2 max pooling. This is a variant of the prevalent *vgg-ish* model, but due to its specific characteristics (i.e., short-instance-level training and densely striding max pooling), it is referred to as short-chunk ResNet [6]. We used a publicly available implementation².

3.2 Music Tagging Transformer

The proposed Music Tagging Transformer consists of two parts: a CNN front end and a Transformer back end as in Figure 1. On a high level, our structure is similar to the previous CNNSA [20] where CNN captures local spectrotemporal features and Transformer globally summarizes the sequence of the extracted features. However, there are two main differences in the front end: the CNN architecture and the reshaping layer.

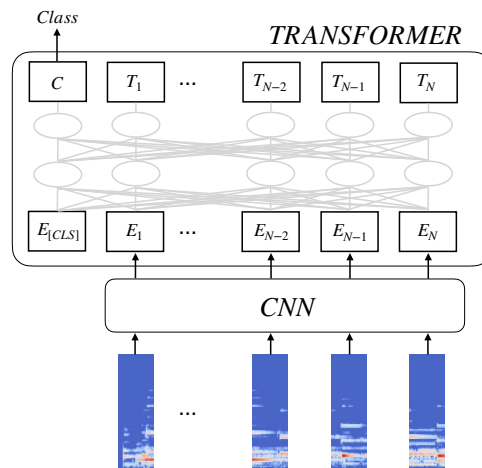


Figure 1: Proposed Music Tagging Transformer.

layer	output shape
Input	$B \times 1 \times F \times T$
Conv (3×3)	$B \times C \times F \times T$
MaxPool (2×2)	$B \times C \times F/2 \times T/2$
Conv (3×3)	$B \times C \times F/2 \times T/2$
MaxPool (2×2)	$B \times C \times F/4 \times T/4$
Conv (3×3)	$B \times C \times F/4 \times T/4$
MaxPool (2×1)	$B \times C \times F/8 \times T/4$
Reshape	$B \times (C \cdot F/8) \times T/4$
Fully-connected	$B \times C' \times T/4$

Table 1: Front end CNN of Music Tagging Transformer.

The CNNSA [20] investigated two different CNN front ends that are from two opposite motivations. One is using hand-crafted filter design on mel spectrogram inputs to leverage domain knowledge [3], the other is, with a fully data-driven spirit, one that learns from raw audio in an end-to-end fashion [4]. However, we follow a suggestion from a more recent work [6] – we use 3×3 convolution filters with residual connections [30] on mel spectrogram inputs. Table 1 outlines our 3-layered CNN front end where B is the batch size, C is the number of convolution channels, F is the number of mel bins, T is the number of frames, and C' is the number of attention channels of Transformer. This CNN front end i) helps the model to capture local representations and ii) reduces the time resolution of the input so that it is feasible to train the following back end.

At the end of the CNN, the second and the third dimensions are reshaped into a single dimension. This flattening is motivated by Vision Transformer (ViT) [25] which reshapes a 2D image patch into a one-dimensional array. As a result, the output of the CNN is a sequence of short-chunk audio features where a chunk corresponds to approximately 0.1 second. It is input to the back end Transformer. This is in contrast to the CNNSA [20] that used the frequency-axis max-pooling at the end of its front end. In other words, in Music Tagging Transformer, the attention layers are given more detailed spectral information.

Our back end Transformer architecture is nearly iden-

² <https://github.com/minzwon/sota-music-tagging-models>

tical to the previous works [20, 21] except for the number of parameters and input lengths. After a hyperparameter search, we chose 4 layers, 256 attention dimensions, and 8 attention heads. At the input stage, positional embedding [21] is applied and a special token embedding $E_{[CLS]}$ is inserted so that the Transformer can perform sequence classification (Figure 1) as a downstream task.

3.3 Noisy Student Training

To leverage unlabeled data, we investigate noisy student training [15], a successful semi-supervised learning approach. Table 2 provides an overview of noisy student training with pseudocode. First, we train a teacher model \mathcal{T} with a conventional supervised learning approach (line 1–6). Then, we train a student model \mathcal{S} with two types of losses. The first loss, l_1 , is coming from the typical supervised approach with labeled data (line 10–11) as done for the teacher model. The other loss, l_2 , is from unlabeled inputs and the corresponding pseudo-labels provided by the teacher model (line 12–15). In order to make the student model perform beyond mimicking the teacher model, data augmentation is applied (line 13). Both hard and soft labels can be used for the pseudo-labels [15] and we use soft labels in our work. If the training is successful, the trained student model would outperform the teacher model. Furthermore, the whole training process can be done iteratively by using the student as a new teacher model and training another student model to obtain an even better performing model. For a stronger teacher model, we used data augmentation in our supervised learning pipeline as well (line 1–6 and 10–11). As a result, the only pipeline without data augmentation is pseudo-label generation (line 12).

The size of the student model can be identical or larger than the teacher model. In this case, one can interpret the training process as *knowledge expansion* [15], meaning the knowledge in the teacher model is upgraded in the student model. One can also design the student model to be smaller than the teacher model, making the process *knowledge distillation* [32]. Knowledge expansion and knowledge distillation are complementary; depending on the use-case, one could pursue either performance or efficiency. We investigate both directions in this paper.

3.4 Data Augmentation

Data augmentation is a key to success in noisy student training. In our experiments, we take advantage of *Audio Augmentations* library [33] which is easily integrated to PyTorch data pipeline. The applied data augmentation methods are as follows:

- **Polarity inversion.**
- **Additive noise** by $k_{snr} \in \{0.3, 0.5\}$.
- **Random gain** by $\mathcal{A} \in \{-20, -1\}$ dB.
- **High-pass filter** by $f_H \in \{2200, 4000\}$ Hz.
- **Low-pass filter** by $f_L \in \{200, 1200\}$ Hz.

Noisy Student Training

Input labeled data X , labels Y , unlabeled data Z

Models teacher model \mathcal{T} , student model \mathcal{S}

Functions loss function \mathcal{L} , data augmentation \mathcal{A} ,
back propagation \mathcal{B}

Train

```

1  for  $x \in X, y \in Y$ 
2  do
3     $p \leftarrow \mathcal{T}(x)$  // predict
4     $l \leftarrow \mathcal{L}(p, y)$  // get loss
5     $\mathcal{T} \leftarrow \mathcal{B}(\mathcal{T}, l)$  // update teacher model
6  end do
7  end for
8  for  $x \in X, y \in Y, z \in Z$ 
9  do
10    $p_1 \leftarrow \mathcal{S}(x)$  // predict
11    $l_1 \leftarrow \mathcal{L}(p_1, y)$  // get supervised loss
12    $\psi \leftarrow \mathcal{T}(z)$  // generate pseudo-label
13    $\hat{z} \leftarrow \mathcal{A}(z)$  // data augmentation
14    $p_2 \leftarrow \mathcal{S}(\hat{z})$  // predict
15    $l_2 \leftarrow \mathcal{L}(p_2, \psi)$  // get semi-supervised loss
16    $\mathcal{S} \leftarrow \mathcal{B}(\mathcal{S}, l_1 + l_2)$  // update student model
17 end do
18 end for

```

Table 2: Pseudocode of noisy student training.

- **Delay** by $t \in \{200, 500\}$ ms.
- **Pitch shift** by $n \in \{-7, 7\}$ semitones.
- **Reverb** by room size $s \in \{0, 100\}$.

Each augmentation method is activated independently with a probability $p \in \{0.3, 0.7\}$.

4. DATASET

We use the million song dataset (MSD) [8] which consists of one million songs with audio features and metadata. Most of the previous works [2–6] relied on the Last.fm tags, a set of crowdsourced music tags, as ground truth. A popular approach is to take the most frequent 50 tags and select tracks that have at least one of the tags. This results in 242k songs, which are split into train, validation, and test sets.³ Hereinafter, we refer to this as a conventional (MSD) split. Since this split has been widely used, we use it to benchmark the proposed Music Tagging Transformer against the previous works.

We also suggest a new split to alleviate some known problems of the conventional split. There are two problems – First, since the MSD music tags are collected from users, some of them are very noisy, and that may lead to noisy (and incorrect) evaluation [34]. Second, a strict split of music items requires taking the artist information into consideration since often, songs and labels from the same

³ https://github.com/keunwoochoi/MSD_split_for_tagging

artist heavily resemble each other. However, the conventional split was done without such consideration, having caused unintended information leakage between the training and evaluation sets. Ultimately, this would cause an overly optimistic evaluation. As a solution, we use manually cleaned data from a previous work [35] and take the top 50 tags. We also propose a new split of MSD that does not share any artist among training/validation/test sets and is extended to more tracks. We name this ‘CALS split’ (cleaned and artist-level stratified split). CALS split consists of 233k labeled tracks and 516k unlabeled tracks. To the best of our knowledge, this is the first attempt to utilize the entire MSD audio, although we have to discard the rest 250k tracks to avoid information leakage by shared artists.

When noisy student training is used, it is common to have an unlabeled set that is significantly bigger than the labeled set. For example, in computer vision, 81 million unlabeled items were used along with 1.2 million labeled items [15], making the ratio of the semi-supervised set to be 67.5 (81/1.2). However, with 233k labeled items and 516k unlabeled items, our ratio is only around 2.3. This might be a factor that limits us from fully exploring the potential advantage of semi-supervised learning as we will discuss in Section 5.

5. EXPERIMENT

We use the MSD that was introduced in detail in Section 4. All the audio signals are pre-processed to 22,050 Hz sample rate and converted to short-time Fourier transform representations with a 1024-point FFT and 50%-overlapping Hann window. Finally, we convert them to log mel spectrograms with 128 mel bins.

All models that are used or introduced in this paper are optimized using Adam [36] with a learning rate of 0.0001. The best model is selected based on the binary cross entropy loss of the validation set and early stopping is applied when the validation loss does not improve for 20 epochs.

Music tagging models are typically evaluated with Area Under Receiver Operating Characteristic Curve (ROC-AUC). However, it is known that ROC-AUC may report overly optimistic results with highly skewed data [37]. Therefore, as our main evaluation metrics, we use not only ROC-AUC but also Area Under Precision-Recall Curve (PR-AUC).

5.1 Performance with the conventional split

Table 3 summarizes the performance of previous systems and the proposed model using the conventional split. The ROC-AUC and PR-AUC of the many previous models have been under 0.89 and 0.33, respectively. Our model, Music Tagging Transformer, outperforms the previous state-of-the-art models, harmonic CNN and short-chunk ResNet. The improvement, especially on PR-AUC, is non-trivial and even larger with data augmentation.

The front end of our Music Tagging Transformer takes a sequence of chunks, where each of which represents a very short duration of the signal (≈ 0.1 second) [6]. Be-

Models	ROC-AUC	PR-AUC
FCN [2]	0.8742	0.2963
Musicnn [3]	0.8788	0.3036
Sample-level [4]	0.8789	0.2959
Sample-level+SE [38]	0.8838	0.3109
CRNN [19]	0.8460	0.2330
CNNSA [20]	0.8810	0.3103
Harmonic CNN [5]	0.8898	0.3298
Short-chunk CNN [6]	0.8883	0.3251
Short-chunk ResNet [6]	0.8898	0.3280
Transformer (proposed) [§]	0.8916	0.3358
Transformer (proposed) + DA [†]	0.8972	0.3479

Table 3: Performance comparison using the conventional MSD split for top-50 music tagging. The § and † marks mean they are based on the identical model architecture and training strategy; compared to the same, marked models in Table 4, only the dataset split is different.

Models	#param	ROC-AUC	PR-AUC
ResNet [6]	13.5m	0.9098	0.3525
ResNet+DA	13.5m	0.9141	0.3705
ResNet+DA+KE	13.5m	0.9165	0.3728
ResNet+DA+KD	3.4m	0.9171	0.3742
Transformer [§]	4.6m	0.9188	0.3775
Transformer+DA [†]	4.6m	0.9191	0.3845
Transformer+DA+KE	4.6m	0.9204	0.3839
Transformer+DA+KD	0.5m	0.9217	0.3889

Table 4: Performance comparison using the CALS MSD split for music tagging.

cause 0.1 second would be too short to represent musical characteristics alone, we interpret that the experimental results would mean our Transformer back end plays a role of sequential feature extractor beyond simple bag-of-feature aggregation. This may be an important aspect of the proposed model since sequential modeling is what the self-attention mechanism is the best suit.

The data augmentation we adopted contributes to improvements of 0.0056 ROC-AUC and 0.0119 PR-AUC. These are bigger than many of the improvements we have seen between different architecture choices. This emphasizes that data augmentation should be considered when developing a music tagging model.

5.2 Performance with the CALS split

For a deeper and more accurate analysis of the proposed models and methods, we use the proposed CALS split and run experiments with various configurations. Table 4 presents the experimental results of short-chunk ResNets (baseline architecture) and Music Tagging Transformers. For both of the models, it also summarizes the results of supervised models (the baseline among training methods), models with data augmentation (DA), models with

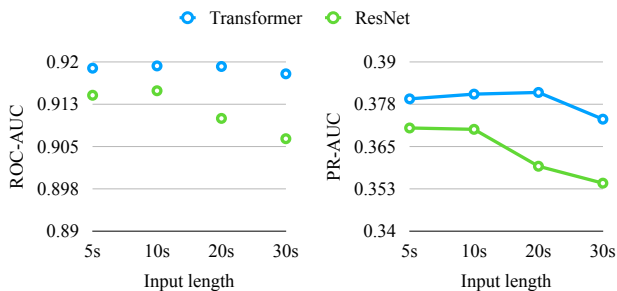


Figure 2: Performance with different input lengths.

width	depth	ROC-AUC	PR-AUC
32	4	0.9118	0.3528
64	4	0.9178	0.3754
128	4	0.9194	0.3776
256	4	0.9191	0.3845
512	4	0.9177	0.3788
768	4	0.9162	0.3707
1024	4	0.9174	0.3736
256	1	0.9180	0.3736
256	2	0.9193	0.3805
256	4	0.9199	0.3814
256	8	0.9181	0.3826
256	12	0.9165	0.3780
256	16	0.9169	0.3785

Table 5: The performance of Music Tagging Transformer with varying width and depth of the attention layers.

DA and knowledge expansion (KE), and models with DA and knowledge distillation (KD). Note that the two bottom rows of Table 3 correspond to the 5th and 6th rows of Table 4 as marked with § and †.

For both short-chunk ResNet [6] and the Music Tagging Transformer, we observe constant improvements when data augmentation and noisy student training (knowledge expansion) are applied accumulatively. This shows that for both of the architectures, the size of the dataset is a factor that limits the performance of the models.

In Section 4, we mentioned that our ratio of the semi-supervised set is relatively small. There are two observations that may be related to it. First, unlike a previous work in computer vision [15], we could not observe any performance gain by iterating the noisy student training (i.e., repeating to use a student model as the next teacher model). Second, interestingly, the student model with smaller parameters (models with DA and KD) showed better performance than larger models (models with DA and KE). This would be explained more clearly if the models are trained with a significantly richer dataset, one that is bigger and/or has more diverse data. Unfortunately, we could not run such an experiment due to the lack of a suitable dataset.

5.3 More Hyperparameter Search

In this section, we present the experiment results with various model configurations. First, we trained our Music Tag-

ging Transformer and short-chunk ResNet with varying input lengths to assess our proposed model’s ability to handle long sequences. As shown in Figure 2, on both of the metrics, short-chunk ResNet shows a noticeable performance degradation as the audio input gets longer. This shows that the global max pooling in the short chunk ResNet is not perfectly suitable for a long signal. Meanwhile, the Music Tagging Transformer shows consistent performances in general. An exception is when the input is 30-second long. We suspect the performance drop of the Music Tagging Transformer happens because the model cannot take advantage of random cropping data augmentation effect since the 30-second is the full length of the MSD previews.

Second, we investigate different Transformer parameters to figure out the best performing setup. As summarized in Table 5, Transformer achieved the best performance with attention channels (width) at 128 and 256, and their depth of 4 and 8 layers. However, these optimal parameters are dataset-dependent; as generally observed, a larger network structure would perform better if a larger amount of training data is provided.

6. CONCLUSION

In this paper, we proposed a new architecture, Music Tagging Transformer, and improved its tagging performance with a semi-supervised scheme: noisy student training. Experimental results showed that the proposed architecture outperforms the previous state-of-the-art models in supervised music tagging using the MSD [8]. The results also indicate that the tagging models can be further enhanced using noisy student training – with either knowledge expansion and knowledge distillation. We also provided an analysis result that shows Music Tagging Transformer can handle long audio inputs better than the previous CNN architectures do.

In future work, our Transformer can be further utilized in various MIR tasks. Since Transformer can perform both sequence-level and token-level classification, it can be used in not only music tagging but also tasks such as beat detection and melody extraction. Finally, by combining the multiple MIR tasks in a multi-task learning scheme, Transformer can be trained as a general purpose music representation learning model.

Another important direction to explore is self-supervised learning. Our Music Tagging Transformer can be pre-trained in a self-supervised scheme such as masked embedding prediction [21] or contrastive learning [11–13]. Finally, the pre-trained model can be optimized together with semi-supervised learning to further improve the performance [16].

7. ACKNOWLEDGEMENT

This work was funded by the predoctoral grant MDM-2015-0502-17-2 from the Spanish Ministry of Economy and Competitiveness linked to the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502).

8. REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems (NIPS)*, 2012.
- [2] K. Choi, G. Fazekas, and M. Sandler, “Automatic tagging using deep convolutional neural networks,” in *In Proc. of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, 2016.
- [3] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, “End-to-end learning for music audio tagging at scale,” in *In Proc. of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.
- [4] J. Lee, J. Park, K. L. Kim, and J. Nam, “Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms,” *In Proc. of the 14th Sound and music computing (SMC)*, 2017.
- [5] M. Won, S. Chun, , O. Nieto, and X. Serra, “Data-driven harmonic filters for audio representation learning,” *In Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [6] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, “Evaluation of cnn-based automatic music tagging models,” *In Proc. of Sound and Music Computing (SMC)*, 2020.
- [7] W. Brendel and M. Bethge, “Approximating cnns with bag-of-local-features models works surprisingly well on imagenet,” *International Conference on Learning Representations (ICLR)*, 2019.
- [8] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, “The million song dataset,” *In Proc. of the 12th International Conference on Music Information Retrieval (ISMIR)*.
- [9] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.
- [10] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International Conference on Machine Learning (ICML)*, 2020.
- [12] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar *et al.*, “Bootstrap your own latent: A new approach to self-supervised learning,” *arXiv preprint arXiv:2006.07733*, 2020.
- [13] J. Spijkervet and J. A. Burgoyne, “Contrastive learning of musical representations,” *In Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2021.
- [14] O. Chapelle, B. Scholkopf, and A. Zien, “Semi-supervised learning,” *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.
- [15] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, “Self-training with noisy student improves imagenet classification,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [16] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, “Big self-supervised models are strong semi-supervised learners,” in *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [17] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, “Solving the multiple instance problem with axis-parallel rectangles,” *Artificial intelligence*, vol. 89, no. 1-2, pp. 31–71, 1997.
- [18] B. McFee, J. Salamon, and J. P. Bello, “Adaptive pooling operators for weakly labeled sound event detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.
- [19] K. Choi, G. Fazekas, M. Sandler, and K. Cho, “Convolutional recurrent neural networks for music classification,” in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2392–2396.
- [20] M. Won, S. Chun, and X. Serra, “Toward interpretable music tagging with self-attention,” *arXiv preprint arXiv:1906.04972*, 2019.
- [21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.
- [22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *Conference on Neural Information Processing Systems (NIPS)*, 2017.
- [23] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, “Music transformer,” *International Conference on Learning Representations (ICLR)*, 2019.
- [24] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, “Vivit: A video vision transformer,” *arXiv preprint arXiv:2103.15691*, 2021.

- [25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *International Conference on Learning Representations (ICLR)*, 2021.
- [26] J. Li, M. L. Seltzer, X. Wang, R. Zhao, and Y. Gong, “Large-scale domain adaptation via teacher-student learning,” *arXiv preprint arXiv:1708.05466*, 2017.
- [27] S. Kum, J.-H. Lin, L. Su, and J. Nam, “Semi-supervised learning using teacher-student models for vocal melody extraction,” *arXiv preprint arXiv:2008.06358*, 2020.
- [28] D.-H. Lee *et al.*, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks,” in *Workshop on challenges in representation learning, ICML*, 2013.
- [29] Y. Kim and A. M. Rush, “Sequence-level knowledge distillation,” *Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.
- [31] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International conference on machine learning (ICML)*, 2015.
- [32] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *Advances in neural information processing systems (NIPS), Deep learning workshop*, 2015.
- [33] J. Spijkervet, “Spijkervet/torchaudio-augmentations,” 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5042440>
- [34] K. Choi, G. Fazekas, K. Cho, and M. Sandler, “The effects of noisy labels on deep convolutional neural networks for music tagging,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2018.
- [35] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, “Multimodal metric learning for tag-based music retrieval,” in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [36] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations (ICLR)*, 2015.
- [37] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *Proc. of international conference on Machine learning (ICML)*, 2006.
- [38] T. Kim, J. Lee, and J. Nam, “Sample-level cnn architectures for music auto-tagging using raw waveforms,” in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

EMOTION EMBEDDING SPACES FOR MATCHING MUSIC TO STORIES

Minz Won¹ Justin Salamon² Nicholas J. Bryan² Gautham J. Mysore² Xavier Serra¹

¹ Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain

² Adobe Research, San Francisco, California, United States

minz.won@upf.edu

ABSTRACT

Content creators often use music to enhance their stories, as it can be a powerful tool to convey emotion. In this paper, our goal is to help creators find music to match the emotion of their story. We focus on text-based stories that can be auralized (e.g., books), use multiple sentences as input queries, and automatically retrieve matching music. We formalize this task as a cross-modal text-to-music retrieval problem. Both the music and text domains have existing datasets with emotion labels, but mismatched emotion vocabularies prevent us from using mood or emotion annotations directly for matching. To address this challenge, we propose and investigate several emotion embedding spaces, both manually defined (e.g., valence/arousal) and data-driven (e.g., Word2Vec and metric learning) to bridge this gap. Our experiments show that by leveraging these embedding spaces, we are able to successfully bridge the gap between modalities to facilitate cross modal retrieval. We show that our method can leverage the well established valence-arousal space, but that it can also achieve our goal via data-driven embedding spaces. By leveraging data-driven embeddings, our approach has the potential of being generalized to other retrieval tasks that require broader or completely different vocabularies.

1. INTRODUCTION

Content creators, both amateur and professional alike, often use music to enhance their storytelling due to its powerful ability to elicit emotion¹. For example, when dissonant music is added to a horror movie, it can amplify the scary mood of the story line. Similarly, cheerful music can emphasize the excited mood in a scene of a birthday party. Matching text and music to create a narrative, typically requires tediously browsing large-scale music collections, significant experience, and musical expertise. In this

¹ We use the terms *emotion* and *mood* interchangeably following previous work [1].

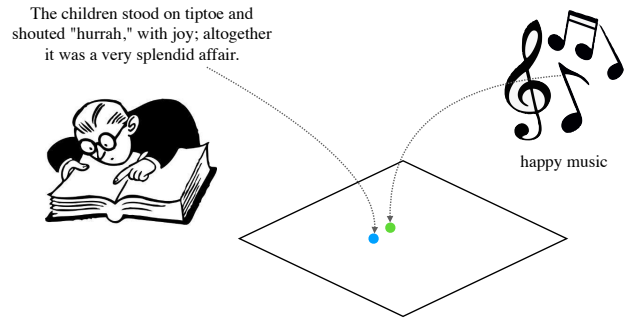


Figure 1: Cross-modal text-to-music retrieval using an aligned, multimodal embedding space.

paper, we therefore address the problem of automatically matching music to text as shown in Figure 1.

We formalize this task as a cross-modal retrieval problem [2] and focus on matching long-form text (multiple sentences, paragraphs) to music. For queried sentences like books and scripts, we seek to retrieve matching music for applications such as podcasts, audio books, movies, and film. To facilitate cross-modal retrieval, a common approach is to first perform feature extraction to convert each data modality into an embedding space. Then, the different embedding spaces must be matched to bridge the *modality gap* by somehow aligning their different distributions [2]. Once aligned, (fast) nearest neighbor search can be used for retrieval.

Various methods have been proposed for cross-modal feature extraction and alignment. For example, canonical correlation analysis has been used to bridge the modality gap [3] as well as modern deep learning techniques that learn common representation spaces [4, 5]. Such methods can be categorized into four groups: unsupervised, pairwise-based, rank-based, and supervised methods [6]. Among these, supervised methods are the most straightforward and in theory can take advantage of existing labeled datasets (e.g., labels of *happy*, *sad*) and themes (e.g., *party*, *wedding* with corresponding text and music). Difficulties, however, immediately arise because of mismatched dataset taxonomies (vocabularies) per modality, making it challenging to use standard techniques directly.

Therefore, in this work we focus on the task of emotion-based text- (e.g. sentences, paragraphs) to-music retrieval, and investigate how we can best perform cross-modal retrieval with heterogeneous dataset taxonomies. To the best of our knowledge, this problem has not been previously ad-



dressed and could be beneficial to media content creation applications. We propose six different deep learning strategies to extract relevant features and bridge the modality gap between text and music including (1) classification (2) multi-head classification (3) valence-arousal regression (4) Word2Vec regression (5) two-branch metric learning and (6) three-branch metric learning. We then evaluate each approach on multiple text and music datasets, report objective results via precision at five and mean reciprocal rank, and conclude with qualitative analysis and discussion. Our results show that our valence-arousal-based method is a powerful baseline for emotion-based cross-modal retrieval, but that our three-branch metric-learning approach is comparable, more general, and does not require manually engineered valence and arousal mappings.

2. RELATED WORK

2.1 Text Emotion Classification

Text emotion classification methods or the task of predicting emotion from text can be divided into three categories: lexicon-based models, traditional machine learning models, and deep learning models. Lexicon-based models take advantage of pre-defined emotion lexicons, such as NRC EmoLex [7] and WordNet-Affect [8] to match keywords. Traditional machine learning approaches recognize emotions using algorithms such as support vector machine (SVM) [9] and Naive Bayes [10]. Finally, deep learning models use deep sequence models such as gated recurrent unit (GRU) [11], bidirectional long-short term memory (BiLSTM) [12], and Transformers [13]. Most recently, Transformer models [14–16] have become quite prevalent. Such models take advantage of transfer learning, are commonly pre-trained to learn language representation with large datasets, and then applied to various downstream tasks including question and answer systems as well as emotion recognition [13].

2.2 Music Emotion Classification

Music emotion classification or the task of predicting emotion from music audio is commonly divided into conventional feature extraction and prediction approaches [17–19], and end-to-end deep learning approaches [20, 21]. Deep learning approaches have become most prevalent and commonly frame emotion recognition as a multi-class or multi-label auto-tagging classification problem [22–26]. Recently, multiple music tagging models were evaluated in a homogeneous evaluation pipeline [27] and found three design recommendations for automatic music tagging models: (1) use mel-spectrogram inputs, (2) use 3×3 convolutional filters, and (3) use short-chunk audio inputs with small hop sizes and max-pooling. Based on this, a model using mel-spectrogram inputs and convolutional neural networks with focal loss [28] won the MediaEval 2020 Emotion-and-Theme-Recognition-in-Music-Task² [29].

² <https://multimediaeval.github.io/2020-Emotion-and-Theme-Recognition-in-Music-Task>

Tag	GoogleNews	Domain-specific [35]
Chill	chilly, cold, chilled, chills, shivers, shiver, warm, frigid, frosty, balmy	chill_out , relax, chilled, kick_back, relaxing, chill-out , chilled_out , downtempo , down_tempo , unwind

Table 1: Nearest words in GoogleNews and domain-specific word embeddings [35]. Music-related words are highlighted in bold.

2.3 Valence-arousal Regression & Word Embeddings

Beyond classification, previous works [30, 31] suggest that regression approaches can outperform classification approaches in music emotion recognition. Here, researchers use the well-known valence-arousal emotion space [32, 33] where valence represents positive-to-negative emotions, and arousal indicates the intensity of the emotions. These annotations can be collected by human annotators directly [30] or by mapping existing mood labels into the valence-arousal space using pre-defined lexicons [21, 34].

As an alternative to using the manually annotated valence-arousal space, we can obtain tag (mood) embeddings in a more data-driven fashion. Pre-trained word embeddings, such as Word2Vec [36] and GloVe [37], represent words as vectors by learning word associations from a large corpus. These embedding spaces use the cosine similarity as a measure of semantic similarity. Recent works [35, 38] show the suitability of pre-trained word embedding in music retrieval and that the embedding can include more music related context by training it with music related documents [35, 39]—see Table 1.

2.4 Cross-modal Retrieval

Instead of targeting a pre-defined embedding space, multimodal metric learning models aim at learning a shared embedding space in which semantically similar items are close together while dissimilar items are far apart in the embedding space. Unsupervised approaches leverage co-occurrence information. For example, when we collect user-created video from the web, the video and audio streams are synchronized, and this correspondence can be exploited for representation learning [40, 41]. On the other hand, supervised methods learn discriminative representations by exploiting annotated labels. Here, data from different modalities are used to train models such that data points with the same label should be close while data with different labels should be far apart. Metric learning is also used for bridging the modality gap between text and audio, such as keyword spotting [42], text-based audio retrieval [43, 44], and tag-based music retrieval [35, 38] in both supervised and unsupervised ways.

Two branch metric learning [45] is one of the most prevalent architectures for cross-modal retrieval. It consists of two branches where each branch extracts features from each modality and maps them into a shared embedding space. When optimized with a conventional triplet loss (e.g. anchor text, positive song, negative song), however, the model loses neighborhood structure within

modalities. To alleviate this issue, previous work [46] added structure-preserving constraints by using additional triplet losses within modalities (e.g., anchor text, positive text, negative text).

3. MODELS

Cross-modal retrieval comprises two parts: feature extraction and bridging the modality gap. Our text and music embeddings, E_{text} and E_{music} respectively, are defined as follows:

$$\begin{aligned} E_{text} &= M(P_{text}(x)) \\ E_{music} &= M(P_{music}(x)) \end{aligned} \quad (1)$$

where P is a pre-trained model to extract features from each modality and M is a multilayer perceptron (MLP) to map them to a multimodal embedding space.

3.1 Pre-trained Models for Feature Extraction

In our work, we leverage the DistilBERT [16] transformer model for text analysis, which is a compact variant of the popular BERT transformer model [14, 16]. We use a pre-trained model from the Huggingface library [47].

For the music representation model P_{music} , we use a CNN with residual connections that are trained with mel-spectrograms (ResNet) [27]. Due to its simplicity and high performance, it is a broadly used architecture not only in music but also in general audio representation learning. Our ResNet consists of 7 convolutional layers with 3×3 filters followed by 2×2 max-pooling. The model is pre-trained with the MagnaTagATune dataset³ [48]. Both pre-trained models are updated during the training process so that they can adapt to the data.

3.2 Embedding Models to Bridge the Modality Gap

3.2.1 Classification

As a starting point, we train two separate mood classification models for text and music (Figure 2-(a)). Then the model returns mood predictions and their likelihood with softmax. From the predicted text mood, songs are re-ranked based on their likelihoods of the text mood. However, this classification approach has an inherent limitation- the model cannot bridge the modalities when they have different mood taxonomies.

3.2.2 Multi-head Classification with Shared Weights

Multi-head model is similar to the classification model but it shares a 3-layered MLP for multimodal fusion in it (Figure 2-(b)). Since the model shares the weights across different modalities, it can predict the mood in different taxonomies by switching the classification head. We included this model to see if the shared MLP can generalize across modalities.

3.2.3 Regression

Following previous work [21], we reformulate the classification task as a regression problem. By using NRC VAD Lexicon [34], emotion labels can be mapped to the valence-arousal space. However, this mapping process is hand-crafted and also they cannot handle bi-grams or tri-grams since the lexicon was created in a word-level. In addition to leveraging the valence-arousal space, we also experiment with a Word2Vec [36] embedding which was pre-trained with music related text [35]. This data-driven space supports a larger vocabulary, including bi-grams and tri-grams, and is thus more flexible.

Regression models are trained separately for each modality (Figure 2-(a)). Then the nearest items are retrieved based on their distance in the embedding space. Note that, distance metrics are Euclidean distance for the valence-arousal space, and cosine distance for the Word2Vec space. However, regression is a one-way optimization, i.e., optimizing text or mood into the pre-defined word embedding space. In this case, neighborhood structure within each modality can be ignored. For example, music with *angry* and *exciting* can share similar acoustic characteristics. However, if two words are far apart in Word2Vec space, this similarity cannot be considered by regression. This obstacle motivates us to learn a shared embedding space in a data-driven fashion using metric learning.

3.2.4 Metric Learning

Finally, we explore metric learning, which is a fully data-driven approach that solves the cross modal text-to-music retrieval in an end-to-end manner. Metric learning is optimized to minimize a triplet loss \mathcal{T} :

$$\mathcal{T}(E_a, E_p, E_n) = [D(E_a, E_p) - D(E_a, E_n) + \delta]_+ \quad (2)$$

where D is a cosine distance function, δ is a margin, and E_a, E_p, E_n are embedding of anchor, positive, and negative examples, respectively. $[\cdot]_+$ is rectified linear unit. Following conventional metric learning models for cross-modal retrieval, we implement a two-branch metric learning model [45] (Figure 2-(c)) that optimizes the loss function L ,

$$L = \mathcal{T}(E_{text}^a, E_{music}^p, E_{music}^n). \quad (3)$$

However, with the triplet function, neighborhood structure or data distribution within modalities can be lost. Structure-preserving constraints [46] can alleviate the issue but our problem is different from the case, since we have different taxonomies across the modality which includes many non-overlapped moods.

To take advantage of different mood distribution of different modalities, we investigate metric learning model with three branches (Figure 2-(d)) that results in three triplet loss functions. Each loss function is designed to optimize tag-to-text, tag-to-music, and text-to-music triplet losses as following:

³ We use the pre-trained model from this open source repository: <https://github.com/minzwon/sota-music-tagging-models>

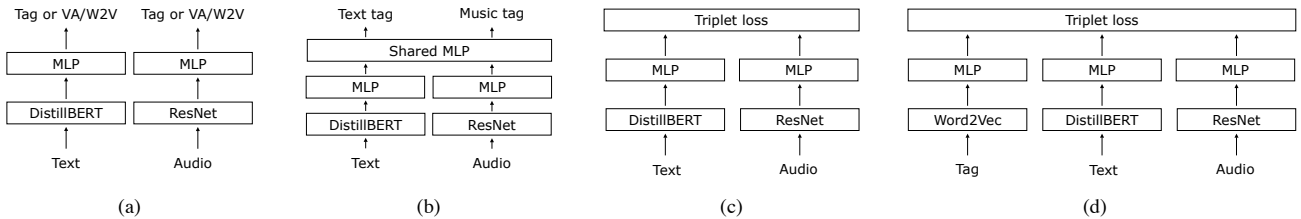


Figure 2: Model architectures. (a) Classification and regression models (b) Multi-head classification model with shared weights (c) Two-branch metric learning (d) Three-branch metric learning.

$$\begin{aligned}
 L_{text} &= \mathcal{T}(E_{tag}^a, E_{text}^p, E_{text}^n), \\
 L_{music} &= \mathcal{T}(E_{tag}^a, E_{music}^p, E_{music}^n), \\
 L_{cross} &= \mathcal{T}(E_{text}^a, E_{music}^p, E_{music}^n).
 \end{aligned}
 \tag{4}$$

The model learns a shared mood space between Word2Vec embedding and text embedding with a loss L_{text} , and a shared mood space between Word2Vec embedding and music embedding with a loss L_{music} . Finally, they are bridged together with a cross-modal triplet loss L_{cross} . We refer to this model as three-branch metric learning.

Since text and music have different vocabularies in our scenario, for both two-branch and three-branch metric learning, we regard the nearest tags in pre-trained Word2Vec space as positive pairs in cross-modal triplet sampling (Table 2). We used distance-weighted sampling [49] for more efficient negative mining following previous work [35].

4. EXPERIMENTAL DESIGN

4.1 Text Datasets

Alm’s affect dataset [50] includes 1,383 sentences collected from books written by three different authors: B. Potter, H.C. Andersen, and the Brothers Grimm. 1,207 sentences in the dataset are annotated with one representative emotion among five: *angry*, *fearful*, *happy*, *sad*, and *surprised*. To avoid unintended information leakage, we decided to split data in an author-level. 1,040 sentences by the Brothers Grimm and H.C. Andersen were used for training and 167 sentences by B. Potter were used for validation and test.

ISEAR dataset [51] is a corpus with 7,666 sentences that are categorized into one of seven emotion: *anger*, *disgust*, *fear*, *joy*, *sadness*, *shame*, and *guilt*. Each sentence describes certain antecedents and those are associated with according reactions (emotions). We split the dataset in a stratified manner with ratio of 70% train, 15% validation, and 15% test set.

4.2 Music Dataset

There are multiple datasets for music emotion recognition such as the Million Song Dataset (MSD) subset [52], the MTG-Jamendo mood subset [53], and the AudioSet mood subset [54]. Before we choose our dataset, we run classification experiments for each subset. AudioSet subset returned the highest accuracy, which means the labeled emo-

Original	VA	W2V	Manual
anger	angry	angry	angry
fearful	sad	scary	scary
happy	happy	happy	exciting, funny, happy
sad	sad	sad	sad
surprised	exciting	happy	exciting
anger	angry	angry	angry
disgust	angry	angry	angry, scary
fear	angry	angry	scary
guilt	sad	angry	angry, sad
joy	exciting	tender	exciting, funny, happy
sadness	sad	tender	sad
shame	angry	sad	angry, sad

Table 2: Similar moods from Alm’s dataset (upper) and ISEAR dataset (lower). Original is from text mood taxonomy and mapped tags are from music dataset.

tions are predictable with our ResNet model. One possible reason for this result is that unlike other datasets, emotion labels of AudioSet subset are exclusive, having a single emotion label per song. This is also beneficial since we can map each song directly to the valence-arousal space or word embedding space using emotion lexicons or Word2Vec model, respectively. Otherwise, to handle multiple tags, we need to average their embedding vectors as previous researchers did [21]. For these simplicity and reliability reasons, we use AudioSet mood subset.

AudioSet [54] mood subset consists of 16,995 music clips collected from YouTube and each audio clip is 10-second long. The dataset is categorized into 7 mood categories: *happy*, *funny*, *sad*, *tender*, *exciting*, *angry*, and *scary*. The dataset is provided with a training set of 16,104 clips and an evaluation set of 540 clips.

4.3 Evaluation

We use two evaluation metrics: Precision at 5 (P@5) and Mean Reciprocal Rank (MRR). However, since our text and audio datasets use different taxonomies, we need a mapping between the different vocabularies in order to compute the metrics directly. Thus, we map the text emotion taxonomy to the music emotion taxonomy — see Table 2. We introduce three possible mappings: (1) mapping based on the Euclidean distance between emotion labels in the valence-arousal space (VA), (2) the cosine distance between emotion labels in Word2Vec space (W2V), or (3) direct manual mapping of emotion labels. Given these mappings, we compute P@5 and MRR. Another challenge is the label distribution in our datasets, which is unbalanced.

Methods	Alm’s dataset						ISEAR dataset					
	VA		W2V		Manual		VA		W2V		Manual	
	P@5	MRR	P@5	MRR	P@5	MRR	P@5	MRR	P@5	MRR	P@5	MRR
Classification	0.2161	0.2436	0.1861	0.2157	0.2161	0.2436	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Multi-head Classification	0.2819	0.4181	0.1271	0.1381	0.3446	0.5304	0.3440	0.5084	0.3325	0.3625	0.3551	0.4803
V-A Regression	0.4325	0.6282	0.4125	0.5749	0.6100	0.7398	0.3018	0.5247	0.1866	0.3709	0.6218	0.7075
W2V Regression	0.3960	0.5010	0.4613	0.5591	0.5413	0.6363	0.3008	0.3829	0.4164	0.4908	0.5527	0.7668
Metric Learning (2-branch)	0.3399	0.3778	0.4897	0.5239	0.5374	0.5579	0.2695	0.3287	0.3951	0.4336	0.4438	0.6175
Metric Learning (3-branch)	0.3574	0.4348	0.5095	0.5863	0.5156	0.5880	0.2591	0.3445	0.4317	0.4953	0.6019	0.6675

Table 3: Retrieval scores

This can lead to over-optimistic results if the model performs well on the majority class, even if it performs very poorly on less common labels in the test dataset. To alleviate this problem, we compute the macro-P@5 and macro-MRR, i.e., we compute the metrics per class (emotion label) then average the per-class results. Henceforth we will use P@5 and MRR to denote *macro* P@5 and MRR, respectively.

Regression models are optimized to reduce mean squared error and metric learning models are optimized with the triplet losses detailed in Section 3.2.4. We use the Adam optimizer with learning rate 0.0001 for all models. Audio inputs are resampled into 16 kHz and then converted to 128-bin mel-spectrograms via a 512-point FFT with 50% frame overlap. Implementation details are available online ⁴.

5. RESULTS

5.1 Quantitative Results

The retrieval results for the different proposed models, using our three different proposed vocabulary mappings (VA, W2V, Manual), for our two text datasets, are presented in Table 3. First, we see that the classification model fails in cross-modal retrieval. Since there are only two emotions in common between Alm’s dataset and AudioSet (i.e., *happy* and *sad*), text inputs with other emotions will not have any retrieval result. Furthermore, there’s no common emotion between ISEAR dataset and AudioSet, hence P@5 and MRR are zero in this case. Classification models can be powerful when there are exactly identical or partially overlapped vocabularies, but since it is less likely in real-world data, classification approach is less desirable for cross-modal retrieval.

The multi-head classification model also performs worse than other regression and metric learning models. Some metrics look optimistic but when we check the confusion matrix of the multi-head classification model, it constantly predicts one or two specific emotions (e.g., predict *angry* for any type of input) no matter what the input is. This means the shared MLP cannot generalize across different modality heads.

The regression model using valence-arousal consistently shows the best metrics as already proven in previous single-modality emotion recognition works [30,31]. Since

the space is carefully designed and the tag-to-space mapping process has been done manually [34], the valence-arousal regression suits our cross-modal retrieval task. However, this method cannot generalize to other datasets that possibly have some tags that do not have manual tag-to-space mapping. Word2Vec regression is suitable in that case. It shows slightly lower but comparable retrieval performance and it can handle abundant vocabulary, even bigrams and tri-grams, without a manual mapping process.

Finally, we assess the performance of metric learning. Instead of predicting manually defined or pre-trained embeddings, metric learning aims at learning a shared embedding space across different modalities. Both two-branch and three-branch approaches claim their suitability for cross-modal retrieval, and the three-branch metric learning model consistently outperforms the two-branch model by leveraging the relationship of tag-to-text and tag-to-music within each modality.

5.2 Qualitative Results

To further investigate the characteristics of various embedding spaces, we visualize them with 2D projection—Figure 3. Due to limited space, we only visualize embedding spaces with Alm’s dataset and AudioSet mood subset. Note that they are all predicted embeddings using the test set. Except valence-arousal space (first row), which is already 2D, high dimensional embedding spaces are projected to a 2D space using the uniform manifold approximation and projection (UMAP) [55]. We use UMAP since it preserves more of the global structure compared to tSNE [56]. In the projection process, we first fit the UMAP with one modality (in our figure: music), then projected other embeddings (in our figure: tag and text) into the fitted 2D space.

First of all, for both the Word2Vec embedding space and the metric learning space, relevant moods from different taxonomies are neighboring together in the embedding space. This is natural for the Word2Vec space because each modality is fitted to optimize the pre-defined word embeddings. But this neighboring also can be found in metric learning space. In Figure 3-(g) and (h) for example, *anger* from text and *angry* from music are together, and *fearful* from text and *scary* from music are together. Note that Figure 3-(e) and (f) do not have word embeddings since the two-branch metric learning model does not have a branch to map the mood tags into the embedding space.

⁴ <https://github.com/minzwon/text2music-emotion-embedding.git>

Model	Retrieval	Distribution	Mapping
Classification	fail	.	.
Multi-head classification	fail	.	.
V-A regression	success	continuous	manual
W2V regression	success	discriminative	data-driven
Metric learning (2 branch)	success	discriminative	data-driven
Metric learning (3 branch)	success	continuous	data-driven

Table 4: Characteristics of different models

One of our main motivations to use metric learning with three branches is to preserve neighborhood structure within modalities. Since Word2Vec regression is a one-way optimization, their embeddings are very discriminative (Figure 3-(c)). Also, the two-branch neural network does not have any means to learn the neighborhood structure of each modality. Especially, as shown in Table 2, when two-branch metric learning uses the mapping of Alm’s mood into AudioSet mood with Word2Vec similarity, *exciting* and *tender* from music are not being used in training. If we compare Figure 3-(f) and (h), *exciting* music in (h) are more continuously distributed between *angry* and *happy* while they are simply with *happy* in (f). Also, when we compare text embeddings (see (e) and (g)), *surprised* is continuously distributed between *anger* and *happy* in (g) but not in (e). This continuity between music and text can be found in the manually annotated valence-arousal space (see (b) and (a), respectively), which means the proposed three-branch metric learning model preserves neighborhood structure within modalities in the learned multi-modal embedding space. We summarize all the introduced characteristics in Table 4.

6. CONCLUSION

In this work we tackled the task of matching music to text with the goal of allowing users to enhance their text-based stories with music that matches the mood of the text. We formulated the problem as a cross-modal text-to-music retrieval problem, and identified the lack of a shared vocabulary as a key challenge for bridging the gap between modalities. To address this challenge, we proposed and investigated several emotion embedding spaces, both manually defined (valence/arousal) and data-driven (Word2Vec and metric learning), to bridge between the text and music modalities. Our experiments showed that by leveraging these embedding spaces, we were able to facilitate cross modal retrieval successfully. We showed that the carefully designed valence-arousal space can bridge different modalities, but this can be also achieved via data-driven embedding spaces. Especially, our proposed three-branch metric learning model preserves the neighborhood structure of emotions within modalities. By leveraging data-driven embeddings, our approach has the potential of being generalized to other cross-modal retrieval tasks that require broader or completely different vocabularies.

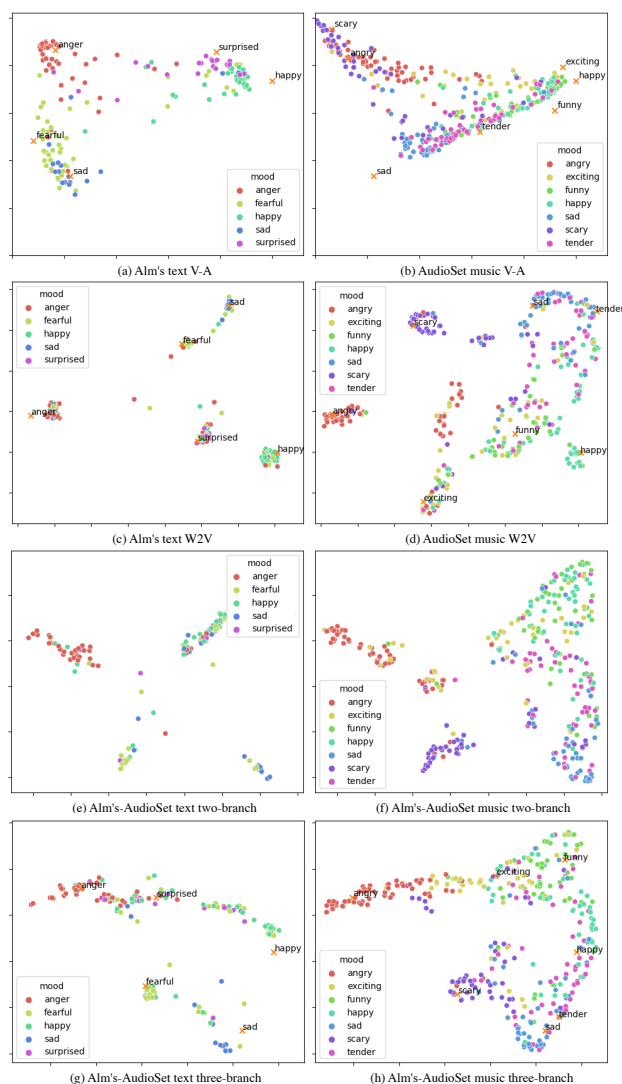


Figure 3: Valence-arousal embedding (first row), UMAP of Word2Vec embedding (second row), UMAP of shared embedding space from two-branch metric learning (third row), and UMAP of shared embedding space from three-branch metric learning (fourth row).

7. ACKNOWLEDGEMENT

This work was funded by the predoctoral grant MDM-2015-0502-17-2 from the Spanish Ministry of Economy and Competitiveness linked to the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502).

8. REFERENCES

[1] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, “Music emotion recognition: A state of the art review,” in *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2010.

[2] B. Wang, Y. Yang, X. Xu, A. Hanjalic, and H. T. Shen, “Adversarial cross-modal retrieval,” in *Proc. of the 25th ACM International Conference on Multimedia*, 2017.

- [3] T. Yao, T. Mei, and C.-W. Ngo, "Learning query and image similarities with ranking canonical correlation analysis," in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015.
- [4] L. Zhen, P. Hu, X. Wang, and D. Peng, "Deep supervised cross-modal retrieval," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [5] Y. Wei, Y. Zhao, C. Lu, S. Wei, L. Liu, Z. Zhu, and S. Yan, "Cross-modal retrieval with cnn visual features: A new baseline," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, 2016.
- [6] K. Wang, Q. Yin, W. Wang, S. Wu, and L. Wang, "A comprehensive survey on cross-modal retrieval," *arXiv preprint arXiv:1607.06215*, 2016.
- [7] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word-emotion association lexicon," *Computational Intelligence*, vol. 29, no. 3, 2013.
- [8] C. Strapparava, A. Valitutti *et al.*, "Wordnet affect: an affective extension of wordnet." in *Proc. of International Conference on Language Resources and Evaluation*, 2004.
- [9] T. Danisman and A. Alpkocak, "Feeler: Emotion classification of text using vector space model," in *AISB Convention: Communication, Interaction and Social Intelligence*, vol. 1, 2008.
- [10] M. Hasan, E. Rundensteiner, and E. Agu, "Emotex: Detecting emotions in twitter messages," 2014.
- [11] M. Abdul-Mageed and L. Ungar, "Emonet: Fine-grained emotion detection with gated recurrent neural networks," in *Proc. of annual meeting of the association for computational linguistics*, 2017.
- [12] E. Batbaatar, M. Li, and K. H. Ryu, "Semantic-emotion neural network for emotion recognition from text," *IEEE Access*, vol. 7, 2019.
- [13] D. Cortiz, "Exploring transformers in emotion recognition: a comparison of bert, distillbert, roberta, xlnet and electra," *arXiv preprint arXiv:2104.02041*, 2021.
- [14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *Proc. of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019.
- [15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.
- [16] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *Neural Information Processing Systems Workshop on Energy Efficient Machine Learning and Cognitive Computing*, 2019.
- [17] G. Tzanetakis, "Marsyas submissions to mirex 2007," *Music Information Retrieval Evaluation eXchange*, 2007.
- [18] G. Peeters, "A generic training and classification system for mirex08 classification tasks: audio music mood, audio genre, audio artist and audio tag," in *Proc. of the International Symposium on Music Information Retrieval (ISMIR)*, 2008.
- [19] C. Cao and M. Li, "Thinkit's submissions for mirex2009 audio music classification and similarity tasks," *Music Information Retrieval Evaluation eXchange*, 2009.
- [20] T. Lidy, A. Schindler *et al.*, "Parallel convolutional neural networks for music genre and mood classification," *Music Information Retrieval Evaluation eXchange*, 2016.
- [21] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music mood detection based on audio and lyrics with deep neural net," in *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [22] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [23] J. Lee, J. Park, K. L. Kim, and J. Nam, "Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms," in *Proc. of Sound and music computing (SMC)*, 2017.
- [24] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2018.
- [25] M. Won, S. Chun, , O. Nieto, and X. Serra, "Data-driven harmonic filters for audio representation learning," in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [26] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Metric learning vs classification for disentangled music representation learning," in *In Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [27] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," in *Proc. of Sound and Music Computing (SMC)*, 2020.

- [28] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [29] “Mediaeval 2020 emotion and theme recognition in music task: Loss function approaches for multi-label music tagging,” *MediaEval2020*, 2020.
- [30] E. M. Schmidt, D. Turnbull, and Y. E. Kim, “Feature selection for content-based, time-varying musical emotion regression,” in *Proc. of the international conference on Multimedia information retrieval*, 2010, pp. 267–274.
- [31] B.-j. Han, S. Rho, R. B. Dannenberg, and E. Hwang, “Smers: Music emotion recognition using support vector regression.” in *In Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2009.
- [32] J. A. Russell, “A circumplex model of affect.” *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.
- [33] R. E. Thayer, *The biopsychology of mood and arousal*. Oxford University Press, 1990.
- [34] S. Mohammad, “Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words,” in *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 174–184.
- [35] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, “Multimodal metric learning for tag-based music retrieval,” in *Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [37] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proc. of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [38] J. Choi, J. Lee, J. Park, and J. Nam, “Zero-shot learning for audio-based music classification and tagging,” in *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2019.
- [39] S. Doh, J. Lee, T. H. Park, and J. Nam, “Musical word embedding: Bridging the gap between listening contexts and music,” *arXiv preprint arXiv:2008.01190*, 2020.
- [40] R. Arandjelovic and A. Zisserman, “Look, listen and learn,” in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 609–617.
- [41] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, “Look, listen, and learn more: Design choices for deep audio embeddings,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [42] J. Huh, M. Lee, H. Heo, S. Mun, and J. S. Chung, “Metric learning for keyword spotting,” in *2021 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2021, pp. 133–140.
- [43] B. Elizalde, S. Zarar, and B. Raj, “Cross modal audio search and retrieval with joint embeddings based on text and audio,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [44] A.-M. Oncescu, A. Koepke, J. F. Henriques, Z. Akata, and S. Albanie, “Audio retrieval with natural language queries,” *arXiv e-prints*, pp. arXiv–2105, 2021.
- [45] L. Wang, Y. Li, J. Huang, and S. Lazebnik, “Learning two-branch neural networks for image-text matching tasks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 394–407, 2018.
- [46] L. Wang, Y. Li, and S. Lazebnik, “Learning deep structure-preserving image-text embeddings,” in *Proc. of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.
- [47] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, “Transformers: State-of-the-art natural language processing,” in *Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.
- [48] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, “Evaluation of algorithms using games: The case of music tagging,” in *In Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2009.
- [49] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, “Sampling matters in deep embedding learning,” in *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [50] E. C. O. Alm, *Affect in* text and speech*. Citeseer, 2008.
- [51] K. R. Scherer and H. G. Wallbott, “Evidence for universality and cultural variation of differential emotion response patterning.” *Journal of personality and social psychology*, vol. 66, no. 2, p. 310, 1994.
- [52] X. Hu, J. S. Downie, and A. F. Ehmann, “Lyric text mining in music mood classification,” *American music*, vol. 183, no. 5,049, pp. 2–209, 2009.

- [53] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, “The mtg-jamendo dataset for automatic music tagging,” *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning*, 2019.
- [54] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [55] L. McInnes, J. Healy, and J. Melville, “Umap: Uniform manifold approximation and projection for dimension reduction,” *arXiv preprint arXiv:1802.03426*, 2018.
- [56] L. Van der Maaten and G. Hinton, “Visualizing data using t-sne.” *Journal of machine learning research*, vol. 9, no. 11, 2008.

COLLAGENET: FUSING ARBITRARY MELODY AND ACCOMPANIMENT INTO A COHERENT SONG

Abudukelimu Wuerkaixi¹ Christodoulos Benetatos² Zhiyao Duan² Changshui Zhang¹

¹ Institute for Artificial Intelligence, Tsinghua University (THUI),
State Key Lab of Intelligent Technologies and Systems,
Beijing National Research Center for Information Science and Technology (BNRist),
Department of Automation, Tsinghua University Beijing, P.R.China

² Department of Electrical and Computer Engineering, University of Rochester

wekxabdk21@mails.tsinghua.edu.cn, c.benetatos@rochester.edu

zhiyao.duan@rochester.edu, zcs@mail.tsinghua.edu.cn

ABSTRACT

When writing pop or hip-hop music, musicians sometimes sample from other songs and fuse the samples into their own music. We propose a new task in the symbolic music domain that is similar to the music sampling practice and a neural network model named CollageNet to fulfill this task. Specifically, given a piece of melody and an irrelevant accompaniment with the same length, we fuse them into harmonic two-track music after some necessary changes to the inputs. Besides, users are involved in the fusion process by providing controls to the amount of changes along several disentangled musical aspects: *rhythm* and *pitch* of the melody, and *chord* and *texture* of the accompaniment. We conduct objective and subjective experiments to demonstrate the validity of our model. Experimental results confirm that our model achieves significantly higher level of harmony than rule-based and data-driven baseline methods. Furthermore, the musicality of each of the tracks does not deteriorate after the transformation applied by CollageNet, which is also superior to the two baselines.¹

1. INTRODUCTION

Recent years witnessed growing interest in symbolic multi-track music generation with the development of deep neural networks [1–3]. In particular, generating an accompaniment for a given melody has been a topic of interest [4]. Current deep learning models for accompaniment generation and music arrangement focus on the generation quality. Only a few methods incorporate user control into the generation process [5, 6].

In this work, we present a new task in the scope of multi-track music generation, specifically, *music fusion*.

¹ Code is available at <https://github.com/urkax/CollageNet>

Taking multiple unrelated music tracks as input, the task is to fuse them into a harmonic multi-track music piece, with some necessary changes to the input tracks; To involve users into the fusion process, users can control how much and on what aspects each input track can be changed. This task is similar to the music sampling practice, which started from hip-hop, and has been influencing pop and electronic music writing as well [7, 8]: Musicians sample melodies, rhythmic patterns, or other musical elements from other songs and fuse them into a new composition after certain changes [9, 10]. Our proposed task can be viewed as the first step towards the automation of the sampling practice. This task opens new possibilities in music arrangement and style fusion, and may lead to many creative applications involving user interaction into the music generation process.

In this paper, we concentrate on the fusion of a monophonic melody and an irrelevant polyphonic accompaniment with the same length. Specifically, we propose a neural network model named *CollageNet* to fuse the two tracks. We use two pretrained VAEs [11], one for the melody and the other for the accompaniment. The melody VAE computes a latent representation that disentangles *pitch* and *rhythm*, while the accompaniment VAE computes a latent representation that disentangles *chord* and *texture* [12, 13]. We then use adversarial training [14, 15] to train an actor model G to apply necessary transformations to the latent representations and decode them back to musical notes, to achieve a harmonic fusion of the two tracks while preserving a similarity to their original content. Because the latent representations are disentangled, the G model allows users to control the amount of changes along the disentangled musical aspects relatively independently by manipulating their corresponding latent vectors. An example of the input and output of the fusion process is displayed in Figure 1.

As this is a new task, there is no existing method to compare with. We therefore design two baselines, a rule-based method and a data-driven method. Objective and subjective experiments show that CollageNet significantly improves the harmony between the two tracks while



maintaining the similarities with the original input along user specified aspects. The achieved level of harmony is close to that of human-composed songs and is significantly higher than that of the two baselines. Besides, results also show that the musicality of each individual track does not deteriorate after the fusion.

The key contributions of this paper are as follows:

- We put forward a new task on symbolic multi-track music fusion, which is similar to the music sampling practice in music writing of modern genres.
- We propose a neural method, which allows users to control the degree of changes along several disentangled aspects of the input tracks in the fusion process.
- Objective and subjective experiments show that our proposed method outperforms two baseline methods in terms of harmony and musical quality.

2. RELATED WORK

2.1 Multi-track Music Generation

Multi-track music generation aims at generating music containing several tracks (parts) with different musical characters but constituting a pleasing whole. Some research focuses on harmonizing or accompanying a music track in an offline fashion [16] or an online fashion [17,18], while others focus on learning the representation of multi-track music [19–23]. DeepBach was proposed to generate Bach chorales using a graphical model [20]. Yan et al. proposed a part-invariant neural model to learn a representation of multi-part music [21]. Dong et al. proposed three models in different scenarios for multi-track generation using the GAN framework [22]. Simon et al. used a hierarchical VAE to model multi-track music [23].

2.2 Controllable Music Generation

There has been much attention to controllable generation in the image domain, such as CVAE [24, 25] and CGAN [26]. In recent years, there are also growing research interest in controllable symbolic music generation. Researchers proposed models to control quantifiable low-level musical attributes like note density, etc. Hadjeres et al. proposed a constrained method to train a VAE model with a regularized latent space [27]. Similarly, Pati et al. used a regularization loss within a mini-batch to train a controllable VAE model [28]. As for high-level musical features like musical arousal, Tan et al. proposed Music FaderNets to control them by sliding the corresponding low-level attributes. Music FaderNets are trained by first modelling the low-level attributes and learn the high-level features through semi-supervised clustering. [29].

2.3 Latent Space Transformations

There have been some studies in the image and text domain that learn transformations in the latent space. Engel et al. proposed to impose attributes on generated images through transformations in the VAE latent space [15]. Similar idea

was applied in music domain for connective fusion [30]. Shen et al. proposed to disentangle textual content from style by learning a shared content latent space for texts in different style [31]. Mueller et al. proposed to improve the input sequence by optimizing its latent vector of VAE [32].

3. PROPOSED METHOD

In this paper, we propose a new user-guided method that can transform and combine a two-measure-long melody and an unrelated accompaniment into harmonic two-track music while maintaining a similarity to their original content. Specifically, we encode the melody and the accompaniment with the encoders of two disentangled VAEs respectively. Then an actor model applies necessary transformations to the pairs of latent representations, and the decoders of the VAEs decode them back to musical notes. The actor model G is trained against the critic model D with adversarial training.

3.1 Model Architecture

Our model is based on the disentangled VAE framework [12, 13], where the encoder takes an input x and outputs a posterior $q(z|x)$ for the latent vector z to sample from, and the decoder $p(x|z)$ reconstructs the input. The latent vector z disentangles different musical aspects, each of which is encoded by a certain part of the vector. Given a pair of melody and accompaniment, we use the encoders of two VAEs to encode each to a latent vector. Specifically, we use EC²-VAE [12] to encode the melody input. The disentangled latent vector z_{mel} is a concatenation of a vector for *pitch* z_p and a vector for *rhythm* z_r , i.e., $z_{mel} = z_p \oplus z_r$ ². For the polyphonic accompaniment, we use the disentangled VAE in [13] to compute the latent vector z_{acc} , which is a concatenation of a *chord* vector z_c and a *texture* vector z_t , i.e., $z_{acc} = z_c \oplus z_t$.

After encoding the pair of melody and accompaniment into latent vectors z_{mel} and z_{acc} , we feed them to the actor model G which transforms them into another latent vector pair \hat{z}_{mel} and \hat{z}_{acc} . The actor model applies changes to the latent vectors to achieve transformations on the music content and the pair is supposed to be more harmonic. By applying different amount of changes to different parts of the latent vectors, the degree of transformations is controlled along the different music aspects. The inference process is displayed in Figure 2 (b).

While the encoders and decoders are pre-trained, the actor model G is trained under an adversarial framework together with a critic model D . The critic model is a binary classifier to distinguish positive samples and negative samples of the latent vectors. The definition of the positive and negative samples is described in Section 3.2. It is noted that the D model is not used in the inference process.

3.2 Training

Firstly, we pre-train the two VAEs for melodies and accompaniments. They are specially designed to learn a seman-

² \oplus denotes for concatenation

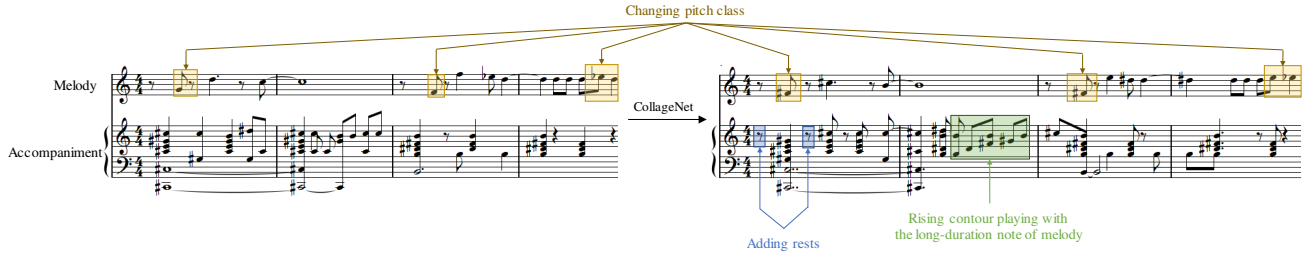


Figure 1. Example fusion result of CollageNet. An irrelevant pair of melody and accompaniment (left) is fused into a more harmonic pair while similarities to the input tracks are maintained. The control parameters are set to $c_{mp} = 0$, $c_{mr} = 1$, $c_{ac} = 0.8$, $c_{at} = 0$, so that *rhythm* of melody and *chord* of accompaniment are more preserved, while *pitch* of melody and *texture* of accompaniment are more altered. The bar lines are for clear visualization, not musically meaningful.

tically disentangled latent space, but fundamentally, they are both trained to maximize the evidence lower bound (ELBO) [12, 13]. The posterior $q(z|x)$ of VAE is trained to be close to the prior $p(z)$, which is the standard normal distribution. We obtain prior samples utilized for adversarial training by sampling latent vectors from $p(z)$.

Afterwards, we adversarially train the G and D models in the latent space. The training diagram is given in Figure 2 (a). Our dataset consists of two-track music segments, each of which has a monophonic melody track and a polyphonic accompaniment track. Suppose there are N music segments $\{x_{mel}^{(i)}, x_{acc}^{(i)}\}_{i=1}^N$, with the i -th melody segment indicated as $x_{mel}^{(i)}$ and the i -th accompaniment segment indicated as $x_{acc}^{(i)}$. We define a pair of melody and accompaniment with the same data index as a *harmonic pair* $\{x_{mel}^{(i)}, x_{acc}^{(i)}\}$, and define the set of harmonic pairs as the *harmonic pair set* Ω_h . To create disharmonic pairs accordingly, we randomly pick a melody $x_{mel}^{(i)}$ and an accompaniment $x_{acc}^{(j)}$ from the dataset with different data indexes ($i \neq j$). The *disharmonic pair* is indicated as $\{x_{mel}^{(i)}, x_{acc}^{(j)}\}$, and the *disharmonic pair set* is denoted as Ω_{dh} .

As discussed in Section 3.1, the D model is trained to distinguish between positive samples and negative samples. Positive samples are latent vectors of the harmonic pairs, indicated as $\{z_{mel}, z_{acc}\} \sim \Omega_h^z$. Negative samples include: (1) latent vectors of the disharmonic pairs $\{z_{mel}, z_{acc}\} \sim \Omega_{dh}^z$, (2) latent vectors sampled from prior $\{z_{mel}, z_{acc}\} \sim p(z)$, and (3) latent vectors produced by the actor model $G(z_{mel}, z_{acc})$. Following [15], we introduce the shorthand:

$$\begin{aligned} \mathcal{L}_{c=1}(z_{mel}, z_{acc}) &\triangleq -\log(D(z_{mel}, z_{acc})), \\ \mathcal{L}_{c=0}(z_{mel}, z_{acc}) &\triangleq -(1 - \log(D(z_{mel}, z_{acc}))). \end{aligned} \quad (1)$$

The training loss of the critic model D is as follows:

$$\begin{aligned} \mathcal{L}_D = & \mathbb{E}_{\{z_{mel}, z_{acc}\} \sim \Omega_h^z} [\mathcal{L}_{c=1}(z_{mel}, z_{acc})] \\ & + \mathbb{E}_{\{z_{mel}, z_{acc}\} \sim N^z} [\mathcal{L}_{c=0}(z_{mel}, z_{acc})] \\ & + \mathbb{E}_{\{z_{mel}, z_{acc}\} \sim N^z} [\mathcal{L}_{c=0}(G(z_{mel}, z_{acc}))], \end{aligned} \quad (2)$$

where $N^z = \Omega_{dh}^z \cup p(z)$.

The actor model G is trained to transform disharmonic latent vector pairs into a more harmonic pair. In other words, it is trained to fool the D model. For simplicity, we omit user control for this subsection, and we have: $\{\hat{z}_{mel}, \hat{z}_{acc}\} = G(z_{mel}, z_{acc})$. The outputs $\{\hat{z}_{mel}, \hat{z}_{acc}\}$ are expected to be close to the inputs $\{z_{mel}, z_{acc}\}$ to preserve a similarity on musical content. Therefore, the training loss of the actor model G consists of both an adversarial loss \mathcal{L}_{Ga} and a distance loss \mathcal{L}_{Gd} . The adversarial loss is:

$$\mathcal{L}_{Ga} = \mathbb{E}_{\{z_{mel}, z_{acc}\} \sim N^z} [\mathcal{L}_{c=1}(G(z_{mel}, z_{acc}))]. \quad (3)$$

The distance loss is to constrain the distance between the output and the input of the G model. For clarity, we define a distance function $\rho(\hat{z}, z) \triangleq \frac{1}{d_z} \|\frac{1}{\bar{\sigma}_z} \log(1 + (\hat{z} - z)^2)\|_1$ for two latent vectors z and \hat{z} in \mathbb{R}^{d_z} . The $\bar{\sigma}_z$ is the averaged scale of distribution $q(z|x)$ over the training set: $\bar{\sigma}_z = \frac{1}{N} \sum_n \sigma_z(x_n)$. We scale the distance penalty by the reciprocal of $\bar{\sigma}_z^2$ because latent vector dimensions with a smaller average scale contribute more to the identity of decoded data samples x [15]. Ignoring user control here, the distance loss is defined as follows:

$$\begin{aligned} \mathcal{L}_{Gd} = & \rho(\hat{z}_{mel}, z_{mel}) + \rho(\hat{z}_{acc}, z_{acc}), \\ \text{where } & \{\hat{z}_{mel}, \hat{z}_{acc}\} = G(z_{mel}, z_{acc}). \end{aligned} \quad (4)$$

The loss of the G model is the sum of these two parts, with distance penalty scaled by λ :

$$\mathcal{L}_G = \mathcal{L}_{Ga} + \lambda \mathcal{L}_{Gd}. \quad (5)$$

3.3 User Control

As discussed in Section 3.1, the latent vectors of melodies and accompaniments are disentangled into shorter vectors related to particular musical aspects. Specifically, $z_{mel} = z_p \oplus z_r$, and $z_{acc} = z_c \oplus z_t$. Users can control the amount of changes along these four musical aspects during the fusion process. To achieve that, aside from $\{z_{mel}, z_{acc}\}$, the input of the G model is extended with four scalars $c_{mp}, c_{mr}, c_{ac}, c_{at} \in [0, 1]$. These scalars respectively control *pitch* and *rhythm* of melodies, *chord* and *texture* of accompaniments.

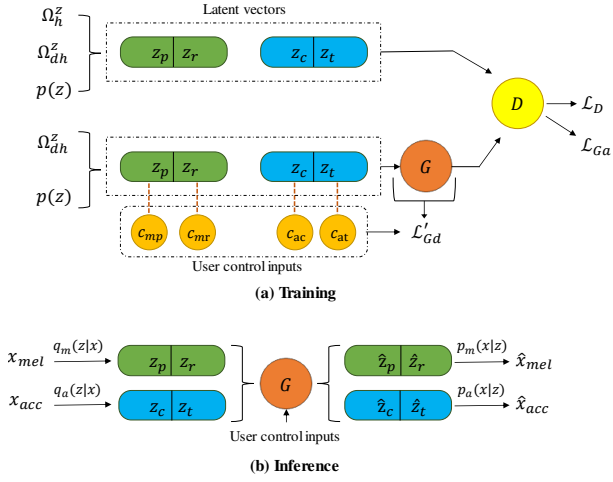


Figure 2. The training diagram of the G and D model and the inference diagram. The $q_m(z|x)$ and $p_m(x|z)$ are the encoder and decoder of the VAE for melodies. The $q_a(z|x)$ and $p_a(x|z)$ are from the VAE for accompaniments.

To impose such constraints on the G model, we randomly sample the scalars from the standard uniform distribution $U(0, 1)$ for each data sample during training. The distance penalty of latent vectors is scaled by the corresponding scalars. Therefore, the distance loss in Eqn (4) becomes:

$$\begin{aligned} \mathcal{L}'_{Gd} &= c_{mp} \cdot \rho(\hat{z}_p, z_p) + c_{mr} \cdot \rho(\hat{z}_r, z_r) \\ &\quad + c_{ac} \cdot \rho(\hat{z}_c, z_c) + c_{at} \cdot \rho(\hat{z}_t, z_t), \end{aligned} \quad (6)$$

where $\{\hat{z}_p \oplus \hat{z}_r, \hat{z}_c \oplus \hat{z}_t\} =$
 $G(z_p \oplus z_r, z_c \oplus z_t, c_{mp}, c_{mr}, c_{ac}, c_{at})$.

After being trained with distance loss \mathcal{L}'_{Gd} , the actor model G can respond differently to the control input. For instance, if the user adjusts c_{mp} to a large value, then the G model will produce \hat{z}_p close to the input z_p . Thus the *pitch* feature of the melody will hardly change after the transformation.

3.4 Implementation Details

For the disentangled VAEs, we use the same settings as original papers [12, 13], except that for EC²-VAE we do not use conditional information. The data representation for melodies and accompaniments also follow the VAE papers. Both the G and D model take in latent vectors z_{mel} and z_{acc} . Different from [30], we sample from $q(z|x)$ to get latent vectors. Before concatenation, each of z_{mel} and z_{acc} are passed through linear layers and ReLU activation. Then the concatenated vector is passed through 8-layer blocks made up of linear layers with 1024 outputs, ReLU activation, and dropout layers with rate of 0.5. For the output of the G model, we use the gate mechanism following [15]. The G and D models are trained using the Adam optimizer [33], with learning rate of $3e-5$, β_1 of 0, and β_2 of 0.9.

4. EXPERIMENTS

4.1 Dataset

We use the POP909 dataset [34], which contains melodies of 909 popular songs. Professional musicians composed piano accompaniments for them. We choose the songs with the time signature of 4/4 and randomly split them into 80%:10%:10% for training, validation, and test sets. Then we extract 8-beat long segments from them with a stride of 1 beat. We randomly select 40k segments for the training set, 5k segments for the validation set and 5k for the test set. We quantize time to 16th notes, so each segment is 32 steps long and we augment the training data by transposing them to all 12 keys.

4.2 Baseline Methods

As this is a new task, there is no existing methods to compare with. Therefore, we design a data-driven method and a rule-based method as baselines.

The data-driven baseline is derived from the proposed method. It also trains a critic model D to distinguish between harmonic pairs and disharmonic pairs. However, different from the proposed method, the D model in the data-driven baseline is pre-trained without the terms involving G . During the inference process, we use the pre-trained D model to implement gradient optimization $GradientDescent(z_{mel}, z_{acc}; \mathcal{L}_{c=1}(z_{mel}, z_{acc}))$. In other words, we optimize the inputs z_{mel} and z_{acc} to maximize the output of the pre-trained D model. We use the Adam optimizer [33] and the learning rate of 0.005 for both z_{mel} and z_{acc} .

The rule-based baseline applies revisions to the pitches and onsets of the melodies. According to music theory, to create a harmonic accompaniment for a melody, their notes should be on the same scale [35]. Besides, they need to be composed of matched rhythm. To fuse a pair of unrelated melody and accompaniment, the rule-based baseline tries to make their pitch class histogram similar and put their notes on the same onsets. At the same time, the revisions should be minor to preserve the identity of the inputs. The rule-based baseline only changes the input melody. For every note of the melody, we find the closest pitch class of the accompaniment notes. If the pitch distance is below the threshold of one semitone, we change the melody pitch to that pitch class. For example, if the pitch classes of the accompaniment notes are {C, E, F}, and a note of the melody is C#4. The closest pitch class is C, and the distance is below the pitch threshold of one semitone, then we change the C#4 to C4. As for rhythm, we move the notes of the melody to the same onsets of the accompaniment if the time distance is under the onset threshold of two steps. Besides, there is a 20% chance that a note retains even if it is changeable.

4.3 Evaluation of Harmony

We aim to fuse disharmonic pairs of melodies and accompaniments into harmonic pairs. In this subsection, we evaluate the level of harmony of the outputs of CollageNet and

	harmony rate	$\rho(\hat{z}_{mel}, z_{mel})$	$\rho(\hat{z}_{acc}, z_{acc})$
Ω_{dh}	10.10%	-	-
Data-driven baseline	61.70%	1.12	1.36
Rule-based baseline	67.27%	1.28	-
CollageNet-vanilla	92.59%	0.98	1.75
CollageNet ($\lambda = 0.1$)	92.71%	0.92	1.89
CollageNet ($\lambda = 0.5$)	89.58%	0.69	1.56
CollageNet ($\lambda = 1.0$)	89.56%	0.66	1.35
CollageNet ($\lambda = 2.0$)	84.58%	0.52	1.27

Table 1. Harmony rates of the disharmonic test set Ω_{dh} , and output from CollageNet (with different distance penalty λ) and two baseline methods, which take data from Ω_{dh} as inputs. Besides, the average latent space distances of melodies and accompaniments between the outputs \hat{z} and inputs z of the methods are also reported.

baseline methods. It is hard to design exhaustive metrics to evaluate the level of harmony. We use both the deep-learning model and musical statistics to evaluate the level of harmony.

We train a deep-learning evaluation model to discriminate between harmonic pairs and disharmonic pairs. The evaluation model uses a PianoTree encoder [36] to encode the polyphonic accompaniments, and a bidirectional GRU to encode the monophonic melodies. Then the encoded vectors are concatenated and passed through a multilayer perceptron (MLP) to produce a score between 0 and 1 for each pair of samples. The binary cross-entropy loss is used to train the evaluation model with the data from Ω_h and Ω_{dh} . After training, the accuracy is about 90% in the test set. We define *harmony rate* as the proportion of samples identified as positive by the evaluation model.

The harmony rates of the four methods are displayed in Table 1. The latent space distances $\rho(\hat{z}, z)$ between outputs \hat{z} and inputs z of the methods are also displayed. We report the results of CollageNet with different distance penalty λ . In addition to CollageNet and two baselines, we also evaluate CollageNet-vanilla, which utilizes vanilla VAEs [36] instead of disentangled VAEs. According to the results, CollageNet can produce music with higher harmony rates while making fewer changes to the inputs. Besides, with a higher distance penalty, the performance of CollageNet degrades slightly, and the latent space distances reduce. It is noted that CollageNet and CollageNet-vanilla achieve comparable harmony rates, but the disentangled VAEs in CollageNet provides user control in the fusion process as described in Section 3.3 and validated in Section 4.5.

Although the deep-learning model evaluates more comprehensively, it is agnostic. Inspired by [37, 38], we adopt several musical statistics to evaluate the level of harmony of each pair of melody and accompaniment. Firstly, we extract several features from both melodies and accompaniments. **PCH** is the pitch class histogram with 12 bins. **OH** is the onset histogram with 32 bins corresponding to 32 time steps. **RE** is the rhythm pattern, a 32-dimensional vector denoting states of every time step, including onsets, holding states of any pitch, and rests. The **PCH** feature reveals the pitch pattern of melodies and accompaniments, while **OH** and **RE** reveal the rhythm pattern. For **PCH** and

	PCH		OH		RE ↑
	KLD↓	OA↑	KLD↓	OA↑	
Ω_h	0.96	0.578	2.171	0.471	0.643
Ω_{dh}	5.02	0.292	4.522	0.299	0.457
Data-driven baseline	2.69	0.397	3.421	0.377	0.531
Rule-based baseline	1.91	0.459	1.831	0.464	0.662
CollageNet	1.38	0.588	2.228	0.476	0.612

Table 2. The musical statistics averaged over datasets for harmony evaluation. The Ω_h is the harmonic test set. Two baseline methods and CollageNet take data from the disharmonic test set Ω_{dh} as inputs. The arrows indicate a better direction.

OH, we calculate the Kullback-Leibler Divergence (KLD) and Overlapping Area (OA) between the melody and accompaniment of each pair. For **RE**, we calculate the ratio of the same pattern between the melody and accompaniment. The average values of these statistics are in Table 2. According to the results, the outputs of the three methods get closer to the harmonic test set Ω_h than inputs from Ω_{dh} . CollageNet is significantly better in most metrics. The rule-based baseline is better than the data-driven baseline because it directly optimizes these metrics.

4.4 Evaluation of Music Quality

To fuse a pair of unrelated melody and accompaniment, CollageNet and baseline methods change the inputs. The fusion process may destroy the musicality of each of the input tracks. Thus, we compare several musical statistics between created datasets and the original dataset to evaluate the quality of transformed melodies and accompaniments respectively. The datasets whose statistics are closer to the test set are more similar to human-made music [37]. Different from Section 4.3 where the comparison is between each pair of melody and accompaniment, the comparison happens between two datasets in this subsection.

We calculate **PC** (pitch count), **PI** (pitch interval), and **IOI** (inter-onset-interval) as in [37] from melodies and accompaniments respectively. Table 3 shows the results. Except for CollageNet and baseline methods, we also calculate the statistics of music generated by VAEs. The VAEs generate music by sampling latent vectors from the prior $p(z)$ and decoding them to musical notes. According to the results, although the rule-based baseline outperforms the data-driven baseline in Table 1 and Table 2, its outputs are very different from the real data. As for CollageNet, the musicality of melodies and accompaniments does not deteriorate after the transformation. The musical statistics of CollageNet are closest to the test set.

4.5 Subjective Experiment

We implement subjective experiments to evaluate CollageNet and the rule-based baseline methods. Before the test, we ask the subjects three questions following [18]:

Do you master any musical instruments? Have you received vocal training before? Have you learned music theory systematically before?

	melody			accompaniment	
	PC	PI	IOI	PC	IOI
test set	10.7	0.024	3.1	31.6	2.02
VAE	-0.30	+0.054	+0.05	-3.7	+0.00
Data-driven baseline	-0.99	+0.099	+0.33	-2.3	+0.14
Rule-based baseline	-1.33	+0.102	+0.33	-	-
CollageNet	-0.10	+0.010	+0.03	-1.2	+0.01

Table 3. The average musical statistics of melodies and accompaniments in the test set Ω_h . For four methods, we report the difference of their outputs from the test set. The rule-based baseline does not alter accompaniments.

We denote the subjects who answer *yes* to any of these questions as *trained*, other subjects as *not-trained*. We invite 38 people to complete the survey. Among them, 20 people are trained and 18 people are not-trained.

Each subject listens to 16 randomly shuffled and anonymous music pieces, comprising of 4 pieces from the disharmonic pair set Ω_{dh} , 4 pieces from outputs of the rule-based baseline, 4 pieces from outputs of CollageNet, and 4 pieces from the harmonic pair set Ω_h . Therefore, 142 pieces of each kind of data are evaluated in total. The music pieces are rendered using violin for melodies and piano for accompaniments. The subjects rate the harmony of these pieces on a 5-point scale where “1” indicates “disharmonic” and “5” indicates “harmonic”. They are told to concentrate on the coherence of melodies and accompaniments. Figure 3 (a) illustrates the average harmony score. The outputs of CollageNet are rated as more harmonic than the rule-based baseline.

Then each subject is asked to listen to four pieces of melodies from outputs of the rule-based baseline, outputs of CollageNet, and Ω_h each; four pieces of accompaniments from outputs of CollageNet and Ω_h each (the rule-based baseline does not revise accompaniments). They judge whether each piece is composed by humans or generated by machine. Figure 3 (b) illustrates the average percentage of pieces rated as human-made by the subjects. Although the rule-based baseline can fuse the disharmonic inputs, nearly half of the output melodies of the rule-based baseline are regarded as machine-made. CollageNet produces both harmonic and high-quality music. Such obser-

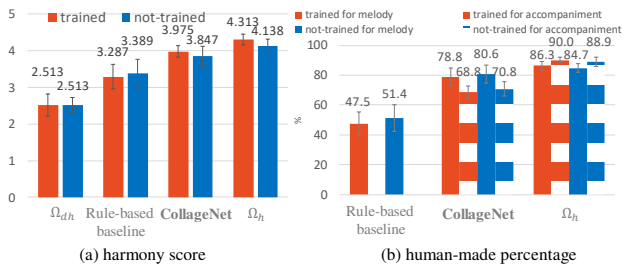


Figure 3. Average harmony score for two-track segments from four datasets are displayed. And human-made percentage are calculated for melodies and accompaniments respectively, which are from three datasets. The scores of trained subjects and not-trained subjects are displayed.

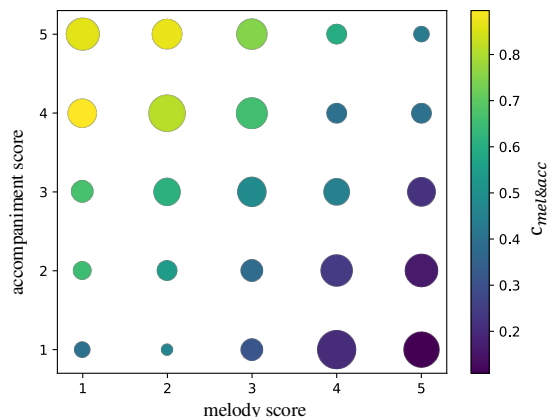


Figure 4. All similarity scores on a 5-point scale given by subjects for melodies and accompaniments. The melodies and accompaniments are produced by CollageNet with different user control $c_{mel&acc}$. Each circle represents all the songs with the specific melody score and accompaniment score. The size of the circles indicates the number of the songs, and the color of the circles indicates the average $c_{mel&acc}$ of the songs.

vation is consistent with the conclusions of Section 4.4.

To demonstrate the validity of CollageNet’s user control, the subjects listen to the output melodies and accompaniments respectively of CollageNet with sliding user control inputs. And the subjects rate the similarity of each output melody and accompaniment to the inputs. We define the term $c_{mel&acc} = c_{mp} = c_{mr} = 1 - c_{ac} = 1 - c_{at}$. Each subject rates two groups of songs, with each group consists of six melodies and accompaniments produced with different $c_{mel&acc}$ sliding from 0 to 1. The scores are on a 5-point scale where “1” indicates “similar” and “5” indicates “different”. Figure 4 displays all the scores. As the $c_{mel&acc}$ increases, the output melodies are considered more similar to the input, while the output accompaniments are the opposite.

5. CONCLUSION

In this paper, we presented a new task and a neural approach on multi-track music fusion, which is similar to the music sampling practice. Specifically, given an unrelated pair of melody and accompaniment of the same length, the proposed approach fuses them to produce harmonic two-track music while maintaining their musical identity. Besides, users can control the magnitude of changes along disentangled musical aspects. We conducted objective and subjective experiments and compared the proposed approach with rule-based and data-driven baseline methods. Experimental results showed that the proposed method achieved significantly higher level of harmony than that of baselines, with musically high-quality outputs.

For future work, CollageNet can be extended to arbitrary tracks and longer music pieces. Besides, similar ideas and systems can be explored in the audio domain.

6. ACKNOWLEDGEMENT

This work is funded by the National Key Research and Development Program of China (No. 2018AAA0100701) and the National Science Foundation grant No. 1846184 of the USA.

7. REFERENCES

- [1] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. C. Courville, and D. Eck, "Counterpoint by convolution." in *Proceedings of the 18th International Society for Music Information Retrieval Conference*. Suzhou, China: ISMIR, Oct. 2017, pp. 211–218.
- [2] C. Donahue, H. H. Mao, Y. E. Li, G. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 685–692.
- [3] N. Jiang, S. Jin, Z. Duan, and C. Zhang, "RI-duet: On-line music accompaniment generation using deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 710–718.
- [4] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020.
- [5] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.
- [6] H. Chu, R. Urtasun, and S. Fidler, "Song from pi: A musically plausible network for pop music generation," *arXiv e-prints*, pp. arXiv-1611, 2016.
- [7] B. J. McCann, *The mark of criminality: Rhetoric, race, and gangsta rap in the war-on-crime era*. University of Alabama Press, 2017.
- [8] N. Patrin, *Bring That Beat Back: How Sampling Built Hip-Hop*. U of Minnesota Press, 2020.
- [9] S. Howell, "The lost art of sampling: Part 4," *Sound on Sound Magazine*, 2005.
- [10] Wikipedia contributors, "Sampling (music) — Wikipedia, the free encyclopedia," 2021, [Online; accessed 26-April-2021]. [Online]. Available: [https://en.wikipedia.org/w/index.php?title=Sampling_\(music\)&oldid=1013854871](https://en.wikipedia.org/w/index.php?title=Sampling_(music)&oldid=1013854871)
- [11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
- [12] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 596–603.
- [13] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020.
- [14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, p. 2672–2680.
- [15] J. Engel, M. Hoffman, and A. Roberts, "Latent constraints: Learning to generate conditionally from unconditional generative models," *arXiv preprint arXiv:1711.05772*, 2017.
- [16] M. Allan and C. K. Williams, "Harmonising chorales by probabilistic inference," *Advances in Neural Information Processing Systems*, vol. 17, pp. 25–32, 2005.
- [17] C. Benetatos, J. VanderStel, and Z. Duan, "Bachduet: A deep learning system for human-machine counterpoint improvisation," in *Proceedings of the International Conference on New Interfaces for Musical Expression*, 2020, pp. 635–640.
- [18] N. Jiang, S. Jin, Z. Duan, and C. Zhang, "When counterpoint meets chinese folk melodies," in *NeurIPS*, 2020.
- [19] F. T. Liang, M. Gotham, M. Johnson, and J. Shotton, "Automatic stylistic composition of bach chorales with deep lstm." in *ISMIR*, 2017, pp. 449–456.
- [20] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1362–1371.
- [21] Y. Yan, E. Lustig, J. VanderStel, and Z. Duan, "Part-invariant model for music generation and harmonization," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*. Paris, France: ISMIR, Sep. 2018, pp. 204–210.
- [22] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

- [23] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," *arXiv preprint arXiv:1806.00195*, 2018.
- [24] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, pp. 3483–3491, 2015.
- [25] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *arXiv preprint arXiv:1906.02691*, 2019.
- [26] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
- [27] G. Hadjeres, F. Nielsen, and F. Pachet, "Glsr-vae: Geodesic latent space regularization for variational autoencoder architectures," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–7.
- [28] A. Pati and A. Lerch, "Latent space regularization for explicit control of musical attributes," in *ICML Machine Learning for Music Discovery Workshop (MLMD), Extended Abstract, Long Beach, CA, USA*, 2019.
- [29] H. H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020.
- [30] T. Akama, "Connective fusion: Learning transformational joining of sequences with application to melody creation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020.
- [31] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," *arXiv preprint arXiv:1705.09655*, 2017.
- [32] J. Mueller, D. Gifford, and T. Jaakkola, "Sequence to better sequence: continuous revision of combinatorial structures," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2536–2544.
- [33] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [34] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020.
- [35] S. Porter, *The Harmonization of the Chorale: A Comprehensive Workbook Course in Harmony and Counterpoint*. Taylor & Francis, 1987.
- [36] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "Pianotree vae: Structured representation learning for polyphonic music," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020.
- [37] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.
- [38] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research*, vol. 50, no. 1, pp. 37–51, 2021.

HUMAN-IN-THE-LOOP ADAPTATION FOR INTERACTIVE MUSICAL BEAT TRACKING

Kazuhiko Yamamoto
YAMAHA Corporation

ABSTRACT

In music information retrieval (MIR), beat-tracking is one of the most fundamental and important task. However, a perfect algorithm is difficult to achieve. In addition, there could be a no unique correct answer because what one interprets as a beat differs for each individual. To address this, we propose a novel human-in-the-loop user interface that allows the system to interactively adapt to a specific user and target music. In our system, the user does not need to correct all errors manually, but rather only a small portion of the errors. The system then adapts the internal neural network model to the target, and automatically corrects remaining errors. This is achieved by a novel adaptive runtime self-attention in which the adaptable parameters are intimately integrated as a part of the user interface. It enables both low-cost training using only a local context of the music piece, and by contrast, highly effective runtime adaptation using the global context. We show our framework dramatically reduces the user’s effort of correcting beat tracking errors in our experiments.

1. INTRODUCTION

Musical beat is among the most important factor in music [1]. Many MIR systems first analyze the beats as the starting point, assume the results would be correct, and use them as a unit for further processing [2–4]. However, although the performance of many MIR algorithms, including beat tracking, has improved dramatically through recent deep learning-based approaches [5–10], a perfect algorithm is difficult to achieve in principle, and errors are bound to occur. In addition, what one interprets as a beat differs for each individual. There is thus no unique answer to the question “*What is the correct beat?*” for a music. Many existing music pieces also show that the beats we want vary from time to time [11, 12]. This means that the ideal beat tracking system needs to produce different outputs from a single input depending on the situation. This is difficult to deal with using machine learning systems.

To address this, we propose an interactive beat-tracking interface for adapting to a specific user and target music using a human-in-the-loop approach (Figure 1). In this system, the user provides feedback for the temporal result

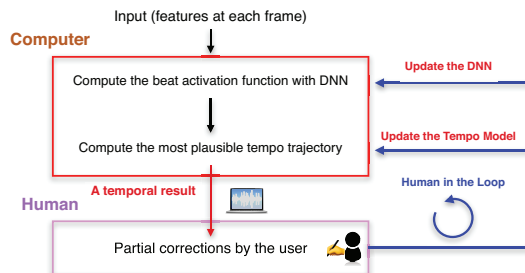


Figure 1. An overview of the system.

of the system by correcting the error, and the system then computes the output again. By iterating this interaction between the user and computer, the system adapts the internal neural network model (DNN) to the user, allowing it to produce more desirable results for the user. The user does not need to correct all errors manually, but rather only a small portion of the errors that are noticed. The system adapts the internal DNN to follow the user’s intention and automatically corrects the remaining errors that the user has not corrected directly. This enables a dramatic reduction in the user’s effort to obtain the desired result¹.

To achieve such an online adaptation, we present a novel adaptive runtime self-attention model (ARSA), in which the adaptable parameters are intimately integrated as a part of the interactive user interface. This is a variant of the self-attention mechanism [13], and similarly to other models, it connects the entire input sequence (intermediate feature sequence) globally throughout the target music piece by the attention map. However, the most significant difference in the ARSA is not to be pre-trained at all. We embed the ARSA into an another pre-trained neural network only at runtime and use it for adaptation. This means that our model architecture changes between the training and runtime.

Our model is trained using only a local context for the training dataset and adapts to the user using the global context at runtime. This locally-aware learning reduces the computational cost during training, and the globally-aware runtime adaptation allows the effects of locally modified user feedback to be distributed throughout the entire piece of music. This strategy is based on our assumption that the amount of local feedback that the user can practically provide could be insufficient for adaptation. To assess the feasibility and effectiveness of the proposed system, we validate it through a simulation environment and a user study. In addition, we show the extent to which the proposed system improves the efficiency of error correction in beat tracking.

¹ The supplemental materials: www.yamo-n.org/hilbeats

2. RELATED WORK

2.1 Musical Beat Tracking

The state-of-the-art techniques in beat tracking are machine learning-based, and use a two-stage processing. They first compute the beat activation function by a DNN, which is a probability-like representation of the beat distribution, from which they then determine the actual beats by estimating the most plausible tempo trajectory throughout the target music. Böck and Schedl [6] use a bidirectional long short-term memory neural network (BLSTM) to compute the beat activation function. Davies and Böck [5] use temporal convolutional networks (TCN) to improve the computational efficiency and the quality of the activations. To determine the actual beat positions from the beat activation function, Krebs et al. [14] use a dynamic Bayesian network (DBN), which is approximated as a hidden Markov model (HMM) solved using the Viterbi algorithm. Some studies have also reported that the combined modeling of the beat, downbeat, and tempo improves the beat tracking quality [7, 9]. The desired beat tends to depend on the genre. Böck et al. [8] use multiple pre-trained models adapted to different types of musical genres, and select one of them at runtime to the most plausible result. However, it is difficult to guarantee whether the classification of the genre would be appropriate.

2.2 Human-in-the-Loop System in MIR

The concept of human-in-the-loop involves humans as a part of the system. Sonic Visualizer [15] and Dixon [16] present interactive editing tools for beat tracking. They visualize the beats, and the user manually corrects the errors. However, such manual corrections are a burden to the user. Driedger et al. [17] propose a hybrid method for annotating beats by manual tapping and through the use of an automatically computed beat activation function. They correct the inaccurate tap to be closer to the activation function. However, this would be difficult to use if the activation is not sufficiently close to the desired result. Nakano et al. [18] attempt to improve the accuracy of the singing voice separation from mixed audio by fine-tuning a trained DNN at runtime. They use the user-corrected F0 curve as the new training data for the DNN. This concept is similar to that of our ours. However, using only a local correction that the user can provide through practical means might be insufficient for proper model adaptation.

Songle [19] provides a web-based error correction tool for several musical factors, including beats, by using crowdsourcing contributions. HumanGAN [20] also uses crowdsourced human evaluations as an evaluation function to determine whether a voice is realistic or not when it trains a singing voice synthesis DNN. However, we can not obtain the desired result for a specific user by applying these approach. Bryan et al. [21] propose an interactive sound source separation method by masking the spectrograms on the GUI. The system uses paintable masks to adapt the internal algorithm. Bazin et al. [22] also present an inpainting-based control on the spectrogram for sound synthesis using token-masked Transformers [13] and VQ-

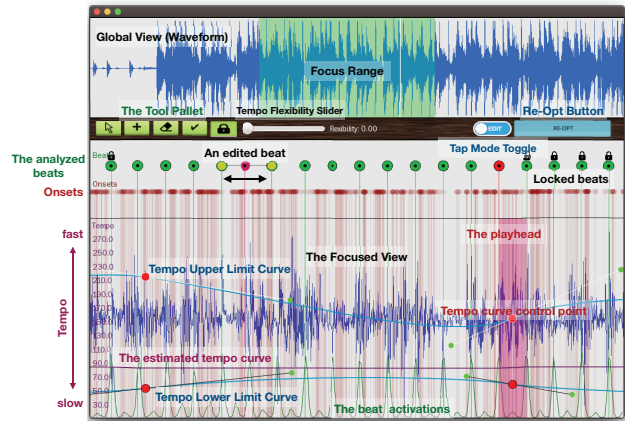


Figure 2. A graphical user interface for the system.

VAE [23]. Zhou et al. [24] use Bayesian optimization to efficiently explore the latent space of the melody generative model and obtain the desired output.

Bongjun and Bryan [25] propose an interactive interface for an effective sound event annotation. When the user makes a selection, several similar candidate positions are presented. The user listens to them and labels them as positive if they are actually the target, and otherwise labels them as negative. The system uses this feedback to update the similarity search model such that it shows more similar candidate locations. Our approach is partly inspired by them, and uses a similar method for treating the beats that are expected to be significantly affected by a user-modification (§.5.3).

3. USER INTERFACE

When the user inputs the target music, the system first displays the result of the beat tracking on the UI screen as the initial state (Figure 2). On the screen, the waveforms of the entire piece of music (top row) and a focused area (bottom row) are displayed. The user can zoom into/out of the arbitrary range by pinching in/out on the trackpad of the PC and check the results by playback. During playback, the system represents the beat by making a clicking sound when the playhead reaches a beat location. The user listens to it, and if the user finds an error or undesired result, the user can modify it. After certain modifications, the user presses the “Re-Opt” button (the blue button), and the system then starts the adaptation. The progress of the optimization process is displayed on the screen and updated at each iteration. The user can stop the process at an arbitrary timing.

To correct the errors by the user, we provide several options. These tools are categorized into three types. The first type is simply editing the beat directory (**Move/Insert/Remove/Lock/Tap**). The move tool allows the movement of the beat position by dragging the mouse. The insert and remove tools are used to add a new beat and remove an existing beat. The lock tool is used to fix the beat positions. By using a tap tool, the user can specify the beat directly during playback by taping a key on the PC keyboard.

The second tool type constrains the tempo trajectory, which affects the distribution of the beats throughout the

entire piece of music (§.5.4). There are two tools. First, the **Tempo Range Limit Setting Tool** allows the upper and lower limits of the tempo search range to be set by a curve editor. The user can add keyframes at arbitrary points and set the tempo range using Bezier curves. The system then excludes tempos that fall outside of this range from the search. Next, the **Tempo Change Flexibility Slider** determines the stability of the tempo tracking. The higher the value is, the more the system allows the tempo to vary widely throughout the music. For example, for rock and pop songs, where the tempo often remains almost constant, this value can be set to a low value, whereas for classical music, where the tempo often changes significantly, a high value can often achieve better results.

The last type is a special compared with the two above. This is the **Label Annotation Tool** that is used to provide the user’s feedback to the system and improve the adaptation ability. This tool can be used only after a beat modification using the first tool type. When the user edits a beat, it recommends beats that have the possibility to be significantly affected by the user’s modification by applying question marks, as below.



Figure 3. The label annotation tool.

In the above figure, the red beat is the beat currently edited the user. The user listens to some (not necessarily all) of the recommended locations (question marks) and judges whether they really need to be modified or not. If a correction is necessary, the user assigns a positive label to the location; otherwise the label is negative. This can be achieved by clicking on the label mark with the mouse. Each time the user clicks on the same mark, the positive and negative labels are alternately switched. The user can also directly modify the recommended beat, but does not have to correct it and leave. This is an important aspect. The effort of this labeling is minimal, whereas the effort required to edit a user’s beat manually is significant. This is because it only takes a moment to listen to the specified portion and judge whether it needs to be modified. In addition, this also has a merit in that we can fix the location of the negative labeled beat at this point. The system uses these assigned labels to compute the attention map in the ARSA (§.5.3).

4. BEAT TRACKING

Our beat-tracking algorithm basically follows the state-of-the-art algorithm [5]. First, we divide the target music into small frames and obtain the features for each frame. We use the time difference of the mel-frequency cepstrums (MFCCs) for the feature (one can also use alternative features such as CQT [26]). We used 44.1kHz for the sampling rate, $W = 4096$ samples for the Hanning window lengths, and $W/2$ for the hop size. We then input this feature series of T frames into the DNN and compute the beat activation function for each frame. As described in §.1, we train this DNN using only a local context of the mu-

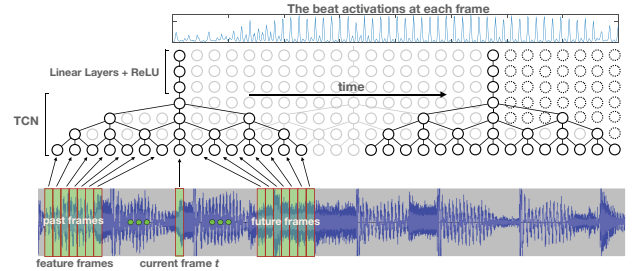


Figure 4. Our DNN is trained with a local manner.

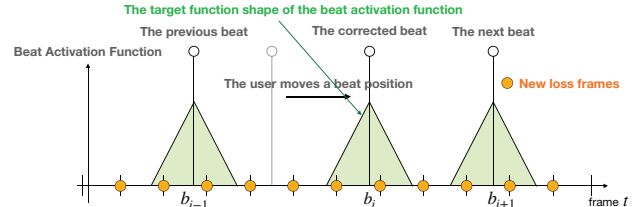


Figure 5. The target function shape of the beat activation.

sic piece. We use three serial connected TCN blocks [5] (five layers with $\{1, 2, 4, 8, 16\}$ dilations and the gated activations [29]), followed by three linear layers with the ReLU (Figure 4). We note that the figure shows only a single TCN block for simplicity. TCN is a variant of a dilated convolutional neural network that is inspired by WaveNet [29]. It takes the past and future frames around the focused frame. We use the frames for approximately 2.4 seconds for past and future frames. To train this DNN, we used four types of datasets: GTZAN [31–33], the RWC popular and genre dataset [34,35], and an in-house dataset of 400 musical pieces of various genres purchased from Amazon Music [36] and annotated by a vendor. Note that although our DNN architecture is slightly different from [5], we could not observe significant differences in our experiments.

To determine the actual beat positions from the beat activation function, we use the the hidden semi-Markov model (HSMM) [27, 28], which applies the activation as the observation state and the tempo as the hidden state. It estimates the most plausible trajectory throughout the entire piece of music by solving the Viterbi algorithm. Our method is basically similar to the state-space tempo model [8, 14] that uses the HMM. As a significant difference, our HSMM treats the midway phase of each tempo as the hidden state (Figure 6: Right). This HSMM divides the duration of a beat of each tempo into microphases, and gradually moves through the intermediate states sequentially until it reaches the next beat head. Only when the state reaches the beat head state, it is allowed to transit to a different tempo.

5. ADAPTATION ALGORITHM

5.1 Loss Making from the User’s Corrections

We treat the user-modified beat and the two adjacent beats, for a total of three beats, as a unit. We describe the situation in which the user moves a beat position as an example here, but the basic idea is similar to other editing methods. Assuming that the user-modified beat position is b_i and the beat positions before and after it are b_{i-1}, b_{i+1} , we make the DNN output to be close to one at b_i , and zero at the midpoint $b_i^- = \frac{b_i + b_{i-1}}{2}$ and $b_i^+ = \frac{b_i + b_{i+1}}{2}$, (Figure 5).

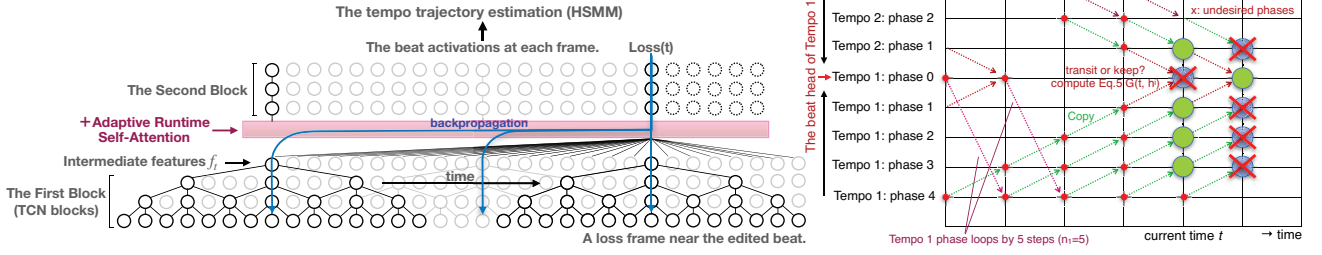


Figure 6. Left: At runtime, we split the pre-trained model into two blocks and insert the adaptive runtime self-attention block between them (§.5.2). Right: Hidden semi-Markov model for estimating the most plausible tempo trajectory (§.5.4).

Therefore, we define the loss function at an orange color frame t (referred to as the loss frame in the figure) as,

$$L(t) = \left| DNN(t) - \left(1 - \min\left(1.0, \frac{|b^c - t|}{W}\right) \right) \right|_2 \quad (1)$$

where b^c and W denote the closest beat from the frame t and the window size, respectively. We minimize the sum of the loss functions at all the loss frames by a gradient descent, such as Adam [37], to adapt the DNN.

5.2 Adaptive Runtime Self-Attention

As described, our DNN is trained using only a local context during pre-training, taking approximately 2.4 seconds of each past and future frame around the interest frame as input, and shifting the frames individual. By contrast, at runtime, we split the DNN into two parts, the first half (three blocks of TCNs) and the second half (three layers of linear layers and ReLU activations), and insert the ARSA between them (Figure 6:Left). This means that the model architecture itself has changed. The ARSA connects all of the intermediate features (the outputs of the first block) throughout the entire piece of the music and outputs the input for the second block at time frame t . This new type of DNN model achieves a better balance between the reduced computational cost during pre-training and an improvement of the runtime adaptation capability that distributes locally modified user feedback to throughout the entire piece of music.

Now supposing that the series of the outputs of the first block (intermediate features) $\{f^1, f^2, \dots, f^T\}, f^t \in \mathbb{R}^D$, similar to general attention, the ARSA first computes three parameters at each frame t as: $Query^t = \mathbf{W}_Q \cdot f^t$, $Key^t = \mathbf{W}_K \cdot f^t$, $Value^t = \mathbf{W}_V \cdot f^t$, where $\mathbf{W}_Q, \mathbf{W}_K$, and $\mathbf{W}_V \in \mathbb{R}^{D \times D}$ are the matrices that are initialized as identity matrices. General attention uses the inner products between queries and keys to construct the attention map. Alternatively, we use the weighted inner product of the $Query^t$ and Key^i as,

$$d(t, i) = (w^t \otimes Query^t) \cdot (P_e(|t - i|) \otimes Key^i), \quad (2)$$

where $w^t \in \mathbb{R}^D$ is the weight that is interactively determined by the user's label annotation (§.3). We describe this in §.5.3. $P_e(\delta t) \in \mathbb{R}^D$, $P_e(\delta t)_k = \sin(\omega_n \delta t)$, if $k = 2n$; otherwise, $\cos(\omega_n \delta t)$, where $\omega_n = 1.0/0.0001^{2n/\frac{D}{2}}$, denotes the relative positional encoding [13] that represents how apart the frame i is from the frame t . This $d(t, i)$ becomes the attention map at the frame t toward the frame i

in the ARSA. After sorting this attention map $d(t, i)$ with respect to i , we then sample N frames in descending order in a manner of an importance sampling, and store them into \mathbb{C}_t (we use $N=512$). Then, the output of the ARSA at frame t can be formulated as follows,

$$y_t = (1 - \alpha) \cdot Value^t + \alpha \sum_{i \in \mathbb{C}_t} \frac{\exp(d(t, i))}{\sum_{j \in \mathbb{C}_t} \exp(d(t, j))} \cdot Value^i, \quad (3)$$

where α denotes the weight as a hyperparameter (we set 0.5 in our experiment).

5.3 Interactively Adapting the Attention

The weight w^t in Eq.2 is a unique parameter of ARSA that is computed using the user's interactive feedback. As described in §.3, when the user edits a beat, the label annotation tool first shows the beats that are expected to be significantly affected by the modification (the question marks in the Figure 3). These locations can be derived naturally from the original principles of the attention mechanism. The fact that the attention map $d(t, i)$ is larger means that the frame t is more focused on the frame i . In other words, the amount of back propagation from a given loss in frame t would be distributed more to frames with larger $d(t, i)$. Therefore, we can state that the beats within the vicinity of frames with large attention map values tend to be more susceptible to the user modification. Thus, the system shows these beats to correct the user feedback.

For the recommended beats, the user judges whether they really need to be corrects, and assigns positive/negative labels to them. The assigned label on a beat is redistributed to the neighboring frames, as shown in the following figure.

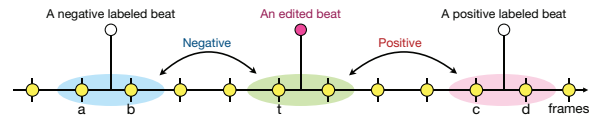


Figure 7. The labels are distributed to the neighbor frames.

These distributed labels are used to determine the relationship between each frame and the interest frame t . For example, in Figure 7, the frames $\{a, b\}$ are negative for t , and $\{c, d\}$ becomes positive for t . Using these relationships of the frames for the interest frame t , we then compute the Fisher's criterion [25, 38] as

$$w^t = \frac{(\text{avg}(A^{t,pos}) - \text{avg}(A^{t,neg}))^2}{\text{std}(A^{t,pos})^2 + \text{std}(A^{t,neg})^2}, \quad (4)$$

where $A^{t,pos}$ and $A^{t,neg}$ represent $Query^t \otimes P_e(|t-i|) \otimes Key^i$ in the positive and negative frames i for the interest frame t , respectively. Note that if no feedback from the user is obtained, we set $w^t = 1$. This weight reduces the effects from the user’s modification on the frames that the user does not want to be changed, and conversely increases the effects simultaneously on the frames that the user wants to more aggressively modify. It is therefore expected to improve the adaptation speed.

5.4 HSMM with The User’s Constraints

We also apply the user’s modifications to the HSMM as constraints using two approaches during the forward path in the Viterbi algorithm. The first approach is to simply reduce the likelihoods of the undesired phases directly. For example, in Figure 6:Right, it is undesirable to pass through the phase states other than the beat head states at the destination frame where the user has moved the beat. Therefore, we prevent such undesired paths by reducing the likelihoods on the undesired grids (the blue circles).

The second approach is to weight the likelihoods at each beat-head state, where the tempo transition occurs. We apply a similar idea to that in §.5.1. First, we prepare a weighting array $U \in \mathbb{R}^T$ and initialize it by $\mathbf{1}$. We set $U_t = 1 + (1 - |t - b_i|/W)$ for the frames near the edited beat and its neighbors, and $U_t = 1 - (1 - |t - b_i^\pm|/W)$ for the frames around their midpoints. Then, our formulation of the integrated likelihood at the beat-head state h^i of the tempo i at the frame t becomes

$$G(t, h^i) = \max_j \{ G(t-1, e^j) + F(t, j, i) P_{t-n_j} U_{t-n_j} \Gamma_{j,i} P_t U_t \}, \quad (5)$$

where e^j denotes the phase state just before the beat head of the tempo j . $\Gamma_{j,i}$ and P_t are the transition probabilities from tempo j to i , and the observable state (the outputs of the DNN). n_j is the number of phase divisions of tempo j . $F(t, j, i)$ constrains the tempo range and flexibility that are set by the tempo-setting tools in §.3. If the tempo i exceeds the upper tempo limit or falls the lower tempo limit, $F(t, j, i) = 0$; otherwise, $F(t, j, i) = 1$. In addition, if $n_i/n_j > flex_t$ or $n_i/n_j < 1/flux_t$, $F(t, j, i) = 0$; otherwise, $F(t, j, i) = 1$, where $flux_t \in [0, 1]$, is the flexibility of the tempo change (the slider value).

6. EVALUATION

6.1 Validation through Simulation

We first conducted a validation using a simulation environment. We compared the results with and without ARSA. This comparative method without ARSA is equivalent to fine-tuning, which corresponds to Nakano et al. [18]. In this experiment, we used the SMC MIREX dataset [39], which includes 217 musical pieces of various genres with annotations of the beat positions. The lengths of all the pieces were aligned to a length of 40 seconds. We used a laptop PC (CPU: 2.9 GHz 6-Core Intel Core i9, RAM 32GB, GPU: Radeon Pro Vega 20) for equipment. We implemented all the our system including the DNN in C++.

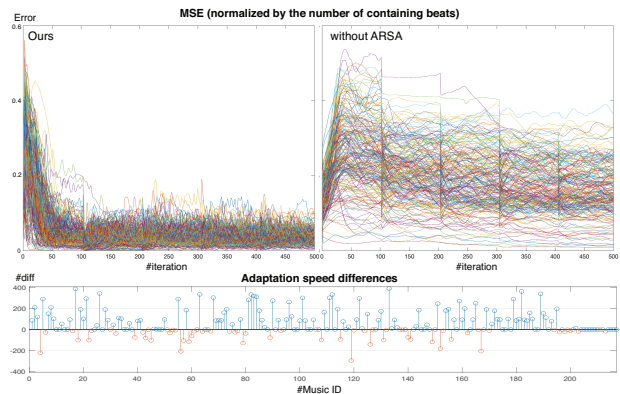


Figure 8. The result in a simulated experiment. The top row shows the transitions of the MSE losses throughout the entire musics. The bottom row shows a comparison of how quickly the adaptation converged in both methods.

The experimental procedure was as follows. We first found the three adjacent beats that have the highest errors (MSE loss), and automatically corrected them (treating three adjacent beats as a single unit, as described in §5.1). The system then iterated the adaptation iterations 100 times. The system repeated this procedure 5 times. We then obtained totally 5 error corrections and 500 adaptation iterations for each piece. For ARSA label annotations, we automatically assigned a positive label if the recommended beat position was sufficiently close to the correct beat annotation, and otherwise assigned a negative label. We added five labels for each error correction. Following to Dixon [40], we considered that an analyzed beat is accurate if it falls within a ± 70 ms tolerance window around the correct position. For adaptation, we updated the parameters in the second block and the ARSA while keeping that of the first block in Figure 6:Left. We found this achieves a better balance between the computational cost and the adaptation ability in our experiments.

Figure 8 shows the result. The top row indicates the transitions of the mean squared errors (MSE loss) throughout the entire piece of music. We note that these errors are normalized by the number of beats contained in each piece. Clearly, we can see that the errors in our method tend to converge to small values, and that our method has an ability to adapt throughout the entire song from only the local corrections. By contrast, with the comparative method, although the error itself tends to be reduced, but it does not converge well, and we can see that it struggles to adapt to the entire piece of music. The bottom row in figure 8 shows a comparison of how quickly the adaptation progresses with both methods. Suppose that the number of iterations when our method reaches an F1-measure of over 0.8 for the first time is N^A , and for the comparative method is N^B , the figure plots the difference, $N^B - N^A$. Then, a larger the positive value indicates that our method reaches the F1-measure faster than the existing method (blue), whereas a smaller negative value indicates that it achieves it at a slower rate (red). Statistically, in the 156/217 pieces ($p < 0.05$ by chi-squared test), our method reached to the goal significantly faster. For that computational cost, this simulation experiment (a total of

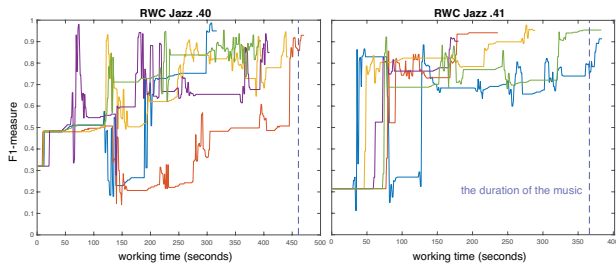


Figure 9. Improvements of the scores by 5 participants.

217 musical pieces) took 5.5 hours for our method and 5 hours for the comparative method. This approximately 8 seconds increases in cost per music on average is a negligible compared to the actual time the user requires for a manual correction in practical.

6.2 User Study

We conducted a user study to evaluate how the proposed interface improves the efficiency of the correction. We measured the working time for correcting the errors contained in the test target music. We set the baseline of this working time as the duration of the music. The reasons for this are as follows: Naturally, the fastest way to annotate the beats is to accurately tap throughout the entire piece of music if it is possible (although it is hard). The ideal working time for this method is equal to the duration. Therefore, our experiment determines whether our interface can reduce the working time to less than this baseline.

We hired 5 participants, all skilled users of sound editing software (e.g., Cubase [41] and Audacity [42]; DAW). The experiment consisted of three parts. The first part was to practice using our interface (45 minutes). We described how to use the interface, and each subject trained the system. The second part was a time trial (45 minutes). Each subject was asked to correct as quickly as possible the errors of the beats in the two pieces of target music using our interface. For the targets, we selected 2 pieces from the RWC Jazz dataset [34]. The first one is No.40, which has a duration of 7 minutes and 41seconds, and contains 481 beats, and the second one is No.41, which has a duration of 6 minutes and 6 seconds, and contains 637 beats. Both pieces were selected because they have appropriate durations and contain many errors from the initial analysis that are impractical to fix manually. Finally, the last part was a survey interview about their experience (10 minutes).

In the second part, each participant processed two pieces of target music. To avoid different interpretations of the beat by different people, we prepared an “answer” music file containing click sounds. We first asked the participants to listen to this answer entirely once through. Next, we asked the subjects to correct the beats as quickly as possible. We monitored the working times and progress of the F1-measure. When the F1 reached above 0.9 stably, we stopped the task even if they had not yet checked the entire piece of musics. Although each subject was allowed to use all tools prepared in §.3, however, we prohibited the use of only the tap tool for more than 5 seconds in a row. We also asked them to press the Re-Opt button as much as possible after each operation.

Figure 9 shows the result of improvements of the F1-measure (vertical axis) for each participants with the passage of the working time (horizontal axis). The vertical dotted lines represents the duration of each music. As the results show, 4/5 and 3/5 of the subjects completed the task in less time than the durations for pieces No. 40 and No. 41, respectively, and the remaining subjects also completed the task at slightly after the target duration. This means that most of the subjects completed the task without realizing it, despite not listening to the entire piece of music. Of course, when we really use our interface, we have to check the entire piece of music at the end. Therefore, the working time can not be less than the duration of the music piece. Therefore, albeit conditionally, we can conclude that our interface improves the efficiency of the error corrections.

Finally, we conducted an interview with each participant. We prepared two predetermined questions. The first question is “*Would you like to use this interface if it is implemented on DAW?*”. For this question, all the participants answered “*Yes*”. The second question is “*Did you have any problems in using the system?*”. For this question, some participants commented that **A**, they were confused about which tool to use for their facing beat error because the system provides many options for modification. Other participants also mentioned that **B**, they were unsure how soon the optimization loop should be stopped.

For the comment **A**, it is expected to be alleviated as the user becomes more proficient with the interface. However, as a better solution, the system could understand the current situation and recommend the most effective tool. This remains an area for a future work. For the comment **B**, because the concept of optimization through iterations is difficult for the common users to understand, we would need to find a criterion for the system to automatically terminate the iteration. However, because our adaptation process does not uniquely converge, it is difficult to find such a criterion. Therefore, this is a challenge for the future. As a similar problem, because of the iterative adaptation, the overall error rate frequently increases even if the user has corrected the error. This is not intuitive and is disconcerting for the user and should be improved.

7. CONCLUSION

In this paper, we proposed an interactive beat-tracking interface for adapting to a specific user and a targeted piece of music using a human-in-the-loop approach. To achieve this, we introduced a novel adaptive runtime self-attention that achieves a better balance between the lower computational cost during training and the high runtime adaptation ability that distributes the local modifications by the user to throughout the entire input sequence globally. We validated the feasibility and effectiveness of our method through several experiments, including a user study with the potential users of our interface. Beyond beat tracking, by training the machine learning model using only a local context and adapting it to a specific target using the global context at runtime, our method is expected to be useful for other broad domains such as chord recognition, singing voice synthesis, and sound source separation.

8. REFERENCES

- [1] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," *ACM Multimedia (ACMMM)*, 1994.
- [2] Y. Ueda, Y. Uchiyama, T. Nishimoto, N. Ono, and S. Sagayama, "HMM-based approach for automatic chord detection using refined acoustic features," *International Society for Music Information Retrieval (ISMIR)*, 2010.
- [3] E. J. Humphrey and J. P. Bello, "Four timely insight on automatic chord estimation," *International Society for Music Information Retrieval (ISMIR)*, 2015.
- [4] G. Shibata, R. Nishikimi, and K. Yoshi, "Music structure analysis based on an lstm hsmm hybrid model," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [5] M. E. P. Davies and S. Bock, "Temporal convolutional networks for musical audio beat tracking," *European Signal Processing Conference (EUSIPCO)*, 2019.
- [6] S. Bock and M. Schedl, "Enhanced beat tracking with context-aware neural networks," *International Conference on Digital Audio Effects (DAFx)*, 2011.
- [7] S. Bock, M. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," *International Society for Music Information Retrieval (ISMIR)*, 2016.
- [8] S. Bock, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," *International Society for Music Information Retrieval (ISMIR)*, 2014.
- [9] S. Bock and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," *International Society for Music Information Retrieval (ISMIR)*, 2020.
- [10] B. D. Giorgi, M. Mauch, and M. Levy, "Downbeat tracking with tempo-invariant convolutional neural networks," *International Society for Music Information Retrieval (ISMIR)*, 2020.
- [11] S. Reich, "Clapping music," 1972.
- [12] —, "Piano phase," 1967.
- [13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [14] F. Krebs, S. Bock, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," *International Society for Music Information Retrieval (ISMIR)*, 2015.
- [15] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," *ACM Multimedia (ACMMM)*, 1910.
- [16] S. Dixon, "An interactive beat tracking and visualisation system," *International Computer Music Conference (ICMC)*, 2001.
- [17] J. Driedger and M. M. Hendrik Schreiberand W. Haas, "Towards automatically correcting tapped beat annotations for music recordings," *International Society for Music Information Retrieval (ISMIR)*, 2019.
- [18] T. Nakano, Y. Koyama, M. Hamasaki, and M. Goto, "Interactive deep singing-voice separation based on human-in-the-loop adaptation," *Intelligent User Interface (IUI)*, 2020.
- [19] M. Goto, K. Yoshii, H. Fujihara, M. Mauch, and T. Nakano, "Songle: A web service for active music listening improved by user contributions," *International Society for Music Information Retrieval (ISMIR)*, 2011.
- [20] K. Fujii, Y. Saito, S. Takamichi, Y. Baba, and H. Saruwatari, "Humangan: Generative adversarial network with human-based discriminator and its evaluation in speech perception modeling," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
- [21] N. J. Bryan and G. J. Mysore, "Interactive refinement of supervised and semi-supervised sound source separation estimates," *IEEE Acoustics, Speech, and Signal Processing*, 2013.
- [22] T. Bazin, G. Hadjeres, P. Esling, and M. Malt, "Spectrogram inpainting for interactive generation of instrument sounds," *Joint Conference on AI Music Creativity*, 2020.
- [23] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *Advances in Neural Information Processing Systems (NeurIPS)*, no. 32, 2019.
- [24] Y. Zhou, Y. Koyama, M. Goto, and T. Igarashi, "Interactive exploration-exploitation balancing for generative melody," *Intelligent User Interface (IUI)*, 2021.
- [25] B. Kim and B. Pardo, "A human-in-the-loop system for sound event detection and annotation," *Intelligent User Interface (IUI)*, 2018.
- [26] J. C. Brown, "Calculation of a constant q spectral transform," *Journal of Acoustics in America*, 1991.
- [27] R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia, "Chord recognition using duration-explicit hidden markov models," *International Society for Music Information Retrieval (ISMIR)*, 2012.

- [28] K. Shibata, R. Nishikimi, S. Fukayama, M. Goto, E. Nakamura, K. Itoyama, and K. Yoshii, “Joint transcription of lead, base, and rhythm guters based on a factorial hidden semi-markov model,” *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.
- [29] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “A. van den oord and s. dieleman and h. zen and k. simonyan and o. vinyals and a. graves and n. kalchbrenner and a. senior and k. kavukcuogluwavenet: A generative model for raw audio,” *CoRR abs/1609.03499*, 2016.
- [30] D. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and accurate deep network learning by exponential linear units (elus),” *International Conference on Learning Representations*, 2016.
- [31] G. Tzanetakis and P. Coo, “Musical genre classification of audio signals,” *IEEE Transactions on Audio and Speech Processing*, 2002.
- [32] U. Marchand, Q. Fresnel, and G. Peeters, “Gtzan-rhythm: Extending the gtzan test-set with beat, down-beat and swing annotations,” *International Society for Music Information Retrieval (ISMIR)*, 2015.
- [33] U. Marchand and G. Peeter, “Swing ratio estimation,” *International Conference on Digital Audio Effects (DAFx)*, 2015.
- [34] M. Goto, “Masataka goto and hiroki hashiguchi and takuichi nishimura and ryuichi oka,” *International Society for Music Information Retrieval (ISMIR)*, 2002.
- [35] —, “Aist annotation for the rwc music database,” *International Society for Music Information Retrieval (ISMIR)*, 2006.
- [36] Amazon, “Amazon music,” <https://music.amazon.co.jp>.
- [37] J. B. Diederik P. Kingma, “Adam: A method for stochastic optimization,” *CoRR abs/1412.6980*, 2014.
- [38] E. Wu and A. Zhang, “A feature re-weighting approach for relevance feedback in image retrieval,” *International Conference on Image Processing*, 2002.
- [39] A. Holzapfel, M. E. P. Davies, J. R. Zapata, and J. L. Oliveira, “Selective sampling for beat tracking evaluation,” *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.
- [40] S. Dixon, “Evaluation of the audiobbbeat ttracking system beatroot,” *New Music Res*, 2007.
- [41] Steinberg, “Cubase,” <https://new.steinberg.net/cubase/>.
- [42] Audacity, “Audacity,” <https://www.audacityteam.org>.

COMPOSER CLASSIFICATION WITH CROSS-MODAL TRANSFER LEARNING AND MUSICALLY-INFORMED AUGMENTATION

Daniel Yang TJ Tsai
Harvey Mudd College
dhyang, ttsai@hmc.edu

ABSTRACT

This paper studies composer style classification of piano sheet music, MIDI, and audio data. We expand upon previous work in three ways. First, we explore several musically motivated data augmentation schemes based on pitch-shifting and random removal of individual notes or groups of notes. We show that these augmentation schemes lead to dramatic improvements in model performance, of a magnitude that exceeds the benefit of pretraining on all solo piano sheet music images in IMSLP. Second, we describe a way to modify previous models in order to enable cross-model transfer learning, in which a model trained entirely on sheet music can be used to perform composer classification of audio or MIDI data. Third, we explore the performance of trained models in a 1-shot learning context, in which the model performs classification among a set of composers that are unseen in training. Our results indicate that models learn a representation of style that generalizes beyond the set of composers used in training.

1. INTRODUCTION

This paper studies composer style classification based on sheet music, audio, and MIDI data. Given a previously unseen page of sheet music or fragment of audio/MIDI, the goal is to predict which one of a fixed set of C composers composed it based on its compositional style.


Previous works on composer classification generally fall into one of three categories. The first category of approaches extract manually designed features from the data and feed them to a classifier. Some features that have been explored include chroma [1, 2], expert musicological features [3–5], musical intervals or counterpoint characteristics [6, 7], piece-level statistics or features describing piece structure [2, 8], and pre-defined feature sets like the jSymbolic toolbox [9, 10]. Many standard classification algorithms have been used, such as decision trees (e.g. [7, 9]), KNN (e.g. [11]), logistic regression (e.g. [5]), SVMs (e.g. [3, 9]), and neural networks (e.g. [2, 12]). The second category of approaches train one sequence-based model for each composer and then select the model that has

the highest likelihood for a given query sequence. These sequence-based models include n-gram language models [8, 13–15] and several variants of Markov models [16, 17]. These models are typically fed with a very low-level representation of the data, such as sequences of note values or intervals between consecutive notes. The third category of approaches train a neural network classifier in an end-to-end fashion. Rather than relying on manually designed features, this approach tries to automatically learn a suitable feature representation from the raw data that is effective for classification. This paradigm was explored early on in [18], and many recent works have focused on convolutional neural networks trained on piano roll-like data [19–21].

The above works generally assume that the data is available in a symbolic format such as MIDI, MusicXML, or `**kern`. Recent works [22, 23] have explored the composer classification task based on raw sheet music images. While this makes the problem more challenging due to the high-dimensional nature of images, it also provides a very distinct advantage: there is an enormous amount of sheet music data available through the International Music Score Library Project (IMSLP).¹ These works first convert each sheet music image into a sequence of musical words based on the bootleg score feature representation [24], and then treat the problem as one of text classification. They incorporate best practices from natural language processing, such as pretraining a language model on a large set of unlabeled data, and then finetuning a classification model based on a small set of labeled data.

This paper expands upon [22] in three different directions.² First, we explore several different forms of data augmentation for the bootleg score representation. These include pitch shifting of noteheads relative to the staff lines and several different forms of dropout that are musically motivated. Second, we explore cross-modal transfer learning, in which a model is first trained entirely on sheet music, and then used for composer classification of audio and MIDI data. Third, we explore the performance of trained models in a 1-shot learning context, in which the goal is to perform classification among a set of unseen composers given only one representative piece from each composer.

This paper has three main contributions, which correspond directly with the three directions above.

 © D. Yang, T. Tsai. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Yang, T. Tsai, “Composer Classification With Cross-Modal Transfer Learning and Musically-Informed Augmentation”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ <https://imslp.org>

² Code can be found at <https://github.com/HMC-MIR/ComposerID>.

- We propose several forms of data augmentation for the bootleg score representation and evaluate their impact on the composer classification task. We demonstrate enormous performance improvements (37.3% to 63.9% accuracy) which are even larger than the benefit of pretraining on all IMSLP piano sheet music (37.3% to 57.5%). Both can be combined to achieve even greater gains (89.0%). We share our optimal settings and selection of augmentation methods, which may be useful for tasks using symbolic data or any piano roll-like representation.
- We successfully demonstrate cross-modal transfer learning. By making a minor modification to the bootleg score representation, we show that it is possible to train a model on sheet music and use it for composer classification of audio and MIDI data. This approach may benefit tasks where the amount of data in one modality is limited or restricted (e.g. due to copyright issues).
- We evaluate the performance of trained models in a 1-shot learning context, in which we use the model’s penultimate layer activations as a feature embedding. Our results strongly indicate that the models are learning a more generalizable notion of compositional style that extends beyond the composers in training.

We describe each of the three new directions in detail in the next three sections.

2. DATA AUGMENTATION METHODS

This section explores several data augmentation schemes for our task. The next five subsections describe the feature representation (Section 2.1), proposed augmentation strategies (Section 2.2), experimental setup (Section 2.3), experimental results (Section 2.4), and analyses (Section 2.5).

2.1 Feature Representation

We first describe how sheet music is converted into a sequence of words as proposed in [22]. This forms the backdrop for our proposed augmentation strategies. This process consists of two steps.

The first step is to compute a bootleg score representation of the sheet music image. The bootleg score is a mid-level feature representation that encodes the position of filled noteheads relative to the staff lines in piano sheet music [24]. The bootleg score itself is a $62 \times N$ binary matrix, where 62 indicates the total number of distinct staff line positions in both the left and right hand staves and where N indicates the number of grouped note events (e.g. a chord containing four notes played simultaneously would constitute a single grouped note event). This representation discards a significant amount of information in the sheet music, such as non-filled noteheads, time signature, key signature, accidentals, note duration, measure boundaries, octave markings, and clef changes.

Nonetheless, it has been shown to be useful for a variety of tasks involving sheet music such as sheet-MIDI passage retrieval [24], audio-sheet music synchronization [25], sheet music identification [26], and finding matches between the Lakh MIDI dataset and IMSLP [27].

The second step is to tokenize the bootleg score. This is done in two different ways, depending on if the language model architecture is word-based or subword-based. For word-based models (e.g. AWD-LSTM [28]), each column of the bootleg score (62 bits) is represented as a single 64-bit integer and interpreted as a word. For subword-based models (e.g. GPT-2 [29], RoBERTa [30]), each column is represented as a sequence of four 8-bit characters so that the bootleg score can be expressed as a length $4N$ sequence of 8-bit characters. Based on a training set of character sequences, a byte pair encoder (BPE) [31] can be trained in an unsupervised fashion to learn a vocabulary of common subwords. The trained BPE can then be used to tokenize the length $4N$ sequence of characters into a sequence of subwords. The result of the tokenization step is a sequence of words or subwords that are fed to the language model.

2.2 Proposed Augmentation Strategies

We explore two different types of data augmentation for the bootleg score representation. These strategies could be applied to symbolic data or piano roll representations as well.

The first type of data augmentation is based on pitch shifting noteheads in the bootleg score. Similar to other musically informed representations like chroma and CQT, shifts in the bootleg score correspond to transpositions in key. We consider pitch shifts up to $\pm K$ positions, where K is a hyperparameter. When a pitch-shifted notehead falls off the edge of the bootleg score (i.e. the top or bottom of the left hand or right hand staff), the notehead is simply removed. Note that a value of K will result in $2K + 1$ times as much data as the original dataset. We consider pitch-shifting at both training time as well as at test time. For the latter, we pitch shift a query bootleg score by up to $\pm K$ positions, pass all $2K + 1$ bootleg scores through the trained model, and average the predictions to generate a single ensemble prediction.

The second type of data augmentation is based on removing noteheads in the bootleg score. This strategy is based on the simple observation that randomly adding a note to a column of the bootleg score is unlikely to yield a musically plausible event, whereas randomly *removing* one or more notes from a column of the bootleg score is likely to yield a musically plausible event. For example, consider a column in the bootleg score that contains octaves in the left hand and a chord in the right hand. If a single note or even an entire hand is removed, the result is still a musically plausible event. We consider three different types of removal: randomly dropping each individual notehead with some probability, randomly dropping an entire hand (i.e. all noteheads in the left or right hand staff) within a single bootleg score column, or randomly drop-

ping an entire column of the bootleg score. Since these methods correspond to applying a form of dropout regularization directly to the bootleg score representation, we refer to these three types of removal as DropNote, DropHand, and DropColumn regularization, respectively.

2.3 Experimental Setup

We use the same experimental setup as [22] for fair comparison. A brief summary is provided below for completeness.

The unlabeled data for language model pretraining consists of all solo piano sheet music images in IMSLP. It contains 29,310 PDFs, 255,539 pages, and 48.5 million bootleg score features. We used the precomputed bootleg score features provided in [26]. We will refer to this unlabeled dataset as the IMSLP data. During language model pretraining, 90% of the data is used for training and 10% for validation.

The labeled dataset for the composer classification task is a carefully curated subset of the IMSLP data. It contains one representative sheet music version from every solo piano piece in IMSLP from nine different classical music composers: Bach, Beethoven, Chopin, Haydn, Liszt, Mozart, Schubert, Schumann, and Scriabin. These PDFs were manually filtered to remove filler pages like title page, foreword, etc. The resulting set contains 787 PDFs, 7,151 pages, and 1.47 million bootleg score features. The labeled data is split by piece into training (4347 pages), validation (1500 pages), and test (1304 pages) sets.

The labeled dataset was further preprocessed to form a fragment dataset in order to solve two problems: too little data (only 4347 training images) and significant class imbalance. A fixed number of bootleg score fragments of length 64 were randomly sampled from each composer. The resulting fragment dataset contains 32400, 10800, and 10800 fragments for training, validation, and test, respectively.

We report classification results on both the fragment classification task and the full page classification task. The fragment dataset is used for training all models, since it has more training samples and is class-balanced. When evaluating a model on the full page task, fragments of length 64 are taken from the query bootleg score with 50% overlap, all fragments are passed through the fragment classification model, and the predictions are averaged to form a single prediction for the entire page. We report results in terms of classification accuracy for the fragment classification task and macro F1 for the full page classification task (since the classes are imbalanced with pages).

2.4 Results

Figure 1 compares the performance of several classification models on the fragment composer classification task (left) and full page classification task (right). The results without data augmentation are shown as black horizontal lines, and the results with optimal data augmentation settings (described in Section 2.5) are shown as colored bars. Note that the results without data augmentation correspond

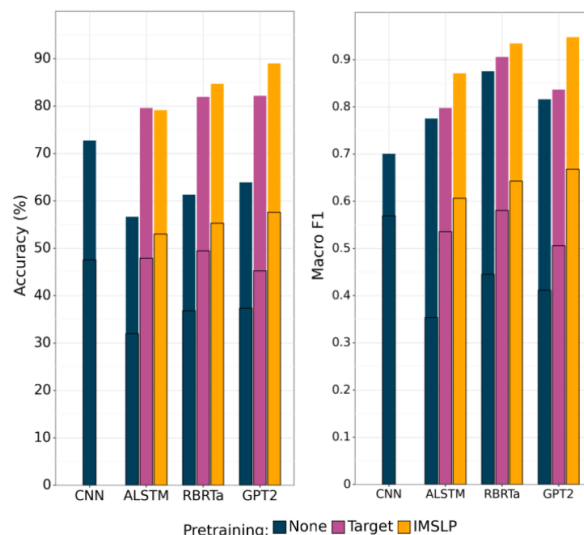


Figure 1. Results for sheet music composer classification of fragments (left) and full pages (right). The colored bars show performance with optimal data augmentation settings, and the horizontal black lines show performance without any data augmentation.

to the results reported in [22]. Results are reported for four different classification models and across three different pretraining conditions. The four model architectures are a CNN model (based on [19]) that has two convolutional layers followed by global pooling across the time dimension and a final output layer, AWD-LSTM [28], RoBERTa [30], and GPT-2 [29]. The three pretraining conditions are: (a) no pretraining, in which models are trained from scratch only on the labeled fragment dataset, (b) target pretraining, in which language models are pretrained on the labeled data and then finetuned for the classification task, and (c) IMSLP pretraining, in which we pretrain the language models on the IMSLP data, finetune the language model on the labeled data, and then finetune the classifier on the labeled fragment dataset.

There are a few things to notice about Figure 1. Across all models and all pretraining conditions, there is an extremely large benefit to using data augmentation. In all cases, the benefit of data augmentation is larger than the benefit of pretraining. For example, for the RoBERTa model, data augmentation improves performance on the full page classification task from 0.44 to 0.88 macro F1 (without any pretraining), while pretraining on IMSLP improves performance from 0.44 to 0.64 (without any data augmentation). When both data augmentation and pretraining are combined, the benefit is enormous: the RoBERTa model increases from 36.8% accuracy to 84.7% and from 0.44 macro F1 to 0.93.

2.5 Analysis

Figure 2 shows the benefit of adding a single type of training data augmentation in isolation. The results on the fragment classification task on the validation set are shown at left, and the results on the full page classifica-

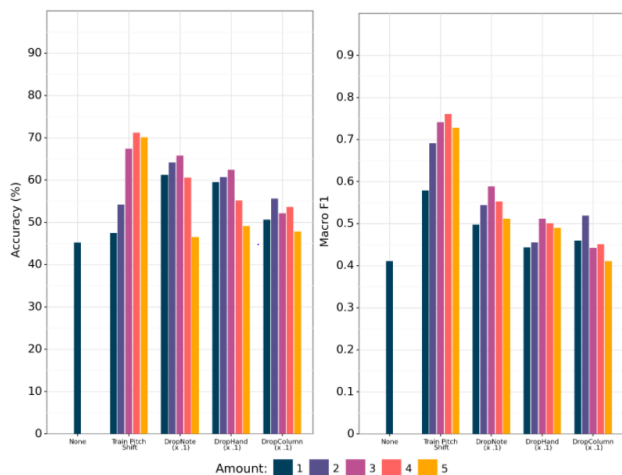


Figure 2. Effect of adding a single type of data augmentation in isolation to the GPT-2 model. Individual bars within each group show the effect of different hyperparameter settings. The leftmost standalone bar shows the performance without data augmentation for comparison.

tion task are shown at right. Within each figure, the leftmost (standalone) bar shows the performance of the GPT-2 model without data augmentation for reference. The four groups of bars correspond to four different types of training data augmentation: pitch shifting, DropNote, DropHand, and DropColumn. Within each group, individual bars show the performance with different hyperparameters settings (e.g. $K = 1, 2, 3, 4, 5$ for pitch shifting and $p = 0.1, 0.2, 0.3, 0.4, 0.5$ for the dropout variants). We can see that pitch shifting seems to be the most effective form of data augmentation, followed by DropNote, DropHand, and then DropColumn. The optimal amount of pitch shifting seems to be $K = 4$, above which the results get slightly worse. Most likely this is because large pitch shifts result in many noteheads simply being removed from the bootleg score canvas.

Figure 3 shows the benefit of adding various forms of data augmentation cumulatively. Again the leftmost (standalone) bar shows the performance without data augmentation. The first (leftmost) group of bars shows the performance with only training pitch shift augmentation. These results are identical to those shown in Figure 2. The second group of bars shows the performance with training pitch shift augmentation ($K = 4$) and DropNote with various values of p . Each successive group adds an additional form of augmentation with the optimal settings of previous augmentation types. This figure allows us to see the cumulative benefit of adding multiple forms of data augmentation. We see dramatic improvements from adding the most effective forms of augmentation, and modest but nontrivial improvements after that. The optimal settings are training pitch shifting with $K = 4$, DropNote with $p = 0.3$, DropHand with $p = 0.3$, DropColumn with $p = 0.3$ and test-time pitch shifting with $K = 4$. These are the settings used in the results shown in Figure 1.

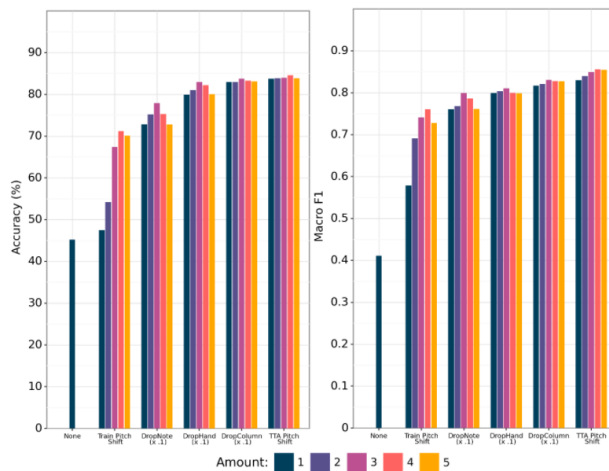


Figure 3. Effect of adding each type of data augmentation cumulatively to the GPT-2 model. Each group uses the optimal settings from previous augmentation types.

3. CROSS-MODAL TRANSFER LEARNING

This section describes our exploration into cross-modal transfer learning, in which a model trained entirely on sheet music is used to perform composer classification of audio and MIDI data. In the next three subsections, we describe the methodology (Section 3.1), experimental setup (Section 3.2), and experimental results (Section 3.3).

3.1 Methodology

The key to cross-modal transfer learning is representing audio, MIDI, and sheet music in a common feature space. That feature space is a modified bootleg score representation. Below, we describe a way to bridge the gap between MIDI and sheet music using this modified bootleg score representation. For audio performances of piano music, we first apply an automatic music transcription (AMT) system [32], and then follow the procedure below.

We can extract a bootleg score from MIDI by mapping MIDI note onset events to staff line positions in sheet music using the conventions of Western musical notation. However, there are two obstacles that prevent a MIDI-generated bootleg score and a sheet music-generated bootleg score from being directly comparable. First, there is ambiguity about left/right hand attribution. For example, if a C4 note onset occurs in a MIDI file, it could appear in the left hand staff (one ledger line above the topmost staff line) or the right hand staff (one ledger line below the bottom staff line). In the sheet music, it will only appear in one of these locations. Second, there is ambiguity about enharmonic representations. For example, a MIDI note number 61 could appear in the sheet music as a C-sharp or a D-flat, and these correspond to two different staff line positions.

These two ambiguities can be resolved in different ways. To handle the ambiguity due to left/right hand attribution, we can simply place noteheads in the middle register in *both* the left hand and right hand staves. For example, if the sheet music contains a notehead one ledger line

below the right hand staff (i.e. C4 in treble clef), an additional notehead will be placed one ledger line above the left hand staff (i.e. C4 in bass clef). Likewise, a MIDI note onset at C4 will result in two noteheads at the same two locations in the MIDI-generated bootleg score. By making this modification to the bootleg score representation, the MIDI-generated bootleg score and sheet music-generated bootleg score will match. To handle the ambiguity due to enharmonic representations, we can generate two different versions of the MIDI bootleg score: one in which all black keys on the piano are interpreted as sharps, and one in which all black keys on the piano are interpreted as flats. We can then pass both versions of the MIDI bootleg score through our classification model and average the resulting predictions. This method for bridging the gap between MIDI and sheet music was first proposed in [27] for a MIDI-sheet retrieval task. Here, we use the same technique for cross-modal transfer learning in composer classification.

Cross-modal transfer learning thus requires two changes to the system described in Section 2. The first change is to use the modified bootleg score representation when converting sheet music to a sequence of words. The models are otherwise trained exactly as before. The second change is to consider both sharp and flat versions of the MIDI-generated bootleg score during inference, and to average the model’s predictions from both.

3.2 Experimental Setup

In order to assess the effectiveness of cross-modal transfer learning, we need to introduce additional datasets of MIDI and audio for the composer classification task. These datasets are derived from the MAESTRO dataset [33], which contains MIDI and audio files of real piano performances. We preprocess the dataset in the following manner. First, we take all MIDI performances of pieces composed by the same nine composers in our labeled sheet music dataset. Because some pieces are performed many times, we take one representative performance for each piece to avoid overemphasizing a small set of popular pieces. Second, we randomly sample $X = 5000$ fragments of length Y seconds from each composer, spread evenly across the composer’s pieces. This sampling strategy produces a dataset of fragments that is class-balanced, and it allows us to study the effect of fragment length on model accuracy. Because we are not doing any additional training or finetuning, we use all of the MIDI data as a test set. This constitutes our MIDI fragment composer classification dataset. A corresponding audio fragment composer classification dataset can be generated using the same methodology.

Note that the above dataset contains many pieces that were in the labeled sheet music dataset, albeit in a different modality. To further study the generalizability of our trained models, we constructed two different versions of the audio/MIDI fragment classification datasets: one in which all of the data is present, and another in which pieces that were in the labeled sheet music training data

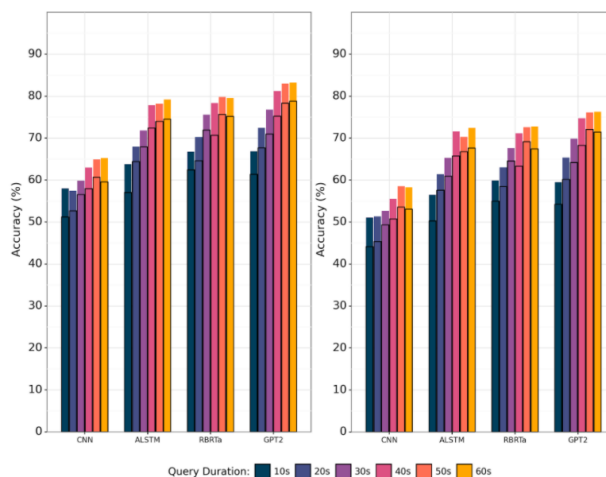


Figure 4. Results for composer classification of MIDI (colored bars) and audio (horizontal black bars) on the full (left plot) and reduced (right plot) datasets. The results are with a GPT-2 model trained only on sheet music.

are excluded. Because the latter has many fewer pieces, we reduce the number of fragments per composer to $X = 500$. The number of pieces for each composer in the full/reduced datasets are: Bach – 101/27, Beethoven 87/21, Chopin – 101/31, Haydn – 29/13, Liszt – 108/25, Mozart – 36/12, Schubert – 103/33, Schumann – 32/8, Scriabin – 25/1. In total, there are 622 pieces in the full dataset and 171 pieces in the reduced dataset.

3.3 Results

Figure 4 compares the performance of four different models on the MIDI and audio fragment classification tasks. The left plot shows performance on the dataset containing all pieces, and the right plot shows performance on the dataset that excludes pieces in the sheet music training dataset. The performance on the MIDI fragment classification task is shown by colored bars, and the performance on the audio fragment classification task is indicated with horizontal black lines. The four models shown are the best version of each model architecture from Figure 1: the CNN, AWD-LSTM, RoBERTa, and GPT-2 models with optimal data augmentation settings and IMSLP pretraining (for the language models). The four groups of bars in each plot correspond to the four different models. Within each group, the individual bars show the model performance for different durations of the audio/MIDI query. Similar to the full page sheet music classification task, we convert each audio/MIDI query to a bootleg score, take fragments of length 64 with 50% overlap, pass each fragment through the classification model, and average the predictions from all fragments. When processing audio queries, the Onsets & Frames AMT system [32] is used to convert the audio to an estimated MIDI representation.

There are four things to notice about Figure 4. First, the results do clearly demonstrate effective cross-modal transfer learning. Because the datasets are balanced by class,

random guessing would correspond to an accuracy of 11%. In contrast, the GPT-2 model is predicting the correct composer of 60-second MIDI fragments 83% and 76% of the time on the two versions of the MIDI classification data. This shows that we can train a model on sheet music data, and then use it to classify MIDI and audio data without any additional training or finetuning. Second, there is a 6-8% difference in accuracy between the two versions of datasets (i.e. comparing the left plot to the right plot). This reflects the benefit of having seen a piece before in training in a different modality. But even when a piece has never been seen before – in any modality – the results in the rightmost plot show that the models still perform well. Third, there is a 4-6% difference in accuracy between the MIDI classification task and the audio classification task (i.e. comparing colored bars to the black horizontal lines). This gap comes from failures in the AMT system when converting from audio to MIDI. Fourth, the query duration strongly affects model performance for shorter queries, but plateaus at a duration of about 50 seconds (i.e. comparing individual bars within each group). This suggests that 40-50 seconds is enough context to recognize the style of a piece, and that using more context beyond that is unlikely to help much.

4. ONE-SHOT LEARNING

This section describes our exploration of using trained models in a 1-shot learning context, in which the model is expected to classify pieces from a set of C unseen composers given only a single representative piece from each composer. In the next three subsections, we describe our methodology (Section 4.1), experimental setup (Section 4.2), and experimental results (Section 4.3).

4.1 Methodology

For 1-shot learning, we use our trained classification models as a feature extractor that projects sheet music into an embedding space that captures some notion of compositional style. We take the penultimate layer activations of the model as our feature representation. When processing a MIDI or audio performance, we first compute the bootleg score (using AMT to convert the audio to MIDI, if necessary), take multiple bootleg score fragments of length 64 with 50% overlap, and then extract the embedding features from each fragment. On training data, each fragment’s embedding serves as a single sample in our KNN database. For a given test bootleg score fragment, we find the $K = 3$ closest samples in Euclidean distance from each composer, and then rank composers by their average KNN distance.

4.2 Experimental Setup

Our data for 1-shot classification experiments also comes from the MAESTRO dataset. We exclude the original nine composers used in training and identify nine other composers with sufficient data: Rachmaninoff, Debussy, Scarlatti, Mendelssohn, Brahms, Mussorgsky, Tchaikovsky, Clementi, and Handel. We sample five pieces from each

Model	MIDI		Audio	
	Acc	Std	Acc	Std
Random	16.6%	2.2%	13.9%	2.0%
CNN	39.1%	2.1%	38.2%	3.7%
AWD-LSTM	45.5%	2.2%	42.8%	4.2%
RoBERTa	50.3%	3.1%	49.1%	3.4%
GPT-2	52.8%	3.9%	52.7%	3.1%

Table 1. Results for 1-shot learning experiments on composer classification of MIDI (left) and audio (right) fragments. The trained models are used to classify among $C = 9$ unseen composers given a single representative piece from each composer.

composer to ensure equal representation. For each episode, we randomly select one of the five pieces from each composer to serve as our training data, and use the remaining data for testing. Each test query is a single bootleg score fragment of length 64 extracted from one of the $9 \times 4 = 36$ test pieces (with 50% overlap between fragments). For each episode, we calculate the classification accuracy across the test queries. We ran 30 episodes with different train/test splits, and report the mean and standard deviation of the classification accuracy across the episodes.

4.3 Results

Table 1 shows the results of our 1-shot classification experiments. We report results with the best versions of each of the four model architectures. We also include the performance of a random guessing baseline as reference.

There are three things to notice about these results. First, all models perform much better than the random guessing baseline. This strongly indicates that the models are learning a more generalizable notion of compositional style that goes beyond the original nine composers in the training data. Second, we see the same relative ordering of performance as before: GPT-2 performs best, followed by RoBERTa, AWD-LSTM, and the CNN model. This suggests that better results on the original composer classification task are indicative of a more useful representation in the style embedding space. Third, we again observe a consistent difference in performance between 1-shot MIDI and 1-shot audio classification due to AMT errors.

5. CONCLUSION

This paper expands upon composer classification models in three ways. First, we propose several forms of data augmentation that lead to dramatic improvements in model performance. Second, we show that it is possible to modify previous models in order to enable cross-modal transfer learning, in which a model trained entirely on sheet music is used to perform composer classification on audio and MIDI data. Third, we show that trained models are effective in a 1-shot learning context, indicating that the models learn a representation of style that generalizes beyond the original composers used in training.

6. ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used for training the models.

7. REFERENCES

- [1] Y. Anan, K. Hatano, H. Bannai, M. Takeda, and K. Satoh, "Polyphonic music classification on symbolic data using dissimilarity functions," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2012, pp. 229–234.
- [2] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis, "Musical composer identification through probabilistic and feedforward neural networks," in *European Conference on the Applications of Evolutionary Computation*, 2010, pp. 411–420.
- [3] W. Herlands, R. Der, Y. Greenberg, and S. Levin, "A machine learning approach to musically meaningful homogeneous style classification," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.
- [4] E. Backer and P. van Kranenburg, "On musical stylometry—a pattern recognition approach," *Pattern Recognition Letters*, vol. 26, no. 3, pp. 299–309, 2005.
- [5] K. C. Kempfert and S. W. Wong, "Where does haydn end and mozart begin? composer classification of string quartets," *arXiv preprint arXiv:1809.05075*, 2018.
- [6] L. Mearns, D. Tidhar, and S. Dixon, "Characterisation of composer style using high-level musical features," in *Proc. of the 3rd International Workshop on Machine Learning and Music*, 2010, pp. 37–40.
- [7] P. Van Kranenburg and E. Backer, "Musical style recognition—a quantitative approach," in *Handbook of Pattern Recognition and Computer Vision*, 2005, pp. 583–600.
- [8] R. Hillewaere, B. Manderick, and D. Conklin, "String quartet classification with monophonic models," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 537–542.
- [9] D. Herremans, D. Martens, and K. Sörensen, "Composer classification models for music-theory building," in *Computational Music Analysis*, 2016, pp. 369–392.
- [10] C. McKay and I. Fujinaga, "jSymbolic: A feature extractor for midi files," in *Proc. of the International Computer Music Conference*, 2006.
- [11] A. Brinkman, D. Shanahan, and C. Sapp, "Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of josquin," in *Proc. of the Biennial International Conference on Music Perception and Cognition*, 2016, pp. 91–97.
- [12] P. Sadeghian, C. Wilson, S. Goeddel, and A. Olmsted, "Classification of music by composer using fuzzy min-max neural networks," in *Proc. of the 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 189–192.
- [13] M. Hontanilla, C. Pérez-Sancho, and J. M. Inesta, "Modeling musical style with language models for composer recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2013, pp. 740–748.
- [14] J. Wołkowicz and V. Kešelj, "Evaluation of n-gram-based classification approaches on classical music corpora," in *International Conference on Mathematics and Computation in Music*, 2013, pp. 213–225.
- [15] J. Wołkowicz, Z. Kulka, and V. Kešelj, "N-gram-based approach to composer recognition," *Archives of Acoustics*, vol. 33, no. 1, pp. 43–55, 2008.
- [16] M. A. Kaliakatsos-Papakostas, M. G. Epitropakis, and M. N. Vrahatis, "Weighted markov chain model for musical composer identification," in *European Conference on the Applications of Evolutionary Computation*, 2011, pp. 334–343.
- [17] E. Pollastri and G. Simoncelli, "Classification of melodies by composer with hidden markov models," in *Proc. of the First International Conference on WEB Delivering of Music*, 2001, pp. 88–95.
- [18] G. Buzzanca, "A supervised learning approach to musical style recognition," in *Proc. of the International Conference on Music and Artificial Intelligence (ICMAI)*, vol. 2002, 2002, p. 167.
- [19] H. Verma and J. Thickstun, "Convolutional composer classification," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 549–556.
- [20] G. Velarde, C. C. Chacón, D. Meredith, T. Weyde, and M. Grachten, "Convolution-based classification of audio and symbolic representations of music," *Journal of New Music Research*, vol. 47, no. 3, pp. 191–205, 2018.
- [21] G. Velarde, T. Weyde, C. E. C. Chacón, D. Meredith, and M. Grachten, "Composer recognition based on 2d-filtered piano-rolls," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 115–121.
- [22] T. Tsai and K. Ji, "Composer style classification of piano sheet music images using language model pretraining," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 176–183.
- [23] D. Yang, K. Ji, and T. Tsai, "A deeper look at sheet music composer classification using self-supervised pre-training," *Applied Sciences*, vol. 11, no. 4, p. 1387, 2021.

- [24] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, “MIDI passage retrieval using cell phone pictures of sheet music,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 916–923.
- [25] M. Shan and T. Tsai, “Improved handling of repeats and jumps in audio-sheet image synchronization,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 62–69.
- [26] D. Yang and T. Tsai, “Camera-based piano sheet music identification,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 481–488.
- [27] T. Tsai, “Towards linking the Lakh and IMSLP datasets,” in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 546–550.
- [28] S. Merity, N. S. Keskar, and R. Socher, “Regularizing and optimizing LSTM language models,” *arXiv preprint arXiv:1708.02182*, 2017.
- [29] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
- [30] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A robustly optimized BERT pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.
- [31] P. Gage, “A new algorithm for data compression,” *C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.
- [32] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 50–57.
- [33] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, “Enabling factorized piano music modeling and generation with the MAESTRO dataset,” in *International Conference on Learning Representations*, 2019.

ALIGNING UNSYNCHRONIZED PART RECORDINGS TO A FULL MIX USING ITERATIVE SUBTRACTIVE ALIGNMENT

Daniel Yang Kevin Ji TJ Tsai

Harvey Mudd College

dhyang, kji, ttsai@hmc.edu

ABSTRACT

This paper explores an application that would enable a group of musicians in quarantine to produce a performance of a chamber work by recording each part in isolation in a completely unsynchronized manner, and then generating a synchronized performance by aligning, time scale modifying, and mixing the individual part recordings. We focus on the main technical challenge of aligning the individual part recordings against a reference “full mix” recording containing a performance of the work. We propose an iterative subtractive alignment approach, in which each part recording is aligned against the full mix recording and then subtracted from it. We also explore different feature representations and cost metrics to handle the asymmetrical nature of the part–full mix comparison. We evaluate our proposed approach on two different datasets: one that is a modification of the URMP dataset that presents an idealized setting, and another that contains a small set of piano trio data collected from musicians during the pandemic specifically for this study. Compared to a standard pairwise alignment approach, we find that the proposed approach has strong performance on the URMP dataset and mixed success on the more realistic piano trio data.

1. INTRODUCTION

This paper explores an application that would enable a group of musicians in quarantine to produce a performance of a piece of chamber music through asynchronous musical collaboration. Asynchronous musical collaboration is usually done with musicians performing their parts synchronized to a reference “click” track. This paradigm works well for many genres of music where the tempo is relatively constant (e.g. pop music) or the musicians are expected to follow a conductor (e.g. choral music). However, this paradigm does not work well with genres of music where musicians are constantly adapting to and influencing one another. As a representative example of the latter, we focus in this paper on the genre of piano trio music, which is ill-suited to a click track paradigm for several

reasons: the tempo is constantly changing and may vary widely across a single movement, the main melodic line is carried by different instruments at different times and may be shared by two or more instruments, parts may contain extended periods of silence, and each instrument is given considerable latitude for individual musical expression. Our goal is to allow each musician the freedom to perform their part as they wish, while still allowing for asynchronous musical collaboration.

Our approach to this problem is to allow the recording of individual parts to be unsynchronized, and to use MIR tools to achieve synchronization post-recording. Figure 1 shows a high-level overview of this paradigm for a piano trio. The primary inputs to the system are three recordings: a recording of the piano part only, a recording of the cello part only, and a recording of the violin part only. These will be referred to as “part” recordings, since they only contain the performance of a single part. The first step (bottom-most block) is to determine the alignment between the part recordings. Because the part recordings are not directly comparable to one another (i.e. they may be playing different notes), we can provide a reference “full mix” recording (e.g. by finding a YouTube video of the piece) that contains all parts played in synchrony, and then use the full mix as additional information to assist our estimate of the joint alignment among the three part recordings. Once we have estimated the alignment among the part recordings, we can then use time scale modification (TSM) to adjust the tempos in each part to produce time scale modified, synchronized part recordings.¹ These synchronized part recordings can be mixed together to produce the final performance. TSM is a well-studied problem [1], and there are effective approaches based on phase vocoding and various overlap-add methods (e.g. [2–4]). The main technical challenge in Figure 1, therefore, is the joint alignment problem among the part recordings and the full mix. We will focus on this technical problem in the remainder of this paper.

Alignment tasks have long been a topic of interest to the MIR community due to their applications in score following, retrieval, and synchronization of various forms of music data. An exhaustive survey of alignment research is beyond the scope of this paper, but here we simply point



© D. Yang, K. Ji, and T. Tsai. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Yang, K. Ji, and T. Tsai, “Aligning Unsynchronized Part Recordings to a Full Mix Using Iterative Subtractive Alignment”, in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

¹ Note that the reference recording is only used to assist in the joint alignment estimation problem. Once we have estimated the joint alignment, we can time scale modify the part recordings however we wish (e.g. modifying two part recordings to match the third part recording).

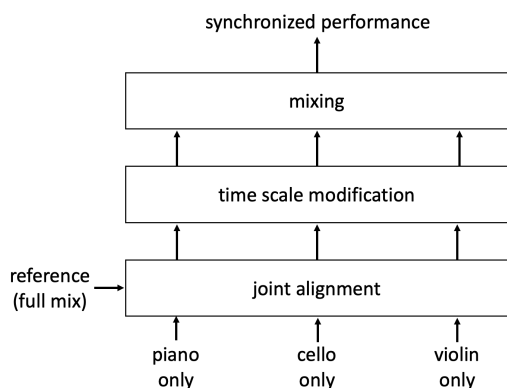


Figure 1. Overview of asynchronous musical collaboration with unsynchronized part recordings. Each musician records their part in isolation in a completely unsynchronized manner, and the part recordings are modified and mixed to produce a synchronized performance.

out current trends in the MIR alignment literature in order to situate our current work in proper context. Recent works on alignment tasks in the MIR literature tend to fall into one of three groups. The first group focuses on alignment across different modalities of music data. The main challenge here is to find a feature representation that enables a direct comparison of similarity between different modalities. Some recent examples of this include aligning lyrics and audio [5, 6], aligning sheet music images and audio [7–9], and aligning sheet music images and MIDI [10, 11]. The second group focuses on performing alignment under a different set of assumptions or conditions than traditional dynamic time warping (DTW). Some examples include handling discontinuities due to jumps or repeats [12–14], handling completely unconstrained jumps such as might be observed in a practice session [15, 16], or aligning two audio mixtures containing a non-disjoint subset of parts from a common piece of ensemble music [17]. The third group focuses on issues of scalability and efficiency of alignment techniques. DTW has quadratic cost in memory and computation, which limits its utility in dealing with long sequences. Previous works have proposed alignment approaches that operate at multiple scales [18, 19], and recent works explore ways to reduce the memory cost [20, 21] or total runtime through parallelization [22].

The alignment problem shown in Figure 1 falls into the second group above — it frames the alignment problem with a different set of assumptions and context. There are at least three significant differences between our proposed scenario and a typical audio–audio alignment scenario. First, we are aligning each part recording against a full mix containing all three parts, so the alignment must be estimated in the presence of other significant sound sources. If we assume that each part has equal volume and interpret the other parts as highly correlated additive noise, we are effectively estimating an alignment in the regime of $10 \log_{10} \frac{1}{2} = -3$ dB SNR. In a typical audio–audio alignment scenario, we might align two full mix recordings of the same piece, which corresponds to a high SNR

regime.² Second, we are estimating the alignment between a *set* of part recordings and a full mix recording, rather than considering a single isolated pairwise alignment. Because we know that the full mix is a mixture of all three parts, the knowledge of one part–full mix alignment is relevant to our estimate of the other alignments. Third, the part recordings may be sparse — they may contain extended periods of silence where the instrument is not playing. In typical audio–audio alignment scenarios, both recordings are usually assumed to be “dense” recordings that contain musical information at all times. Because of the characteristics above, we will refer to the problem in Figure 1 as the part–full mix joint alignment problem.

Our approach to the part–full mix joint alignment problem has two distinct characteristics. First, we adopt an iterative subtractive approach, in which we align each part recording to the full mix, time scale modify the part recording to match the full mix, and then subtract the part recording from the full mix. Second, we explore different cost metrics that have the desired asymmetric behavior: we do not want to penalize the full mix for having spectral peaks that are not present in the part recording (since these peaks may come from the other parts), but we do want to reward the part recording for “explaining” spectral peaks that are observed in the full mix.

This paper has two main contributions. First, we propose and motivate a part–full mix joint alignment problem that would allow musicians to produce chamber music performances without any synchronization or communication, and we present two different datasets to enable its systematic study. One dataset is a modification of the URMP dataset [23], which contains ensemble works of various instrumentation. The other dataset is a small set of real world data that serves as a case study for our application of interest. It was collected during the pandemic specifically for this study and contains multiple performances of a single piano trio work. Second, we propose an iterative subtractive alignment approach, in which each individual part recording is aligned against a reference full mix recording and then subtracted from it. We explore several different cost metrics to account for the asymmetrical nature of the part–full mix comparison. We find that the proposed approach has strong performance on the URMP benchmark and mixed success on the more realistic piano trio data. We present experimental results on both datasets to provide more insight into the performance of our proposed approach.³

2. SYSTEM DESCRIPTION

Figure 2 shows an overview of our proposed iterative subtractive alignment approach. There are two key steps which are repeated multiple times: aligning a single part to the full mix and subtracting the part from the full mix.

² In this case, the primary distortion (apart from the time warping) comes from differences in the instrument, the performer’s interpretation & articulation, and recording conditions. These distortions are also present in our scenario.

³ Code and data can be found at <https://github.com/HMC-MIR/PianoTrioAlignment>

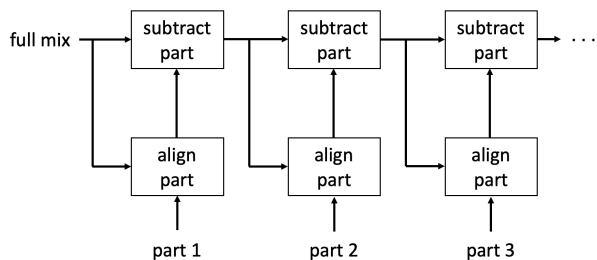


Figure 2. Overview of the subtractive alignment approach to solve the part–full mix joint alignment problem.

These two steps will be described in the next two subsections.

2.1 Aligning a Single Part

The first key step is to align a single part to the full mix. If we were aligning two full mix recordings, we could simply use standard chroma features combined with a cosine distance metric. In our case, however, the full mix contains a mixture of parts, only one of which matches the part recording. In computing a pairwise cost $C[i, j]$ between part frame i and full mix frame j , we do not want to penalize the full mix for containing energy at frequencies not found in frame i of the part recording. This motivates the need for a different feature representation and cost metric. We explore two different methods to account for this asymmetry.

The first method is based on a constant Q transform (CQT). We first compute a CQT on the full mix and part recording using 12 bins per octave between C1 to C10. Let $x_i \in \mathbb{R}^{109}$ be the CQT values for the i^{th} frame in the part recording and let $y_j \in \mathbb{R}^{109}$ be the CQT values for the j^{th} frame in the full mix. We compute the pairwise cost between x_i and y_j as

$$C[i, j] = -\text{sum}(\min(x_i, y_j)) / \text{sum}(x_i)$$

where the min operator computes the elementwise minimum between two vectors and the sum operator sums the elements in a vector to produce a single scalar. The numerator term $\text{sum}(\min(x_i, y_j))$ is a single scalar that indicates how much of the CQT energy in y_j is “explained” by x_i . The denominator term $\text{sum}(x_i)$ normalizes this value by the total amount of energy in x_i .⁴ Therefore, this cost metric rewards x_i for explaining the spectral peaks in y_j , but does not penalize y_j for having more energy than x_i in a frequency bin. This exhibits the type of asymmetrical behavior we desire in our scenario. Note that $C[i, j]$ will always be in the range $[-1, 0]$, where -1 indicates a strong agreement between x_i and y_j .

The second method is based on a binarized constant Q transform (BCQT). We binarize the part CQT and the full mix CQT by applying a hard threshold γ_k to the k^{th} CQT frequency bin. The threshold γ_k is determined by considering 6 bins above and below the k^{th} frequency bin, treating

⁴ We do not square the elements in order to avoid too heavily penalizing large differences.

the resulting $13 \times N$ matrix as a grayscale image (where N is the number of frames in the CQT), and applying the triangle binarization algorithm to determine a threshold [24]. Let $x_i \in \{0, 1\}^{109}$ be the BCQT values for the i^{th} frame in the part recording and let $y_j \in \{0, 1\}^{109}$ be the BCQT values for the j^{th} frame in the full mix. We compute the pairwise cost between x_i and y_j as the negative normalized inner product $C[i, j] = -x_i^T y_j / \text{sum}(x_i)$ where the sum operator sums the elements of a vector to produce a scalar. Again, $C[i, j]$ will be in the range $[-1, 0]$, where -1 indicates a strong agreement between x_i and y_j . Note that the first method weights the importance of each frequency bin according to the amount of energy in it, whereas the binarized approach gives all frequency bins equal importance.

Once we compute the pairwise cost matrix, we use DTW to estimate the alignment between the part recording and the full mix. Because some parts may not be active at the beginning of the piece, we use subsequence DTW to estimate the alignment. Subsequence DTW is a variant of DTW that finds the best alignment between a short query sequence and any subsequence in a longer reference sequence. This allows the alignment path to begin and end anywhere in the full mix, rather than assuming that the full mix and part recording both begin and end at the same time. We allow for (part, full mix) transitions of (1, 1), (1, 2), and (2, 1) with multiplicative transition weights of 1, 1, and 2, respectively.

At the end of the first key step, we have an estimated alignment between a single part and the full mix recording. This estimated alignment is passed to the subtraction block, which we describe in the next subsection.

2.2 Subtracting a Single Part

The second key step is to subtract the part recording from the full mix. This is done on the CQT representation through spectral subtraction. This process consists of four substeps, which are described in the next four paragraphs.

The first sub-step is to time warp the part recording to match the timing of the full mix. This can be done very easily by using the estimated alignment between the part recording and full mix. For example, if frame $y_k \in \mathbb{R}^{109}$ in the full mix CQT is aligned to frame $x_i \in \mathbb{R}^{109}$ in the part CQT, then frame \tilde{x}_k in the *time-warped* part CQT will be $\tilde{x}_k = x_i$. When there are (1, 2) transitions in the estimated alignment, we can estimate “missing” frames through interpolation. In order to handle frames that fall outside the estimated alignment (e.g. the part recording begins matching the full mix 20 seconds into the performance), we simply pad additional frames with zeros at the beginning and end as needed. At the end of this first substep, we have a time-warped part CQT \tilde{X} which has the same dimensions as the full mix CQT Y .

At this point, we *could* simply subtract the time-warped part CQT from the full mix CQT. However, this does not account for volume differences between the two recordings. For example, a violin part recording containing a single solo violin player may be much louder than the violin signal in the full mix recording. These differences may

be global differences due to microphone recording levels or local differences due to musical interpretation (e.g. the recording levels are the same, but one violinist prefers to play a particular section more softly than the other violinist). In order to account for these volume differences, we break the part recording into segments and estimate a volume gain factor for each segment.

The second substep, then, is to break the part recording into segments. We do this by performing silence detection on the part recording, and then consider contiguous regions of silence or non-silence as segments. Because the part recording only contains a single instrument, the silence detection is relatively straightforward, and we use a simple energy-based approach. We first compute the amount of energy in windows of length 0.75 seconds across the entire recording. We then model the distribution of log energy (within a single window) with a Gaussian mixture model with 3 mixtures. We interpret the Gaussian with the smallest mean as a model for silence in the recording. We compute the probability that a frame is silence $P(\text{silence}|\log \text{energy})$ using Bayes' rule and threshold at 0.5. We find this energy-based silence detection approach to be sufficiently robust for our application. Because the piano part is playing throughout the entire piece, we use a very simple scheme for segmenting the piano recording: we simply break it up into non-overlapping 5 second segments.

The third substep is to estimate a volume gain factor for each (time-warped) segment. We estimate the optimal volume gain factor α^* in the following manner. Let $\tilde{X}_s \in \mathbb{R}^{109 \times L}$ be the CQT representation of a single time-warped segment of length L frames, and let $Y_s \in \mathbb{R}^{109 \times L}$ be the CQT representation for the corresponding section of the full mix. We calculate the optimal gain factor α^* as

$$\alpha^* = \arg \max_{\alpha} \text{sum}(\min(\tilde{X}_s \alpha, Y_s) - \max(\tilde{X}_s \alpha - Y_s, 0))$$

where the min and max operators are performed elementwise between two matrices and the sum operator sums all elements in a matrix to produce a single scalar. The $\min(\tilde{X}_s \alpha, Y_s)$ term indicates how much of the CQT energy in the full mix segment is explained by the volume-scaled & time-warped part recording. The $\max(\tilde{X}_s \alpha - Y_s, 0)$ term is a penalty for overestimating the energy in the full mix CQT. To approximately solve this optimization problem, we simply consider a range of values for α between 0.1 and 100 and use the value that maximizes the objective function. After this third substep, we have a volume gain factor α^* for every segment in the part recording.

The fourth substep is to perform the spectral subtraction. For each segment, we subtract the volume-scaled & time-warped part CQT from the full mix CQT as

$$Y_{s,mod} = \max(Y_s - \tilde{X}_s \alpha^*, 0)$$

where the max operator is performed elementwise. The max operator ensures that the modified full mix CQT remains a non-negative matrix. The output of the subtraction block in Figure 2 is the modified full mix CQT with the part recording subtracted out.

Recording Type	# Recordings		Duration	
	Train	Test	Train	Test
Piano only	2	2	21.8m	22.4m
Violin only	2	2	17.5m	18.9m
Cello only	2	2	19.0m	18.9m
Full mix (YouTube)	2	2	19.1m	21.2m
Total	8	8	77.3m	81.3m

Table 1. Summary of the Mendelssohn piano trio data collected from musicians in quarantine. All possible combinations of the recordings are considered, resulting in 16 training episodes and 16 test episodes.

2.3 Aligning Multiple Parts

Once the first part has been aligned and subtracted out from the full mix CQT, we repeat the entire process with the next part recording. At each iteration of this process, we always use the most updated version of the full mix CQT with previous parts subtracted out. The final output of the system is an estimated alignment between each part recording and the full mix recording.

3. EXPERIMENTAL SETUP

Doing a rigorous empirical study of our proposed application of interest presents several significant challenges. As with constructing any alignment dataset, annotating beat timestamps is a very time-consuming task. But even beyond that, simply *collecting* suitable audio data for our application is an even more significant challenge. Because musicians don't have any incentive to record themselves playing a single part of a chamber work, such data is not readily available in the wild. Because of the challenges of getting realistic data, we opted for a two-pronged approach: we collected a small amount of data that is specifically tailored to our application to serve as a case study, and we also modified an existing dataset to study the same alignment problem in an idealized setting.

The application-specific data was collected in the following manner. In the midst of the pandemic, we recruited three musicians to participate in our study: a cellist and a violinist in the Claremont Colleges Orchestra and a pianist who has studied privately in college. Based on the expressed preferences of all three musicians, they agreed to learn the first movement of the Mendelssohn piano trio no. 1 in D minor. The musicians were asked to learn their parts and then record themselves playing their part in isolation. During the period when the musicians were learning their parts, they had no joint practices together and did not have any communication about their interpretation of the piece (e.g. at what tempo to play the piece). Each musician used a cell phone to record themselves playing their part four times from beginning to end, including counting out measures of rest. In addition to the individual part recordings, we also found four different performances of the Mendelssohn piano trio on YouTube. These YouTube performances serve as the reference full mix recording shown

Piece Type	# Pieces		Duration	
	Train	Test	Original	Modified
Duos	6	5	20.8m	215.0m
Trios	6	6	17.7m	181.8m
Quartets	6	8	27.5m	285.8m
Quintets	2	5	14.2m	149.0m
Total	20	24	80.2m	831.6m

Table 2. Summary of the URMP data used as an additional evaluation benchmark. The full mix recordings were randomly modified in tempo to generate 10 episodes per piece, resulting in 200 training episodes and 240 test episodes.

at left in Figure 2. Put together, the piano trio data contains a total of 16 recordings and 159 minutes of data.

After this data had been collected, we asked each musician to annotate the downbeats in all 16 audio recordings using SonicVisualizer. Because there are many extended periods of silence in the cello and violin part recordings, the musicians were asked to selectively annotate only those downbeats that they felt could be reasonably inferred. We collected all $16 \times 3 = 48$ annotation files (23,274 total beat annotations) and merged them by taking the average of annotated timestamps for all downbeats containing three annotations. Downbeats with less than three annotations were discarded. The result of this merging process is a set of 16 ground truth annotation files for the 16 audio recordings.

In summary, the application-specific data consists of 16 audio recordings containing 4 violin part recordings, 4 cello part recordings, 4 piano part recordings, and 4 full mix recordings. All 16 recordings are unsynchronized, and each recording has ground truth annotations of downbeats. We will refer to this dataset as the piano trio data. Table 1 shows an overview of this dataset. As discussed above, it has the benefit of being real world data that is collected in the exact application scenario of interest, but has the drawback of being very limited in diversity (only one piece and one set of performers). The results on this dataset should therefore be seen as a case study.

We evaluated system performance in the following manner. We set apart two violin part recordings, two cello part recordings, two piano part recordings, and two full mix recordings for training, and we consider all possible combinations of training recordings, resulting in $2 \times 2 \times 2 \times 2 = 16$ different training episodes. For test evaluation, we likewise consider all possible combinations of test recordings, resulting in 16 test episodes. For each episode, we evaluate the accuracy of the predicted alignment between each part recording and the full mix recording. We calculate the percentage of annotated downbeats whose predicted alignment error is greater than a maximum allowable error tolerance, and we calculate this error rate for several different error tolerances. Because the alignment accuracy of different instruments varied widely, we report results for each instrument separately (i.e. piano–full mix,

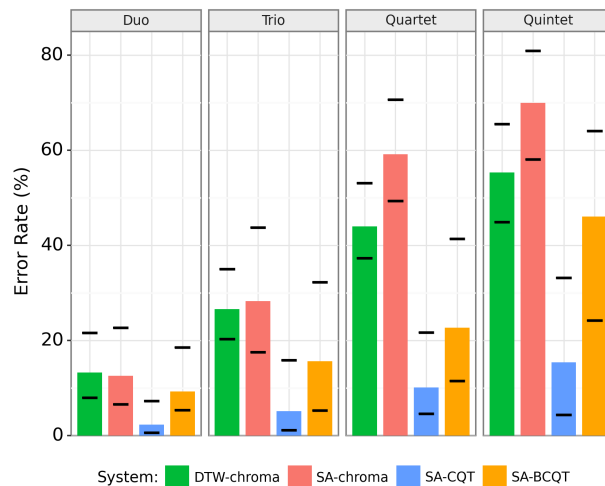


Figure 3. Results on the URMP dataset. Colored bars indicate the error rate at an error tolerance of 200 ms, and the horizontal bars above and below each colored bar indicate the error rate at error tolerances of 100 ms and 400 ms, respectively. Results are separated by piece type.

cello–full mix, and violin–full mix).

In parallel with the piano trio data, we also constructed a benchmark using the URMP dataset [23] to study the same alignment problem, albeit in an idealized setting. The URMP dataset contains recordings of 44 different ensemble works of various instrumentation and ranging from duos up to quintets. For each piece, the dataset contains recordings of each individual part played in isolation, as well as a full mix recording containing all parts mixed together. When recording the data, the musicians listened to a reference track on earphones, so all part recordings are synchronized. The full mix recording was generated synthetically by mixing the individual part recordings together with appropriate offset. The dataset contains a symbolic score for each piece, annotations of F0 trajectories, and timestamps for note onsets.

We modify the URMP dataset to study the part–full mix joint alignment problem. Because each part recording is already synchronized with the full mix recording, the alignment problem is trivial. To make the problem non-trivial, we modify the full mix recording in the following way. First, we split the full mix recording into three segments of equal length. For each segment, we use time scale modification to change the tempo by a constant, random factor while preserving the pitches of all parts. We use the phase vocoder implementation in [25] to perform the time scale modification. The tempo is uniformly sampled between $0.66r_{nom}$ and $1.5r_{nom}$ on a log scale, where r_{nom} indicates the nominal tempo of the full mix segment. Because this process is probabilistic, we generate 10 different modified full mix recordings for each piece in URMP, which provides us with 10 episodes per piece.

We evaluate performance on our modified URMP data in the same fashion as for the piano trio data. We set apart 20 pieces for training and 24 pieces for testing. Because

ground truth timestamps are provided for note onsets, we evaluate alignment accuracy at each note onset (rather than at downbeats). As before, we compute the error rate at several different error tolerances. The alignment accuracy varied widely based on the number of parts in the piece, so we report results for duos, trios, quartets, and quintets separately. Table 2 provides an overview of the URMP data used in our experiments.

4. RESULTS

Figure 3 shows the performance of four different systems on the URMP benchmark. The first system (‘DTW-chroma’) simply performs an independent pairwise alignment between each part recording and the full mix using chroma features and cosine distance metric. This can be considered our baseline, since it is a default choice in many audio–audio alignment applications. The second and third systems use the subtractive alignment approach based on the CQT representation (‘SA-CQT’) and BCQT representation (‘SA-BCQT’). We also include a fourth system that uses the subtractive alignment approach with standard chroma features and cosine distance metric (‘SA-chroma’) as a way to tease apart the effect of the feature representation and the iterative subtraction approach. The colored bars indicate the error rate at an error tolerance of 200 ms, and the horizontal bars above and below each colored bar indicate the error rate at error tolerances of 100 ms and 400 ms, respectively. The results are shown separately for duos, trios, quartets, and quintets.

There are three things to notice about Figure 3. First, the performance of all systems gets progressively worse as we move from duos to trios to quartets to quintets. This is to be expected, since the problem becomes progressively more challenging as more “noise” sources are added and the effective SNR becomes lower. Second, the SA-CQT approach has the best performance by a wide margin, far outperforming the baseline DTW-chroma system. For example, the DTW-chroma baseline has a 44.0% error rate with 200 ms tolerance on quartet pieces, while the SA-CQT approach achieves 10.1% error rate. Third, even with a subtractive approach, the SA-chroma and SA-BCQT approaches scale poorly with the number of instruments.

Figure 4 shows the results of the same four systems on the piano trio data. Results are shown separately for each instrument’s alignments against the full mix. For the subtractive approaches, we use a piano–cello–violin ordering for spectral subtraction, which we found to work best on the training data. There are three things to notice about these results. First, the performance depends a lot on the instrument: the piano alignments are best by far, and the cello and violin alignments are much worse. This is perhaps not too surprising, since the piano part has a much more distinctive spectral profile due to being a polyphonic instrument. Second, the subtractive approach potentially provides a significant improvement for the cello alignment, but at best only marginal improvements for violin. Third, SA-CQT is no longer a clear winner as it was on the URMP data. Instead, we can see that different approaches seem to

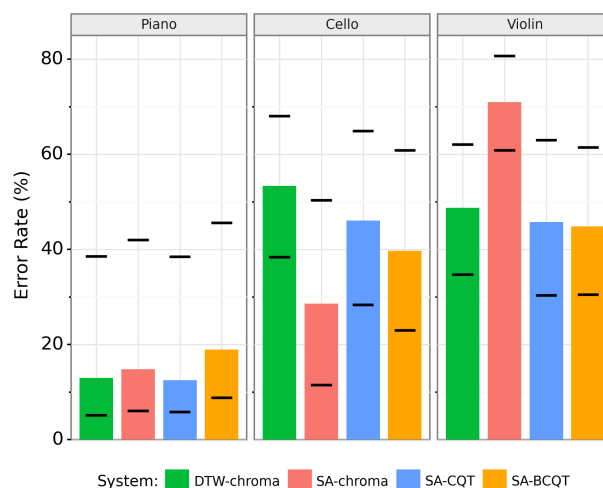


Figure 4. Results on the piano trio dataset. Results are shown separately for each instrument’s alignments against the full mix. All subtractive approaches use a piano–cello–violin ordering when performing spectral subtraction.

work best for different instruments: SA-CQT and DTW-chroma work best for piano, SA-chroma works best for cello, and SA-BCQT works best for violin. One area that might be interesting to explore in the future is using different feature representations and cost metrics for different part recordings in order to exploit the unique characteristics of each instrument.

5. CONCLUSION

This paper envisions an application in which a group of musicians in quarantine can generate a performance of a chamber work by recording each part in isolation in a completely unsynchronized manner, and then aligning, time scale modifying, and mixing the recordings. We focus on the main technical challenge of aligning the individual part recordings. Our approach is to use an auxiliary “full mix” recording of the piece as a reference, and to align each part recording against the full mix. We explore an iterative subtractive alignment approach in which each part is aligned against the full mix and then subtracted from it. We characterize the performance of several variants of this approach on two different datasets: one derived from the URMP dataset that contains ensemble works of various instrumentation, and the other consisting of multiple recordings of a piano trio collected from musicians in quarantine during the pandemic. We find that the subtractive alignment approach works reasonably well on the URMP data, but has mixed success on the piano trio data. We present experimental analysis and suggest directions for future improvement.

6. REFERENCES

- [1] J. Driedger and M. Müller, “A review of time-scale modification of music signals,” *Applied Sciences*, vol. 6, no. 2, p. 57, 2016.
- [2] J. Driedger, M. Müller, and S. Ewert, “Improving time-scale modification of music signals using harmonic-percussive separation,” *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 105–109, 2013.
- [3] A. Moinet and T. Dutoit, “PVSOLA: a phase vocoder with synchronized overlap-add,” in *Proc. of the 14th Int. Conference on Digital Audio Effects (DAFx)*, 2011, pp. 269–275.
- [4] J. Laroche and M. Dolson, “Improved phase vocoder time-scale modification of audio,” *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, 1999.
- [5] D. Stoller, S. Durand, and S. Ewert, “End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model,” in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 181–185.
- [6] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. D’alché-Buc, “Multilingual lyrics-to-audio alignment,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 512–519.
- [7] F. Henkel, R. Kelz, and G. Widmer, “Learning to read and follow music in complete score sheet images,” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2020, pp. 780–787.
- [8] M. Dorfer, J. Hajič, A. Arzt, H. Frostel, and G. Widmer, “Learning audio-sheet music correspondences for cross-modal retrieval and piece identification,” *Trans. of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 22–33, 2018.
- [9] M. Dorfer, F. Henkel, and G. Widmer, “Learning to listen, read, and follow: Score following as a reinforcement learning game,” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 784–791.
- [10] D. Yang, T. Tanprasert, T. Jenrungrot, M. Shan, and T. Tsai, “Midi passage retrieval using cell phone pictures of sheet music,” in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 916–923.
- [11] T. Tsai, D. Yang, M. Shan, T. Tanprasert, and T. Jenrungrot, “Using cell phone pictures of sheet music to retrieve midi passages,” *IEEE Transactions on Multimedia*, vol. 22, no. 12, pp. 3115–3127, 2020.
- [12] C. Fremerey, M. Müller, and M. Clausen, “Handling repeats and jumps in score-performance synchronization,” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2010, pp. 243–248.
- [13] M. Grachten, M. Gasser, A. Arzt, and G. Widmer, “Automatic alignment of music performances with structural differences,” in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 607–612.
- [14] M. Shan and T. Tsai, “Improved handling of repeats and jumps in audio-sheet image synchronization,” in *Proc. of the International Society for Music Information Retrieval Conference*, 2020, pp. 62–69.
- [15] T. Nakamura, E. Nakamura, and S. Sagayama, “Real-time audio-to-score alignment of music performances containing errors and arbitrary repeats and skips,” *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, vol. 24, no. 2, pp. 329–339, 2015.
- [16] Y. Jiang, F. Ryan, D. Cartledge, and C. Raphael, “Offline score alignment for realistic music practice,” in *Sound and Music Computing Conference*, 2019.
- [17] A. Maezawa and H. G. Okuno, “Audio part mixture alignment based on hierarchical nonparametric bayesian model of musical audio sequence collection,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 5212–5216.
- [18] M. Müller, H. Mattes, and F. Kurth, “An efficient multiscale approach to audio synchronization,” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2006, pp. 192–197.
- [19] S. Salvador and P. Chan, “FastDTW: Toward accurate dynamic time warping in linear time and space,” in *Proc. of the KDD Workshop on Mining Temporal and Sequential Data*, 2004.
- [20] C. Tralie and E. Dempsey, “Exact, parallelizable dynamic time warping alignment with linear memory,” in *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, 2020, pp. 462–469.
- [21] T. Prätzlich, J. Driedger, and M. Müller, “Memory-restricted multiscale dynamic time warping,” in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (CASSP)*, 2016, pp. 569–573.
- [22] T. Tsai, “Segmental dtw: A parallelizable alternative to dynamic time warping,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.
- [23] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, “Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications,” *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

- [24] G. W. Zack, W. E. Rogers, and S. A. Latt, "Automatic measurement of sister chromatid exchange frequency," *Journal of Histochemistry & Cytochemistry*, vol. 25, no. 7, pp. 741–753, 1977.
- [25] S. Yong, S. Choi, and J. Nam, "PyTSMoD: A python implementation of time-scale modification algorithms," in *Late-Breaking Demo Session at the International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

ADTOF: A LARGE DATASET OF NON-SYNTHETIC MUSIC FOR AUTOMATIC DRUM TRANSCRIPTION

Mickaël Zehren
Umeå Universitet
mzehren@cs.umu.se

Marco Alunno
Universidad EAFIT Medellín
malunno@eafit.edu.co

Paolo Bientinesi
Umeå Universitet
pauldj@cs.umu.se

ABSTRACT

The state-of-the-art methods for drum transcription in the presence of melodic instruments (DTM) are machine learning models trained in a supervised manner, which means that they rely on labeled datasets. The problem is that the available public datasets are limited either in size or in realism, and are thus suboptimal for training purposes. Indeed, the best results are currently obtained via a rather convoluted multi-step training process that involves both real and synthetic datasets. To address this issue, starting from the observation that the communities of rhythm games players provide a large amount of annotated data, we curated a new dataset of crowdsourced drum transcriptions. This dataset contains real-world music, is manually annotated, and is about two orders of magnitude larger than any other non-synthetic dataset, making it a prime candidate for training purposes. However, due to crowdsourcing, the initial annotations contain mistakes. We discuss how the quality of the dataset can be improved by automatically correcting different types of mistakes. When used to train a popular DTM model, the dataset yields a performance that matches that of the state-of-the-art for DTM, thus demonstrating the quality of the annotations.

1. INTRODUCTION

Automatic drum transcription (ADT) consists of creating a symbolic representation of the notes played by the drums in a music piece. Two targets that would benefit from such transcriptions are musicians, for example when learning a musical piece, and music information retrieval (MIR), that can leverage the location of the notes to draw a deeper knowledge of a music track (e.g., its structure). ADT is known to be difficult to achieve. In fact, as explained in a recent state-of-the-art review by Wu et al. [1], it is tackled in several manners and with different levels of complexity. The most basic aspect undertaken is the automatic classification of isolated drum sounds. Here, we are interested in solving the more general and complex task of

drum transcription in the presence of melodic instruments (DTM). In DTM, the input consists of polyphonic music (drums and accompanying instruments); the output is a log with time stamp and instrument for each drum note. As Wu et al. argued, much progress has been made recently in ADT (and, therefore, in DTM) thanks to deep learning approaches. However, a high volume of annotated data is needed for neural networks to perform well, and such data is difficult to obtain, mainly because the annotation process is labor-intensive. This explains why the publicly available datasets are usually either small (e.g. [2–4]) or consist of augmented data (e.g. [5,6]) or synthesized audio (e.g. [7,8]), both of which are not direct representations but only estimations of real music. Therefore, current datasets seem to be suboptimal for DTM either because of quantity or data authenticity.

In this work, we explore a way to reach both the required quantity and realism of the data needed for DTM by using crowdsourced annotations of a high volume of real-world (not synthetically created) audio tracks. In fact, we realized that this amount of data can be found in rhythm games such as RockBand¹ or PhaseShift². In these games, one of the goals is to correctly play the drum line of a song on a toy drum kit. Songs come with the game, but players can also add audio tracks and their own annotations of the drums parts. Because of this feature, a large online community of players and musicians emerged to extend the catalog of playable tracks and share custom game files, also known as “custom charts”. These data have the advantage of being fundamentally similar to the content of current ADT datasets and contain the audio source along with the representation of the notes being played on the drums.

The outcome of our work is a new methodology to build a dataset from custom charts. Following our method, we build a dataset named Automatic Drums Transcription On Fire (ADTOF)³ that is composed of a large amount of realistic data. As mentioned above, the large amount is achieved through crowdsourcing annotations from a much larger group of people than previously observed, to our knowledge. Realism is due, instead, to the use of real-world as opposed to augmented or synthesized music tracks. Yet, quantity and realism are useful only if the

¹ <https://www.rockband4.com/>

² https://store.steampowered.com/app/865250/Phase_Shift/

³ The name is a reference to “Frets On Fire”, one of the earliest rhythm game.



Dataset	Hours	Classes	Real music
ENST [2]	1.02	20	✓
MDB [3]	0.35	21	✓
RBMA [4]	1.72	24	✓
SDDS [7]	467	14	×
TMIDT [8]	259	18	×
ADTOF (ours)	114	5	✓

Table 1. List of datasets for DTM.

annotations contain as few mistakes as possible.

In order to ensure a sufficient quality, we used a systematic way of curating the data from an online source by selecting the tracks that are less likely to contain wrong annotations. In fact, while manually assessing the annotations, we found many discrepancies between the locations of the annotations and the positions of the actual sound onsets. To overcome this issue we adapted the automatic alignment technique described in the work of Driedger et al. [9] to correct the time precision of the annotations. We also found many inconsistencies in the labels used to designate specific instruments of the drum kit, which we solved by reducing the set of instrument classes to be detected. Finally, in order to assess how useful the annotations were after being processed, we evaluated our new dataset as both a training and test data for the popular convolutional recurrent neural network (CRNN) illustrated in the work of Vogl et al. [8]. The result is that ADTOF allows for the direct training of a model that achieves comparable performance to the state-of-the-art model trained on multiple other datasets. It also provides complementary information and generalization capability.⁴

The rest of this article is organized as follows: Section 2 contains a survey of related works. The data with the annotation and curation process are presented in Section 3, and the methodology to automatically clean them is detailed in Section 4. In Section 5, experiments on training and testing are presented. The results are then discussed in Section 6. Conclusions are drawn and future works are described in Section 7.

2. RELATED WORK

Multiple datasets with different characteristics have been created to solve specific aspects of ADT. Since we deal with DTM, though, in this section we discuss only datasets containing polyphonic music (see Table 1).

To our knowledge, the oldest public dataset suitable for DTM is ENST [2] created in 2006. This dataset contains the recordings of three professional drummers playing along with a variety of musical accompaniments composed for drum kit practicing. More recently, in 2017, MDB drums [3] was created by adding drum transcriptions to 23 tracks from MedleyDB [10], and RBMA [4] was released, with annotations, on the freely available album

“Various Assets - Not For Sale: Red Bull Music Academy New York 2013”. These datasets are standard in DTM, but they are limited in several ways. First, because of the difficulties inherent in annotating music, these datasets are small, with a cumulative time just above three hours. Second, the number of occurrences of each instrument in a drum kit is generally unbalanced, with some instruments (e.g., crash cymbal, ride cymbal) appearing much less than others (e.g., bass drum, snare drum). Lastly, in these datasets, data diversity is largely reduced (e.g., ENST contains audio from a limited number of drum kits, RBMA is biased toward a few music genres). As a consequence, the majority of DTM research [4, 5, 11–13] narrows down to the identification of three main drum classes — kick drum, snare drum and hi-hat.

As an effort to increase the size of the manual annotations, data augmentation was employed by Vogl et al. [5] and, more recently, by Jacques and Röbel [6]. In these studies, data augmentation techniques (e.g., pitch-shifting, time-stretching the audio) usually increased the performances of the model trained on the augmented data. However, according to some of the authors in a later work [8, p. 4], this improvement is limited.

Another approach taken to contrast data paucity is the generation of synthetic datasets which consist of synthesized audio generated from a symbolic representation of music (i.e. MIDI files). This technique allows to create larger datasets because it removes the labor needed to annotate the audio tracks, since the ground truth is deduced from the generation process. Moreover, audio synthesis gives the flexibility to balance instrument distribution by artificially replacing more common drum classes with sparser ones.

Following this approach, Cartwright and Bello proposed in 2018 the Synthetic Drum Dataset (SDDS) [7] that is multiple orders of magnitude larger than the previous datasets. In their work, the audio has been rendered from a collection of MIDI drum loops using randomly selected drum samples, augmented with harmonic background noise and other data augmentation methods. The same year, Vogl et al. created another synthetic dataset, which we refer to as TMIDT [8], by using MIDI files available online to synthesize both drums and the accompaniment parts in such a way that drums classes would be distributed in a natural and balanced fashion. Both these works indicate that models trained on a large synthetic dataset alone do not outperform models trained on small real-world datasets, with still possible performance improvements for some underrepresented classes when using TMIDT. Furthermore, Vogl et al. [8, p. 5] raised the concern that the atypical nature of drum patterns that underwent a balancing process could harm the model and they showed that this technique is ineffective when making evaluation on real-world datasets. In conclusion, results improve only when real data is somehow involved: by training with both synthetic and real data [7], or by training first with synthetic data and then refining the outcome with real data [8].

⁴ The set of tools developed to curate and process the annotations as well as the pre-trained models and complementary material can be found at <https://github.com/MZehren/ADTOF>

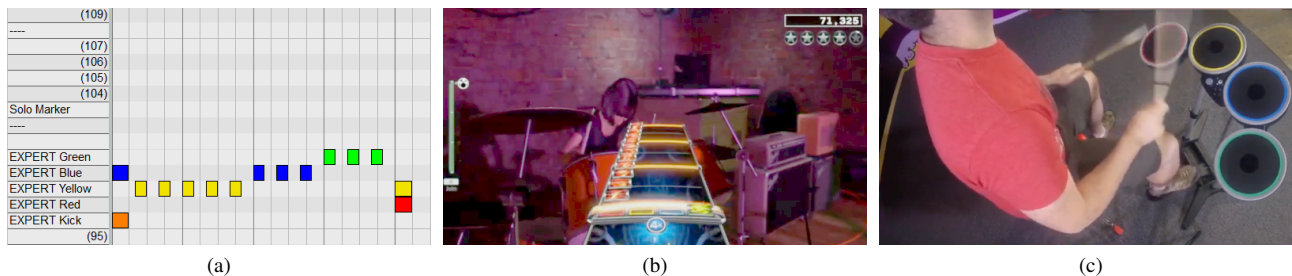


Figure 1. Digital audio workstation interface used to create the annotations with mouse and keyboard (a). The result is then playable in-game (b) using a drum-shaped game controller (c). Pictures from <http://docs.c3universe.com> and [14].

Even though the synthetic datasets are pushing further the state-of-the-art of DTM by providing more data, they do not seem to solve completely the need for more real annotated audio. In order to tackle this issue, we built the ADTOF dataset, the first to our knowledge that is both large and contains real-world audio tracks, with more than 114 hours of annotated music and the transcription of five different classes of drums.

3. DATASET

To build our dataset, we downloaded openly shared custom charts made for rhythm games. In this section, we discuss how the annotations were made and how we selected them.

3.1 Annotation process

The custom charts are created by players and musicians who desire to play along with their favorite tracks. They consist of the symbolic annotations of the drum onsets, usually in a MIDI file with standardized pitches representing the notes the player is supposed to play on the drums (we refer to them as GAMEPLAY annotations). In addition to the transcription, a chart contains beat information, the track’s meta-data and the audio track to play along with. All this information is similar to what is found in an ADT dataset such as those introduced in Section 2. Some of the charts also contain other labels used to animate in-game musicians and are known as ANIMATIONS.

To create the annotations while ensuring quality and consistency in the data, guidelines and tools have been built by the players community and shared online.⁵ These guidelines specify how to create the annotations in a digital audio workstation by manually building a grid of beats filled offline with mouse and keyboard (Fig. 1a). The result is then packaged with a tool and finally manually controlled in-game (Fig. 1b) and playtested on the game controller (Fig. 1c).

3.2 Data selection

Once the transcription is ready, it is shared on a collaborative website such as the popular “Rhythm Gaming World.”⁶ This specific website offers a listing of the transcriptions, a space for users’ feedbacks and comments, and

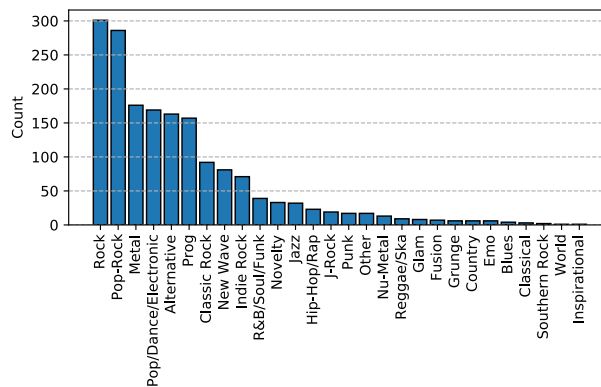


Figure 2. Genre distribution of tracks in ADTOF.

a forum to discuss and report mistakes. At the time of writing, Rhythm Gaming World lists more than 25,000 transcribed tracks. Most of them contain annotations for drums (but other instruments such as guitar, keyboards, and bass are also playable in rhythm games) and are labeled as “Pro drums”, a tag added by the authors and meaning that the notes annotated should be all those found in the real song rather than an approximation or a simplification thereof. From this repository, we downloaded the top-rated 1700 tracks (the largest number of tracks we could fit into the local memory of the server used for the evaluation); this is the dataset we used in this study.

This subset contains an uneven distribution of music genres with a bias toward “Rock”, possibly the most popular genre in the rhythm games community (see Fig. 2). The vast majority of the music is classified as Western with very few occurrences of “World”, as the former is the most likely to contain parts for a typical drum kit like those used in rhythm games.

In order to use this subset of data, we created an open-source set of tools that automatically convert any custom charts into a standard format for ADT.

4. AUTOMATIC DATA CLEANSING

The annotations collected from custom charts are not directly usable for ADT. This is due in part to the crowd-sourced nature of the annotations, and in part to the specific target for which the annotations were created, that is, video games. On the one hand, different people will inevitably

⁵ <http://docs.c3universe.com/rbndocs/index.php>

⁶ <https://rhythmgamingworld.com/>

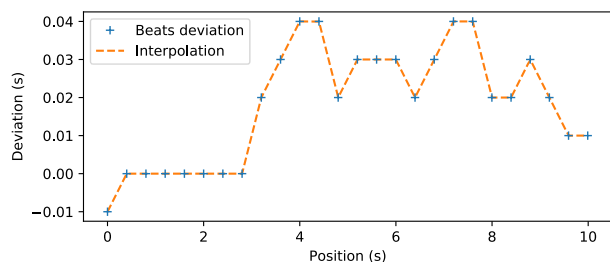


Figure 3. Deviation of the beats annotation as computed with the method from [9] and the interpolation used to correct the drum notes for the first 10 secs of a track.

have different annotation expertise, and on the other hand, some of the annotations do not aim at being accurate, but at improving the players’ experience. By conducting a visual and auditive inspection of random tracks in our dataset, we identified two recurring issues which could be detrimental for ADT. In this section we illustrate how such issues can be corrected automatically.

4.1 Inaccurate timing

The first issue is a lack of time accuracy: We identified that many annotations are placed in a neighborhood of the actual notes. This might be due to human imprecision, but also to the fact that in some tracks the tempo is not constant (for instance because they come from live performances), making them especially difficult to annotate. The observed distance between real and annotated events might be too extreme for the training of a typical algorithm used for ADT. For example, the models discussed by Vogl [8] require the annotations to be roughly within 10 ms from the actual onset,⁷ whereas we usually witnessed discrepancies around 50 ms.

To improve the alignment of the annotations, we adapt the work of Driedger et al. [9] whose idea is to correct human annotations of beats by “snapping” them onto their most likely position according to a beat tracking algorithm. The assumption behind this reasoning is that human annotations are meant to correspond to actual events but, due to human error, they are likely slightly misplaced; instead, beat tracking algorithms, if successful, locate beats accurately. Then, by comparing human annotations with algorithmically identified beats, it should be possible to correct wrongly placed human annotations. The beat tracking algorithm used is Böck’s [15], available in the library Madmom [16]. We extend this initial method, originally meant to correct human taps recorded live, to also correct the annotated drum notes occurring in between the beats. To this end, we interpolate the corrections to intermediate positions as represented in Fig. 3. We use a linear interpolation as it is likely to represent the true deviation of the drum notes.

To further improve the overall quality of the dataset, we also include an automatic sanity check, to make sure that

⁷ This is because those models are tuned to work at 100Hz and, thus, are making a prediction every 10 ms.

ANIMATION	GAMEPLAY	ADTOF
Bass drum	Orange drum	BD
Snare drum	Red drum	SD
Rack tom 1	Yellow drum	TT
Rack tom 2	Blue drum	
Floor tom	Green drum	
Hi-hat open	Yellow cymbal	HH
Hi-hat close		
Crash 1	Blue and green cymbal	CY + RD
Crash 2		
Ride Cymbal		

Table 2. Classes used for the ANIMATION and GAMEPLAY annotations (presented in Sec. 3.1), and mapping onto ADTOF classes.

the majority of the corrected beats align to the algorithmically detected beats, and that the magnitude of the corrections does not exceed 80ms. A total of 140 tracks that do not satisfy these requirements were discarded.

4.2 Inconsistent labeling

The second issue we identified in the annotations is the inconsistent use of labels. This issue is apparent for specific drums sounds such as the three variants of the toms (i.e., yellow, blue, and green drums), which are challenging to discriminate even within one track. Moreover, since the same variant of toms might sound drastically different depending on mix and playing style, it is especially difficult to achieve consistent labeling across tracks. Our solution consists in merging different classes into one (see Table 2, 2nd and 3rd columns), a simplification which is not uncommon in ADT, as the discrimination of the toms sounds is not as relevant as correctly identifying their presence.

Another cause of inconsistencies is the fact that some drums have an ambiguous representation in the toy drum kit used as game controller. For instance, we found that on the toy drum kit, the yellow in-game cymbal ambiguously represents both the open and closed hi-hat; similarly, the blue and green in-game cymbals are used interchangeably for crash and ride sounds. For a subset of tracks, these ambiguities can be resolved by looking at the ANIMATION annotations (Table 2, 1st column), which are highly accurate and span a larger drumset. However, not all the tracks do have such annotations. In our dataset, we map the vocabularies used in ANIMATION or GAMEPLAY down to five classes, the largest number of classes for which most ambiguities are removed.

We also observed that in accordance with the guidelines for the annotators, to ease the gameplay, specific sounds that cannot be played on a toy drum kit, have been represented by combinations of drum hits (e.g., a snare flam is represented as a hit on both the red and yellow drum; an

accentuated open hi-hat sound is played on the green cymbal). For these cases, we use the ANIMATION annotations, when available, to detect discrepancies with the GAMEPLAY annotations, and then we adopt the former as ground truth.

Finally, aware that the guidelines do not cover all possible cases and that not all annotators follow the guidelines, we performed one last check on the 10% tracks with the lowest prediction score according to a preliminary trained ADT algorithm. With this extra check, we removed a total of 88 tracks; among these, there were tracks containing multiple drums but with annotations for only one, and tracks with classes outside of the vocabulary wrongly annotated.

5. EXPERIMENTS

To evaluate the quality of our dataset (ADTOF), we use it in two typical tasks, namely the evaluation, and the training of a DTM model. In the first case, we compare the performance achieved by a state-of-the-art algorithm on ADTOF and on other datasets. Should the performance on ADTOF be worse than on the other datasets, then we would conclude that the ADTOF’s annotations contain errors and/or that ADTOF’s tracks are especially difficult to transcribe for the given algorithm. By contrast, a good performance on ADTOF indicates that the annotations and the algorithmic estimations are in good agreement with each other. Since it is very unlikely that annotations and estimations agree on mistakes, the agreement is a strong indication that both are correct. In the second case, one same model is trained (and tested) on different datasets. By evaluating the performance that the model trained on ADTOF achieves on other datasets, we can assess both the quality of ADTOF’s annotations, and how representative ADTOF’s tracks are for the task.

5.1 Model selection

The state-of-the-art model we evaluate and train on our dataset is the convolutional recurrent neural network (CRNN) presented in the work of Vogl et al. [4, 8]. This model achieves the best overall results according to the community evaluation MIREX⁸ on their private data, is well studied, and it has been reproduced [7]. The strength of this model resides in the two types of building blocks used in its architecture, namely convolutional and recurrent layers (see Fig. 4). The convolutional layers model the local acoustic features of the onsets, while the recurrent layers model the temporal aspect of the drum patterns. This combination of types of layers has been found to give the best results.

The input of the neural network is a spectrogram depicting the variation of the frequencies over time. Specifically, due to its good performance with this model, the *log-frequency log-magnitude short-time Fourier transform* is used. A window size of 2048 samples and a hop size of

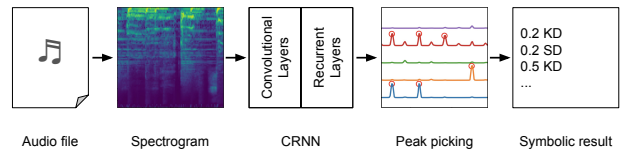


Figure 4. Overview of the automatic transcription.

441 samples are used for a target frame rate of 100Hz; the frequency bins are transformed to a logarithmic scale with 12 triangular filters per octave between 20 to 20,000Hz. The output of the network is an activation function displaying a value between zero and one for the five classes inferred. This value represents the confidence of the network that a note is played at this specific location. To extract a symbolic representation from these functions, a simple peak picking algorithm is used [17]. In the experiments, we carefully reproduced the whole procedure presented in Vogl’s work and we implemented this network architecture with Tensorflow.

5.2 Evaluation

We compare results for ADTOF with three other datasets (ENST, MDB, RBMA) that contain real music (see Table 1) and by mapping their vocabulary onto our five classes. To carry out the comparison, we trained the CRNN on three different (combinations of) datasets, resulting in three different versions of the model. The first version is trained on ENST, MDB, and RBMA, and represents a baseline. The second version is first trained on TMIDT, and then on ENST, MDB, and RBMA; this is our reproduction of the state-of-the-art method. The last version is trained exclusively on ADTOF. In Fig. 5, these three versions are represented by the blue, orange, and green bars, respectively.

For each version, we followed a three-fold cross-validation strategy: testing is done on one split of each dataset while training and validation are done on the remaining splits of the dataset(s) used for training. This methodology ensures that the test data stays consistent across all models. When training on multiple datasets at the same time, we merged the corresponding training and validation sets together. In practice, for ENST, MDB, and RBMA, we used the three splits from [8] by iteratively selecting one as test data and further partitioning the remaining two as 15% validation and 85% training data. Whenever drum solo version of the tracks were available, they were added with their full mix version as additional training material, but not when testing. In contrast, since ADTOF is much larger, we partitioned the data in ten splits without overlap of artists between them to prevent leakage of information on similar-sounding tracks. The splits are used by iteratively selecting one as test data, one as validation and the remaining eight as training data.

The performance metric used is the well-known F-measure, computed with the package *mir_eval* [18] with a tolerance window of 50 *ms* to stay consistent with previous research [1, 7]. We return values by counting the

⁸ Results for “RV3” at the URL https://www.music-ir.org/mirex/wiki/2018:Drum_Transcription_Results

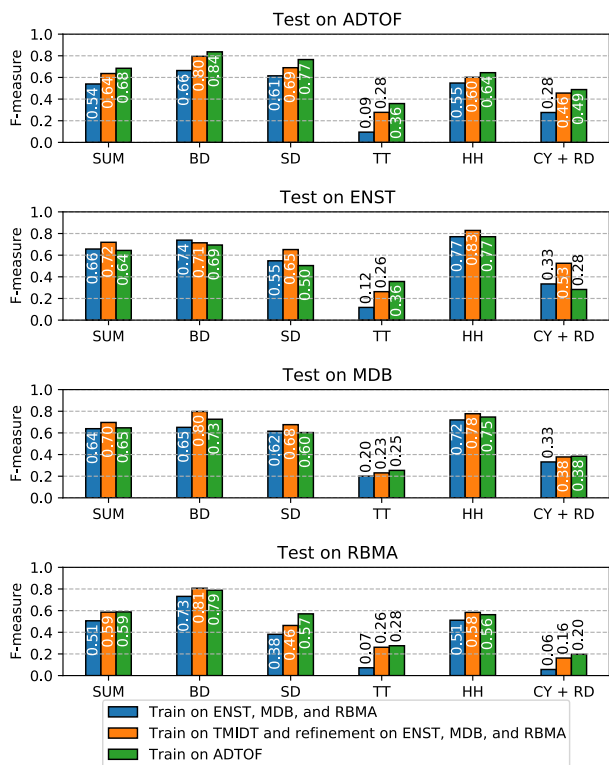


Figure 5. Plots representing the overall (SUM) and class-specific F-measure for multiple training and testing configurations.

class specific true positives, false positives and false negatives across all tracks. We also compute an overall “SUM” value by counting across all tracks and all classes. This process is repeated three times on the different test sets of the cross-validations and we report the averaged results.

6. RESULTS

The quality of the ADTOF dataset is assessed by (1) the performance that a state-of-the-art ADT algorithm achieves on it, and (2) the performance that a state-of-the-art ADT model trained on it achieves on multiple datasets. Fig. 5 depicts the results for both evaluations.

(1) In Fig. 5, the orange bars refer to the performance of the state-of-the-art algorithm, and different panels refer to different datasets. By comparing the orange bars in the top panel (ADTOF) and in the other three panels, we observe that the algorithm’s performance on our dataset is entirely comparable with that on the other datasets. In detail, the classes toms (TT) and crash + ride cymbals (CY + RD) are difficult to estimate both in our and the other datasets. This fact is well understood, and typically linked to the sparsity of these classes in the training data. We note a slight increase in the snare drum (SD) performance on ADTOF compared to the other datasets, possibly meaning that the snare sounds in our tracks are more easily identified by the model. On the contrary, the hi-hat cymbal (HH) score is rather low, suggesting that the specific combinations of drum hits meant to ease the gameplay (see Sec. 4.2) are not

fully corrected and harm the accuracy.

(2) We now focus on the performance achieved by one model when trained differently. The green bars refer to the model trained on ADTOF. Overall, our dataset trains the model equally well, and possibly better, than the current state of the art.

In detail, the model trained on ADTOF outperforms the baseline (blue bars) in the majority of the evaluations, indicating that the increased volume of the training data resulted in improved performance even on datasets unknown during training. Additionally, this fact suggests that the audio is representative of the task and enables generalization to other datasets. The model trained on ADTOF, and the model trained on TMIDT and refined on ENST, MDN, and RBMA, mostly outperform one another when evaluated on the same dataset(s) that was used for training; a few exceptions occur, in favor of the model trained on ADTOF. On the one hand, this is an indication that those datasets might be complementary, i.e., they all contribute useful information to the training. On the other hand, the exceptions in favor of ADTOF suggest that the inclusion of a larger number of real-world tracks results in better generalization capability of the model.

7. CONCLUSIONS

One of the main problems in ADT is to find a high volume of good quality data that the models can use during training. Unlike the most recent approaches that rely on synthesized and/or augmented audio to gather enough data, we proposed a new technique that leverages annotated tracks for rhythm video games. This source of data, to our knowledge unexplored until now in ADT, has the advantage of both containing authentic audio and being available online in a large quantity, thanks to the annotation process realized by a broad community of players. However, since the annotations originate from people with different expertise and are initially meant for a different purpose than ADT, they are not readily usable. Therefore, we illustrated a method to automatically select, transform, and clean the data. We further demonstrated that an algorithm trained on this dataset generalizes well and achieves state-of-the-art results, which is an indication that our method of gathering and curating the data is successful and should be further explored.

In future works, we are interested in evaluating repositories of custom charts other than “Rhythm Gaming World” in order to identify which of them is more suited to be used to build a dataset with our methodology. As an example, since the original tracks included in the video game RockBand are professionally annotated by the Harmonix gaming company, they may constitute a good quality set (because of better annotations) to explore.

8. REFERENCES

[1] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Muller, and

- A. Lerch, “A Review of Automatic Drum Transcription,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1457–1483, Sep. 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8350302/>
- [2] O. Gillet and G. Richard, “ENST-Drums: an extensive audio-visual database for drum signals processing,” in *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR)*, Victoria, Canada, 2006, pp. 156–159.
- [3] C. Southall, C.-W. Wu, A. Lerch, and J. Hockman, “MDB drums- An annotated subset of MedleyDB for Automatic Drum Transcription,” in *Late Breaking/Demos of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017.
- [4] R. Vogl, M. Dorfer, G. Widmer, and P. Knees, “Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 150–157.
- [5] R. Vogl, M. Dorfer, and P. Knees, “Drum transcription from polyphonic music with recurrent neural networks,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 201–205. [Online]. Available: <http://ieeexplore.ieee.org/document/7952146/>
- [6] C. Jacques and A. Roebel, “Data Augmentation for Drum Transcription with Convolutional Neural Networks,” *arXiv:1903.01416 [cs, eess]*, Mar. 2019. [Online]. Available: <http://arxiv.org/abs/1903.01416>
- [7] M. Cartwright and J. P. Bello, “Increasing Drum Transcription Vocabulary Using Data Synthesis,” in *Proceedings of the International Conference on Digital Audio Effects (DAFx-18)*, 2018, pp. 72–79.
- [8] R. Vogl, G. Widmer, and P. Knees, “Towards multi-instrument drum transcription,” *arXiv:1806.06676 [cs, eess]*, Jun. 2018. [Online]. Available: <http://arxiv.org/abs/1806.06676>
- [9] J. Driedger and H. Schreiber, “Towards automatically correcting tapped beat annotations for music recordings,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 200–207.
- [10] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, “MedleyDB: a multitrack dataset for annotation-intensive MIR research,” in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 155–160.
- [11] C. Southall, R. Stables, and J. Hockman, “Automatic Drum Transcription for Polyphonic Recordings Using Soft Attention Mechanisms and Convolutional Neural Networks,” in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 606–612.
- [12] R. Stables, J. Hockman, and C. Southall, “Automatic Drum Transcription using Bi-directional Recurrent Neural Networks,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, United States, Aug. 2016, pp. 591–597.
- [13] O. Gillet and G. Richard, “Transcription and Separation of Drum Signals From Polyphonic Music,” *IEEE Trans. Audio Speech Lang. Process.*, vol. 16, no. 3, pp. 529–540, Mar. 2008. [Online]. Available: <http://ieeexplore.ieee.org/document/4443887/>
- [14] RockBand, “5 Things You Should Know About PDP’s NEW Drums For Rock Band Rivals,” 2016. [Online]. Available: <https://youtu.be/GTYMOtCPN1E>
- [15] S. Böck, F. Krebs, and G. Widmer, “Joint Beat and Downbeat Tracking with Recurrent Neural Networks,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, United States, Aug. 2016, pp. 255–261. [Online]. Available: <https://doi.org/10.5281/zenodo.1415836>
- [16] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: A New Python Audio and Music Signal Processing Library,” in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1174–1178. [Online]. Available: <http://dl.acm.org/citation.cfm?doid=2964284.2973795>
- [17] S. Böck, F. Krebs, and M. Schedl, “Evaluating the Online Capabilities of Onset Detection Methods,” in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, Oct. 2012, pp. 49–54. [Online]. Available: <https://doi.org/10.5281/zenodo.1416036>
- [18] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, “mir_eval: a transparent implementation of common MIR metrics,” in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 367–372.

LEARN BY REFERENCING: TOWARDS DEEP METRIC LEARNING FOR SINGING ASSESSMENT

Huan Zhang¹ Yiliang Jiang² Tao Jiang² Peng Hu²

¹ School of Music, Carnegie Mellon University, Pittsburgh, US

² Tencent Music Entertainment, Shenzhen, China

huanz@andrew.cmu.edu yiljiang@tencent.com marsjiang@tencent.com stevenhu@tencent.com

ABSTRACT

The excellence of human singing is an important aspect of subjective, aesthetic perception of music. In this paper, we propose a novel approach to tackle Automatic Singing Assessment (ASA) task through deep metric learning. With the goal of retrieving the commonalities of good singing without explicitly engineering them, we force a triplet model to map perceptually pleasant-sounding singing performance closer to the reference track compared to others, and thus learning a joint embedding space with performance characteristics. Incorporating mid-level representations like *spectrogram* and *chroma*, this approach takes advantage of the feature learning ability of neural networks, while using the reference track as an important anchor. On our designed testing set that spans across various styles and techniques, our model outperforms traditional rule-based ASA systems.

1. INTRODUCTION

Automatic Singing Assessment (ASA) deals with the task of assessing singing performances based on audio recordings. Ever since the development of Karaoke, a popular entertainment form and practice means for singers, there has been a high demand for ASA systems that's able to judge the excellence of singing performance just like human experts do.

However, ASA is not an easy task. Singing quality is often judged with respect to professional standards, where music experts rate singing performances based on their music knowledge and perceptual appeal. These dimensions include basic, objective criteria such as vowel quality (proper pronunciation of lyrics), accuracy of pitch and rhythm. Meanwhile, higher-level, subjective dimensions like singers formant, dynamics and expressiveness, techniques like vibrato and breathing are also taken into account.

Based on these criteria, some ASA systems compare a singing performance with a reference such as a profes-

sional singing performance [1,2] or melody contours [3,4], and thus place more emphasis on accuracy and intonation. On the other hand, unreferenced ASA systems aims to evaluate singing quality based on only the performance itself, addressing voice-related characteristics like voice formants or expressiveness. However, all these rule-based evaluation systems from hand-crafted indicators can easily be song- and style-dependent, resulting in poor generalizability. What's worse is that sometimes human perception standards seem to conflict with each other. For example, a good vibrato technique implies pitch instability, and an ASA system that's focused on intonation will fail on certain songs while human can easily perceive the trick. Given the vast dimensions of our perception space of singing quality, we seek data-driven, deep learning solutions.

In this work, we aim to tackle the ASA task through metric learning. With the goal of retrieving the commonalities of good singing without explicitly engineering them, we force the model to map perceptually pleasant sounding performances closer to the reference track compared to others, and thus learning a joint embedding space with performance characteristics. Section 2 reviews the related works. In Section 3, we first present our audio representations that incorporate multi-channel features. Then we introduce our proposed triplet network in comparison to other skeleton models for assessing singing quality. Experiment results and discussions are shown in Section 4, where our proposed architecture achieved the highest correlation with human perception of singing rating on the mixed testing set, when comparing with existing singing assessment algorithms.

2. RELATED WORK

Currently, the majority of literature on ASA systems, whether referenced or unreferenced, largely focused on extraction of perceptually motivated features such as f_0 pitch sequences [3], f_0 pitch histogram [5], vibrato rate (periodic fluctuation of f_0) [6]. The main issue of these hand-crafted features is that they only reflect specific aspects of the singing. Thus, some of the systems [1, 7, 8] will then feed features into simple machine learning regressors or classifiers which predict ratings that take advantage of multiple features.

With the rise of deep learning, deep neural network is



found to outperform traditional methods in terms of feature learning. To our knowledge, the only work that tackles ASA task though end-to-end deep learning without feature engineering step is [9], which takes in a mid-level time-frequency representation of singing clips and evaluate singing as binary (good-poor) classification with a bi-dense neural network. However, this approach does not incorporate the song reference melody, and thus only assesses the discernible features which are independent of the particular singer or melody.

How to make use of the song reference information while taking advantage of deep learning? We observe that the technique of deep metric learning has recently attracted much attention in various MIR tasks. As a method of learning discriminative features by measuring similarities from samples, most existing studies used Siamese and Triplet networks to correlate among samples while using shared weights [10]. In [11–13], these architectures has been applied to tasks of Singer Identification, Cover Song Detection and Singing Style Investigation. What’s more, [14] assesses woodwind instruments performances via a joint embedding network, confirming the feasibility of learning a performance-reference joint latent space. In [15], an attempt with deep metric learning was used for assigning scores for singing tracks, but the foundation was laid with a classification network.

Compared to previous works, the novelty of our work is in two folds: **1)** We take a deep metric learning approach for singing assessment, which utilizes reference tracks (original, accompaniment) while not being constrained by it. **2)** Besides classical time-frequency representation of audio, we propose a set of mid-level audio representations which concerns pitch, tempo, timbre and so on. At the end, we construct a style-mixture testing set that comprehensively evaluates systems’ ability in assessing different dimensions of singing, and provide a detailed discussion.

3. METHODOLOGY

3.1 Audio Representation

Given that we are not explicitly engineering perceptually motivated features, it is important to present neural network with a comprehensive view of the audio data. In such comprehensive task like singing assessment, it is unusual to judge only the pitch accuracy of a performance and ignore tone production, or only care about hitting the right beat but not pronunciation of the lyrics. Thus, for input features, we employ five channels of 2-dimensional audio representations that concern with musical dimensions such as pitch, rhythm, timbre:

- i *Log-Mel Spectrogram (Spec)*: We extract the Log-Mel Spectrogram for each 3s segment with a hop size of 512. Given 16kHz sampled audio, the resulting representation contains 94 frames and 96 bins.
- ii *Chroma (Chroma)*: Chromagram gives us the pitch class profiles for the clips. Note that in order to achieve the same dimension (96) with other channels, we give 6 bins ($96 \text{ bins} / (12 \text{ pitch classes})$) for

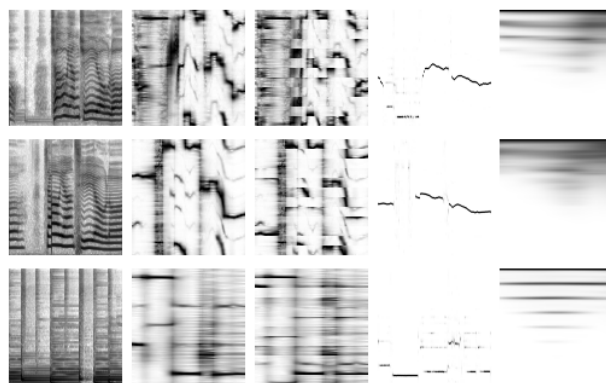


Figure 1. Top: *Spec*, *Chroma*, *TChroma*, *F0*, *Tempo* 2D visualization of a clip that’s labeled as ‘good’; **Mid**: another clip that’s labeled as ‘bad’ at the same timestamp; **Bottom**: features from their corresponding accompaniment clip.

each pitch class.

- iii *Tonal Shifted Chroma (TChroma)*: From a functional harmony perspective, pitch *G* and *C* are closer while *C♯* and *C* differs more. Thus, in this channel we take inspiration from [16] and rearrange the rows of chromagram by circle of fifths, in the hope that the features are more sensitive to non-harmonic mistakes.
- iv *F0 (F0)*: As seen in Section 2, f_0 is the most crucial feature for assessing intonation. Pitch extraction algorithm Crepe [17] is used to obtain an activation matrix of estimated pitch.
- v *Tempogram(Tempo)*: We also extracts the cyclic tempogram [18] of the clip, representing the estimated tempo that evolves over time.

We adopted Librosa’s [19] implementation for all of the audio features above except F0. In [20], it was studied that a short voiced sequence (3-5 sec.) is sufficient for assessing singing quality. Thus, for all of input audio we extract the features above from 3s clips, with 96×94 in dimension concatenated together like a 5-channel image. See Figure 1 for comparison of the 5 channel representations of a pair of 3s clips and their corresponding reference clip.

3.2 Architectures

For the schematic model for embedding, we adopted the CNNSA (CNN + Self Attention) model proposed in [21, 22]. Originally designed for music-tagging task, the architecture achieved compelling results in learning local characteristics and temporal representations via interpretable attention maps.

Structurally, the model employs a front-end / back-end division of deep neural network for MIR task that was first proposed by [23] (Figure 2): The front-end consists of a 7-layer CNN with (3×3) filters with residual connections [24], aiming to extract local information such as timbre and pitch. The back-end utilizes stacks of self-attention layers to achieve temporal summarization like rhythmic patterns,

melodic contours, and chord progressions. Here, the self-attention layers are BERT [25] encoders where Q, K, V are feature maps obtained from front-end.

3.2.1 Baseline: Direct Score

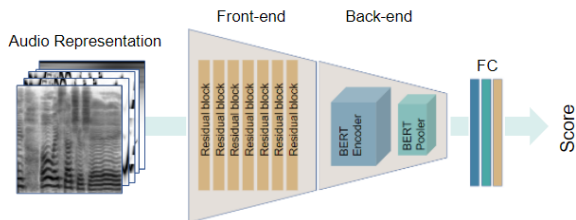


Figure 2. Baseline architecture of deriving score from audio representation, without referencing or contrasting.

As shown in Figure 2, our baseline model is a direct regression model that learns a mapping between audio representations and the labeled score directly. No reference is used. For the architecture, the five-channel audio representation is fed into the schematic *CNN + self attention* model mentioned in 3.2, with 3 fully connected layers at the end to output a score that measures the excellence of singing.

3.2.2 Delta Spectrogram

The "delta spectrogram" (Delta) model is a natural extension of the baseline architecture with a reference clip: Given the audio representation, we subtract it from the audio representation of reference clip. Given most of our audio inputs have time-frequency attributes, a singing clip misaligned with pitch or rhythm will be reflected in its delta with reference clip. In comparison with other architectures, this method is not attempting to learn a joint latent space, and can be viewed as extracting local similarities at the front of the pipeline.

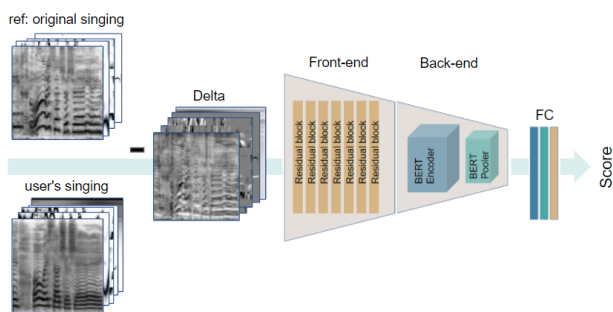


Figure 3. The Delta model, where the notion of distance with anchor is incorporated as delta at the front of the pipeline.

3.2.3 Triplet:

The triplet model is our main proposed architecture for learning a joint embedding space with singing characteristics. The objective of this architecture is to learn a mapping from audio representation to an embedding space where

good singings are closer to reference while poor singings are further away in this space. We implement this idea via a triplet network shown in Figure 4, which takes in (reference (as anchor), good singing (as positive), poor singing (as negative)).

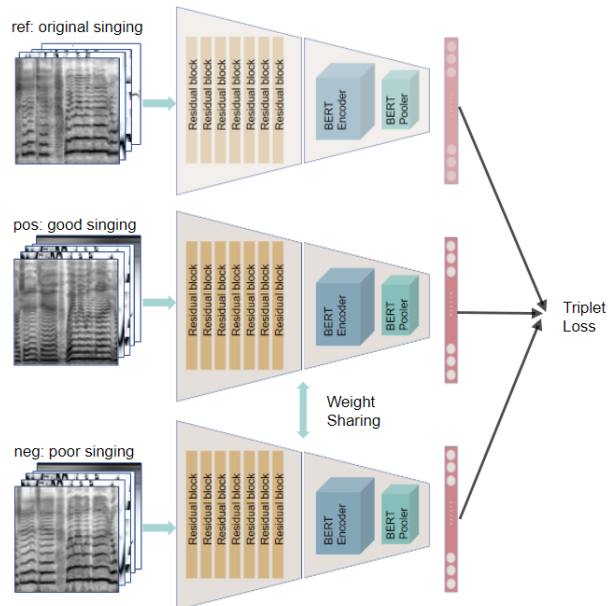


Figure 4. Metric learning architecture.

All three towers have the same architecture, but the two towers of positive and negative singing clips enabled weight-sharing while the reference tower doesn't. This is because the performance tracks and reference tracks came from different audio domains, and thus benefits from projecting to different embedding spaces. Afterwards, a 128-dimensional embedding for the performance is learnt through the model, where we optimize toward Triplet Margin Loss with Cosine Similarity as distance.

The Triplet Margin Loss is computed by

$$L(a, p, n) = \max\{D(a, n) - D(a, p) + \alpha, 0\}$$

where a, p, n denotes *anchor, positive, negative* respectively, and α is the margin. For the distance metric D we utilize the inverse of Cosine Similarity, as a larger similarity represents smaller distance:

$$D(x, y) = -\frac{x \cdot y}{\|x\| \cdot \|y\|}$$

Given the 128-dimension output of the model, the final assessment output is the cosine similarity between the embedding of the given performance and the embedding of its reference track. Thus, the score assesses the proximity of the performance towards reference. For the triplet architecture, we utilized both original singing and accompaniment as reference tracks, and the experiments are denoted by R_{ori} (reference with original) and R_{acc} (reference with accompaniment).

3.2.4 Embedding Direct Score

With the belief that the joint latent space encodes the characteristics of commonalities of good and poor singing, we also trained an embedding direct score (Emb_{Direct}) that takes the assessment as a downstream task after a singing embedding is learnt from Section 3.2.3.

Specifically, for a given clip, we take the R_{ori} model pre-trained in previous section and output the clip embedding of 128 dimension. In this system, we train three fully connected layers with ReLU activation to regress and match the labelled score. In inference, the final score is obtained from the audio clip through the embedding and final output layers directly. Thus, this is also an unreferenced system.

3.3 Data

The training data are collected from the solo singing clips without accompaniment from WeSing¹. The singers are volunteers who were the common users of this application.

For reference tracks used in metric learning, we perform experiments with both *accompaniment* tracks and *original singing* tracks, which are the pure singing voice from the published version of song. The accompaniments are purchased by WeSing from music production studios, and original singing tracks are source-separated by Spleeter [26].

After making sure that the reference track and solo singing track are exactly aligned, WebRTC vad [27] is used to detect the voiced segments from the singing clip. Each 3s segment from the voiced segment, along with the referenced track clip on the same position, is extracted into audio representation in Section 3.1. Our data preprocessing pipeline is shown in Figure 5.

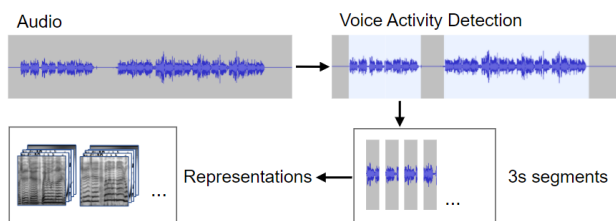


Figure 5. Preprocessing pipeline

Another question is on how we obtain the "good" and "poor" singing for deep metric learning. For the clips used in our training, a quality score within [0, 100] was labeled by the company’s contractor employees. Note that these quality scores are very rough as they come from multiple people’s standards without a detailed listening, and there is no guarantee for their coherency. We take all clips with scores ≥ 80 as "good" and < 40 as "poor". In the mash-up, we randomly align a "good" performance and a "bad" performance of the same song to form a contrasting pair.

We believe the quality score is not exact in modeling the excellence of singing, but gives a rough direction on what’s good and what’s poor. Meanwhile, the weak labelling is

¹ <https://wesingapp.com/> is a Karaoke application.

actually advantageous to our exploration as they represent a general perception and prevents our models to overfit to any specific assessment standard.

In total, we obtained 15487 3s singing clips pairs, and an equal number of positive and negative data are exactly aligned. These clips are obtained from 1240 full length recordings and from 102 songs. All clips were resampled to 16K Hz. In terms of genre, the songs roughly consists of 75% of Chinese pop with a variety of tempo and style (published after 2000), 15% of folk songs, 5% of rock, and 5% of other genres. There are no jazz or classical singing styles in the training set.

The testing set we designed will be introduced in Section 4.

3.4 Experiments

We trained our models on 3 NVIDIA V100 GPUs on a single machine. For the CNN models, the optimizer of the triplet loss is ADAM [28] with a learning rate 10^{-6} . For Emb_{Direct} system, we used learning rate 10^{-7} as it only trains 3 linear layers. We used batch size of 128 and trained our models for a maximum of 200 epochs with early stopping based on validation loss with patience of 5 epochs. We choose $\alpha = 1$ for the margin in Triplet loss.

Song	Genre	Character	BPM	Ori.singing
茧 (Cocoon)	pop	lyrical, spans over large range	78	nasal, genderless voice; young male
夜夜夜漫长 (Night is Long)	electronic, pop	rhythmical, low register, rap-like, energetic	98	husky, hoarse; middle-aged male
白月光与朱砂痣 (White Moonlight and Cinnabar Nevus)	pop	simple melody and slow harmonic rhythm, small range	89	deep, melodious; middle-aged female
Love Story	country, pop	high tempo, high note density, short notes and syllables	120	silvery, sweet; young female
今夜草原有雨 (It's Gonna Rain in the Steppe Tonight)	Chinese folk-style, pop	high register, a lot of vibrato, extremely expressive	62	operatic, soprano; middle-aged female

Table 1. Characteristics and style analysis of the songs in testing set.

4. EVALUATION AND DISCUSSION

Our testing set consists of 5 songs with different styles, that spans over genres like pop, electronic, country and folk, as summarized in Table 1². The dataset consists of a mix of songs that spans over various registers, tempo, techniques and even language; they also attracts different cultural and age groups. For each of the 5 songs we subjectively choose 9 different performances with various quality, creating a testing set of 45 recordings.

² bpm of the songs are estimated using Madmom [29]

	Configuration	Reference Used	Song1	Song2	Song3	Song4	Song5	Mix
Baseline	Direct	None	0.785	0.223	0.472	0.528	0.253	0.417
Proposed	Delta	Original Track	0.718	0.383	0.684	0.288	0.276	0.430
Proposed	Rori	Original Track	0.912	0.860	0.521	0.839	0.480	0.652
Proposed	Racc	Accompaniment Track	0.635	0.708	0.853	0.663	-0.222	0.459
Proposed	EmbDirect	None	0.861	0.503	0.256	0.487	0.753	0.533
Histogram	peakBW	None	0.875	0.581	0.714	0.872	0.281	0.626
Histogram	peakConc50	None	0.651	0.836	0.822	0.736	0.407	0.520
Histogram	binning	None	0.874	0.776	0.688	0.882	-0.206	0.521
DTWDist	pitch	MIDI	0.812	0.732	0.907	0.324	0.068	0.589
DTWDist	volume	Energy Sequence	0.723	0.524	0.761	0.468	0.432	0.467
DTWDist	rhythm	MIDI	0.361	0.542	0.606	0.042	0.043	0.279

Table 2. Pearson’s correlation between human scoring and algorithm scoring. Correlations among each individual song and mixture of all songs are shown.

4.1 Subjective Ground Truths

For each of the 45 recordings, we asked 5 professionals (graduates from music conservatoire with 10+ years of performance training) to assign them scores based on the quality of singing. Overall, the scores reached an average inter-judge correlation of 0.78, and thus we consider them as valid ground truth. For evaluation of the systems, we computed the Pearson’s correlation coefficient between the output scores from the algorithms and the human judge’s mean score.

4.2 Objective

As mentioned in Section 2, there are plenty of rule-based methods for singing assessment tasks. We implemented our version of two existing methods: the pitch histogram measures proposed in [5] and feature (pitch, dynamics, rhythm) evaluation method proposed in [3, 30].

4.2.1 Histogram

The pitch histogram method is an unreferenced system that computes a series of attributes for evaluating unaccompanied singing. With bins of 100 cents within an octave, pitch histogram represents the distribution of pitch values in a performance. Good singings will usually have sharp peaks on specific note values, while poor singings will have dispersed distribution as they sing out of tune. Thus, a series of measurements relating to the spread of peaks in the histogram can be used for evaluation. We computed scores like kurtosis, skewness, peakBW, peakConc50, peakConc110, kMeans, binning as described in [5], and listed the best 3 measurements in Table 2.

4.2.2 DTWDist

As specified in [3, 30], we also implemented traditional Pitch-based Rating, Volume-based Rating, Rhythm-based Rating. In general, these methods computes the **DTWDist** (Dynamic Time Wrapping Distance) between users’ performances with reference (MIDI note sequence for pitch and rhythm, recording energy sequence for volume), and more similar sequences indicates

better singing. We also presents the performance of these ratings in Table 2.

4.3 Results Discussion

Table 2 shows the comparative performances for all systems. We are able to make the following observations:

- i **Performances of deep learning systems.** Among all deep learning systems, our Rori system proved to correlate the most with human perception of singing ranking. The models (Direct, Delta) that don’t incorporate reference tracks were outperformed, showing that learning a joint latent space of singing quality indeed helps with our assessment goal. Between the deep metric learning systems, we noticed that Racc doesn’t perform as good as Rori, since the accompaniment track as anchor does not provide details on singing, but only helps with judging the rhythm and tonality.
- ii **Learnt embeddings are more robust to song variations.** For each individual song, there is at least one system that reaches 0.75 of correlation with human perception, but on the mixture of 5 songs the maximum correlation we can obtain is 0.65. This confirms the concern mentioned in Section 1, that it’s difficult for ASA systems to evaluate different songs on the same scale. The pitch histogram measures, while performing great among the rating within each songs, suffers when we cross compare performances from different songs: The ‘spikes’ within the histogram are influenced by the number of pitches used, and some songs are naturally going to achieve a higher score in their metrics. In comparison, deep metric learning approaches are more robust. See also Section 4.4 for more demonstrations.
- iii **Ability to evaluate on more nuanced techniques.** Song 5 is a Chinese folk style piece that demands singing techniques such as vibrato. It’s difficult to hand-craft perception motivated features for such techniques, and neither of the traditional feature-based method perform well on this piece. The

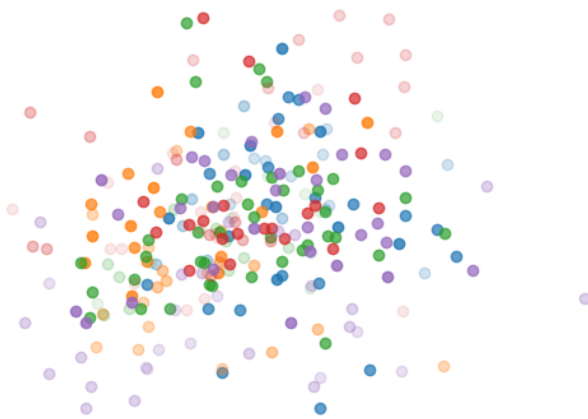


Figure 6. PCA Projection of clip embeddings from the testing set. 5 colors correspond to songs, and transparency is scaled with the human evaluation score of performance, where higher scores are darker. Best viewed in color.

deep learning based models *Rori*, *EmbDirect*, however, both outperform traditional methods on this song while retaining high performance on other songs. This confirms the ability of deep neural network to model performance-related features.

- iv **Does good implies similar?** The best performing model for Song 5 is actually *EmbDirect* system (Section 3.2.4), that assesses the singing through features learnt from the good-poor metric space without computing the similarity with original singing directly. Thus, we speculate that there is this conceptual gap between "Good" and "Similar": For a given piece, there are multiple ways of performing it nicely, and they don't necessarily need to be similar to the original. The *EmbDirect* system demonstrates that the learnt joint embedding space encodes singing characteristics and can be used to tell good or poor singing apart - to learn by reference while not being constrained by it.

4.4 Embedding Space Visualization

In Figure 6, we take all the 3s clips from 45 singing performances in the testing set, and project their 128-dimensional embeddings by PCA. The embedding is trained from our *Rori* configuration that obtained the best cross-song result in Table 2, contrasting good-poor performance with original singing. 5 songs are distinguished by colors, while transparency represents the assessment from ground truth human judgements, where poorer singing has a higher transparency.

The embedding space visualization supports our observations in Section 4.3: In the vocal embeddings, song differences are largely eliminated, while the clips from higher scored performances tends to cluster in the middle and lower scored performances are dispersed. This demonstrates that, regardless of songs and references, the embeddings capture certain universal characteristics in distin-

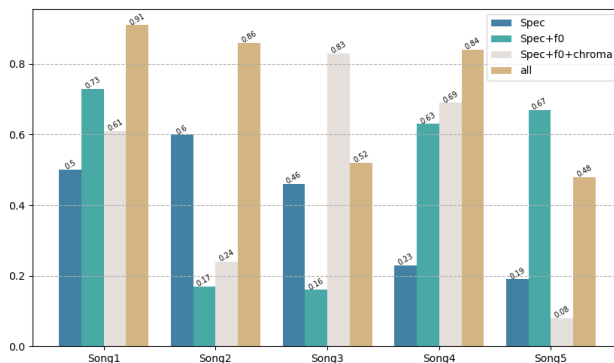


Figure 7. Comparison between different input audio representations.

guishing excellency of singing voice.

4.5 Ablation Study for Input Audio Representation

To demonstrate the effect of our multi-channel audio representation described in Section 3.1, ablation experiments were performed to show that the combination of audio representation indeed achieved better assessment results. For the *Rori* architecture, we performed experiments using 4 combinations: *Spec*, *Spec+f0*, *Spec+f0+chroma*, and all 5 representations together.

Figure 7 shows the performance of different audio representation input on 5 songs respectively. Overall, the input that utilized all 5 channels achieved the best result. Given that Convolutional Neural Network is still one of the most popular architecture for 2D audio feature learning nowadays, this idea of presenting a multi-channel view to deep networks may be applied to other interesting tasks.

5. CONCLUSION AND FUTURE WORK

This paper presents a novel approach of automatic singing assessment task via metric learning. Through training a triplet model that anchors at a reference track of the performance, we were able to learn a joint embedding space where characteristics of good and poor singing were extracted. Comparative experiments were performed on a designed testing set that evaluates assessment systems across variety of singing styles and techniques. Results demonstrate that the proposed system outperforms baseline and feature-based assessment systems in cross-song ratings when correlates with human judgments.

Given the intrinsic subjective aspect of human perception of music performance, singing assessment as well as broader music assessment has been little investigated through deep learning approach. Our work demonstrates that it's possible to direct deep neural network in learning performance related characteristics via comparing weakly labeled data. Future explorations may expand on this "learn by reference" idea with other paradigm such as contrastive learning [31], or apply to neighboring domains like instrumental assessment. Also, we wish our experiments with multi-channel audio representations would facilitate more explorations in musically-motivated input design.

6. REFERENCES

- [1] C. Gupta, H. Li, and Y. Wang, "Perceptual evaluation of singing quality," in *Asia-Pacific Signal and Information Processing Association*, 12 2017.
- [2] P. Lal, "A comparison of singing evaluation algorithms," in *INTER_SPEECH 2006 and 9th International Conference on Spoken Language Processing, INTER_SPEECH 2006 - ICSLP*, vol. 5, 01 2006.
- [3] W. Tsai and H. Lee, "Automatic evaluation of karaoke singing based on pitch, volume, and rhythm features," *IEEE Trans. Speech Audio Process.*, vol. 20, no. 4, pp. 1233–1243, 2012. [Online]. Available: <https://doi.org/10.1109/TASL.2011.2174224>
- [4] E. Molina, I. Barbancho, E. Gómez, A. M. Barbancho, and L. J. Tardón, "Fundamental frequency alignment vs. note-based melodic similarity for singing voice assessment," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 744–748.
- [5] C. Gupta, H. Li, and Y. Wang, "Automatic evaluation of singing quality without a reference," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 11 2018, pp. 990–997.
- [6] T. Nakano, M. Goto, and Y. Hiraga, "An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features," in *INTER_SPEECH 2006 and 9th International Conference on Spoken Language Processing, INTER_SPEECH 2006 - ICSLP*, vol. 4, 01 2006.
- [7] Bar, Bozkurt, O. Baysal, and D. Yüret, "A dataset and baseline system for singing voice assessment," in *13th Int. Symposium on Computer Music Multidisciplinary Research*, 2017.
- [8] J. Böhm, F. Eyben, M. Schmitt, H. Kosch, and B. Schuller, "Seeking the superstar: Automatic assessment of perceived singing quality," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1560–1569.
- [9] N. Zhang, T. Jiang, F. Deng, and Y. Li, "Automatic singing evaluation without reference melody using bi-dense neural network," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 466–470.
- [10] M. Kaya and H. . Bilge, "Deep metric learning: A survey," *Symmetry*, vol. 11, p. 1066, 2019.
- [11] K. Lee and J. Nam, "Learning a joint embedding space of monophonic and mixed music signals for singing voice," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, Netherland, 2019.
- [12] M. Stamenovic, "Towards cover song detection with siamese convolutional neural networks," 2020. [Online]. Available: <https://arxiv.org/abs/2005.10294>
- [13] C. Wang and G. Tzanetakis, "Singing style investigation by residual siamese convolutional neural networks," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 116–120, 2018.
- [14] J. Huang, Y.-N. Hung, A. Pati, S. K. Gururani, and A. Lerch, "Score-informed networks for music performance assessment," in *21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.
- [15] T. Tan, "Singing evaluation based on deep metric learning," in *Proceedings of the 2019 3rd International Symposium on Computer Science and Intelligent Control*, ser. ISCSIC 2019. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: <https://doi.org/10.1145/3386164.3389096>
- [16] R. Abecidan, M. Giraud, and G. Micchi, "Towards Custom Dilated Convolutions on Pitch Spaces," *International Society for Music Information Retrieval Conference (ISMIR 2020)*, Late-Breaking Demo Session, 2020, poster. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-02959676>
- [17] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 161–165.
- [18] P. Grosche, M. Müller, and F. Kurth, "Cyclic tempo-pograma mid-level tempo representation for musicsignals," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, 04 2010, pp. 5522 – 5525.
- [19] B. McFee, V. Lostanlen, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, J. Mason, D. Ellis, E. Battenberg, S. Seyfarth, R. Yamamoto, K. Choi, viktorandreevichmorozov, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, and T. Kim, "librosa/librosa: 0.8.0," Jul. 2020. [Online]. Available: <https://doi.org/10.5281/zenodo.3955228>
- [20] T. Nakano, M. Goto, and Y. Hiraga, "Subjective evaluation of common singing skills using the rank ordering method," in *9th International Conference on Music Perception and Cognition*, 2006.
- [21] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

- [22] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, “Evaluation of cnn-based automatic music tagging models,” in *Proc. of 17th Sound and Music Computing*, 2020.
- [23] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, “End-to-end learning for music audio tagging at scale,” *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, vol. abs/1711.02520, 2018.
- [24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 06 2016, pp. 770–778.
- [25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2019.
- [26] R. Hennequin, A. Khlif, F. Voituret, and M. Mousallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models,” *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020, deezer Research. [Online]. Available: <https://doi.org/10.21105/joss.02154>
- [27] A. B. Johnston and D. C. Burnett, *WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web*. St. Louis, MO, USA: Digital Codex LLC, 2012.
- [28] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, 12 2014.
- [29] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, “madmom: a new Python Audio and Music Signal Processing Library,” in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.
- [30] W. Tsai, C. Ma, and Y. Hsu, “Automatic singing performance evaluation using accompanied vocals as reference bases,” *Journal of Information Science and Engineering*, vol. 31, no. 3, pp. 821–838, 2015. [Online]. Available: http://www.iis.sinica.edu.tw/page/jise/2015/201505_04.html
- [31] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 1597–1607. [Online]. Available: <http://proceedings.mlr.press/v119/chen20j.html>

ACCOMONTAGE: ACCOMPANIMENT ARRANGEMENT VIA PHRASE SELECTION AND STYLE TRANSFER

Jingwei Zhao

Music X Lab, NYU Shanghai
jz4807@nyu.edu

Gus Xia

Music X Lab, NYU Shanghai
gxia@nyu.edu

ABSTRACT

Accompaniment arrangement is a difficult music generation task involving intertwined constraints of melody, harmony, texture, and music structure. Existing models are not yet able to capture all these constraints effectively, especially for long-term music generation. To address this problem, we propose *AccoMontage*, an accompaniment arrangement system for *whole pieces* of music through unifying phrase selection and neural style transfer.¹ We focus on generating piano accompaniments for folk/pop songs based on a lead sheet (i.e., melody with chord progression). Specifically, *AccoMontage* first retrieves phrase montages from a database while recombining them structurally using dynamic programming. Second, chords of the retrieved phrases are manipulated to match the lead sheet via style transfer. Lastly, the system offers controls over the generation process. In contrast to pure learning-based approaches, *AccoMontage* introduces a novel hybrid pathway, in which rule-based optimization and deep learning are both leveraged to complement each other for high-quality generation. Experiments show that our model generates well-structured accompaniment with delicate texture, significantly outperforming the baselines.

1. INTRODUCTION

Accompaniment arrangement refers to the task of reconceptualizing a piece by composing the accompaniment part given a lead sheet (a lead melody with a chord progression). When designing the texture and voicing of the accompaniment, arrangers are simultaneously dealing with the *constraints* from the original melody, chord progression, and other structural information. This constrained composition process is often modeled as a *conditioned generation problem* in music automation.

Despite recent promising advances in deep music generative models [1–7], existing methods cannot yet generate musical accompaniment while capturing the aforementioned constraints effectively. Specifically, most al-

gorithms fall short in preserving the fine granularity and structure of accompaniment in the long run. Also, it is difficult to explicitly control the generation process. We argue that these limits are mainly due to the current *generation from scratch* approach. In composition practice, however, arrangers often resort to existing pieces as accompaniment references. For example, a piano accompanist can improvise through off-the-shelf textures while transferring them into proper chords, which is essentially re-harmonizing a reference piece to fit a query lead sheet. In this way, the coherence and structure of the accompaniment are preserved from the reference pieces, and musicians also have control over what reference to choose.

To this end, we contribute *AccoMontage*, a *generalized template-based* approach to 1) given a lead sheet as the query, search for proper accompaniment phrases as the reference; 2) re-harmonize the selected reference via style transfer to accompany the query. We model the search stage as an optimization problem on the graph, where nodes represent candidate phrases in the dataset and edges represent inter-phrase transitions. Node scores are defined in a rule-based manner to reveal query-reference fitness, while edge scores are learned by contrastive learning to reveal smoothness of phrase transitions. As for the re-harmonization stage, we adopt the chord-texture disentanglement and transfer method in [8, 9].

The current system focuses on arranging piano accompaniments for a full-length folk or pop song. Experimental results show that the generated accompaniments not only harmonize well with the melody but also contain more intra-phrase coherence and inter-phrase structure compared to the baselines. In brief, our contributions are:

- **A generalized template-based approach:** A novel hybrid approach to generative models, where searching and deep learning are both leveraged to complement each other and enhance the overall generation quality. This strategy is also useful in other domains.
- **The *AccoMontage* system:** A system capable of generating long-term and structured accompaniments for full-length songs. The arranged accompaniments have state-of-the-art quality and are significantly better than existing pure learning-based and template-based baselines.
- **Controllable music generation:** Users can control the generation process by pre-filtering of two texture features: rhythm density and voice number.

¹ Codes and demos at <https://github.com/zhaojw1998/AccoMontage>.



2. RELATED WORK

We review three topics related to symbolic accompaniment arrangement: conditional music generation, template-based arrangement, and music style transfer.

2.1 Conditional Music Generation

Conditional music generation takes various forms, such as generating chords conditioned on the melody [10, 11], generating melody on the underlying chords [6, 7], and generating melody from metadata and descriptions [12]. In particular, accompaniment arrangement refers to generating accompaniment conditioned on the lead sheet, and this topic has recently drawn much research attention. We even see tailored datasets for piano arrangement tasks [13].

For accompaniment arrangement, existing models that show satisfied arrangement quality typically apply only to *short* clips. GAN and VAE-based models are used to maintain inter-track music dependency [14–16], but limit music generation within 4 to 8 bars. Another popular approach is to generate longer accompaniment in a seq2seq manner with attention [5, 6, 8], but can easily converge to repetitive textural patterns in the long run. On the other hand, models that arrange for complete songs typically rely on a library of fixed elementary textures and often fail to generalize [17–19]. This paper aims to unite both high-quality and long-term accompaniment generation in one system, where “long-term” refers to full songs (32 bars and more) with dependencies to intra-phrase melody and chord progression, and inter-phrase structure.

2.2 Template-based Accompaniment Arrangement

The use of existing compositions to generate music is not an entirely new idea. Existing template-based algorithms include learning-based unit selection [20, 21], rule-based matching [17, 18], and genetic algorithms [19]. For accompaniment arrangement, a common problem lies in the difficulty to find a perfectly matched reference especially when the templates contain rich textures with non-chordal tones. Some works choose to only use basic accompaniment patterns to avoid this issue [17–19]. In contrast, our study addresses this problem by applying the style transfer technique on a selected template to improve the fitness between the accompaniment and the lead sheet. We name our approach after *generalized* template matching.

2.3 Music Style transfer

Music style transfer [22] is becoming a popular approach for controllable music generation. Through music-representation disentanglement and manipulation, users can transfer various factors of a reference music piece, including pitch contour, rhythm pattern, chord progression, polyphonic texture, etc [1, 8]. Our approach can be seen as an extension of music style transfer in which the “reference search” step is also automated.

3. METHODOLOGY

The AccoMontage system uses a generalized template-based approach for piano accompaniment arrangement. The input to the system is a lead sheet of a complete folk/pop song with phrase labels, which we call a *query*. The search space of the system is a MIDI dataset of piano-arranged pop songs. In general, we can derive the chord progression and phrase labels of each song in the dataset by MIR algorithms. In our case, the chords are extracted by [13] and the phrases are labeled manually [23]. We refer to each phrase of associated accompaniment, melody, and chords as a *reference*. For the rest of this section, we first introduce the feature representation of the AccoMontage system in Section 3.1, and then describe the main pipeline algorithms in Section 3.2 and 3.3. Finally, we show how to further control the arrangement process in Section 3.4.

3.1 Feature Representation

Given a lead sheet as the query, we represent it as a sequence of ternary tuples:

$$q = \{(q_i^{\text{mel}}, q_i^{\text{chord}}, q_i^{\text{label}})\}_{i=1}^n, \quad (1)$$

where q_i^{mel} , the melody feature of query phrase i , is a sequence of 130-D one-hot vectors with 128 MIDI pitches plus a hold and a rest state [24]; q_i^{chord} , the chord feature aligned with q_i^{mel} , is a sequence of 12-D chromagram vectors [1, 2]; q_i^{label} is a phrase label string denoting within-song repetition and length in bar, such as A8, B8, etc. [23]. n is the number of phrases in lead sheet q .

We represent the accompaniment reference space as a collection of tuples:

$$r = \{(r_i^{\text{mel}}, r_i^{\text{chord}}, r_i^{\text{acc}})\}_{i=1}^N, \quad (2)$$

where r_i^{mel} , and r_i^{chord} are the melody and the chord feature of the i -th reference phrase, represented in the same format as in the query phrases; r_i^{acc} is the accompaniment feature, which is a 128-D piano-roll representation the same as [8]. N is the volume of the reference space.

3.2 Phrase Selection

Assuming there are n phrases in the query lead sheet, we aim to find a reference sequence:

$$\mathbf{x} = [x_1, x_2, \dots, x_n], \quad (3)$$

where we match reference x_i to the i -th phrase q_i in our query; $x_i \in r$ and has the same length as q_i .

Given the query’s phrase structure, the reference space forms a graph of layered structures shown as Figure 1. Each layer consists of equal-length reference phrases and consecutive layers are fully connected to each other. Each node in graph describes the fitness between x_i and q_i , and each edge evaluates the transition from x_i to x_{i+1} . A complete selection of reference phrases corresponds to a path that traverses through all layers. To evaluate a path, We design a *fitness model* and a *transition model* as follows.

dataset is limited. To maintain a better harmonic fitness, we resort to music style transfer.

3.3 Style Transfer

The essence of style transfer is to transfer the chord sequence of a selected reference phrase while keeping its texture. To this end, we adopt the chord-texture disentanglement VAE framework by [8]. The VAE consists of a chord encoder Enc^{chd} and a texture encoder Enc^{txt} . Enc^{chd} takes in a two-bar chord progression under one-beat resolution and exploits a bi-directional GRU to approximate a latent chord representation z_{chd} . Enc^{txt} is introduced in Section 3.2.2 and it extracts a latent texture representation z_{txt} . The decoder Dec takes the concatenation of z_{chd} and z_{txt} and decodes the music segment using the same architecture invented in [9]. Sustaining texture input and varying chords, the whole model works like a conditional VAE which re-harmonizes texture based on the chord condition.

In our case, to re-harmonize the selected accompaniment x_i^{acc} to query lead sheet q_i , the style transfer works in a pipeline as follows:

$$\begin{aligned} z_{chd} &= Enc^{chd}(q_i^{chord}), \\ z_{txt} &= Enc^{txt}(x_i^{acc}), \\ x'_i &= Dec(z_{chd}, z_{txt}), \end{aligned} \tag{10}$$

where x'_i is the re-harmonized result. The final accompaniment arrangement result is $\mathbf{x}^{*'} = [x_1^{*'}, x_2^{*'}, \dots, x_n^{*'}]$.

3.4 Controllability

In the phrase selection stage, we essentially traverse a route on the graph. Intuitively, we can control generation of the whole route by assigning the first node. In our case, we filter reference candidates for the first phrase based on textual properties. The current design has two filter criteria: *rhythm density* and *voice number*. we define three intervals *low*, *medium*, and *high* for both properties and mask the references that do not fall in the expected interval.

- Rhythm Density (RD): the ratio of time steps with note onsets to all time steps;
- Voice Number (VN): the average number of notes that are simultaneously played.

4. EXPERIMENT

4.1 Dataset

We collect our reference space from POP909 dataset [13] with the phrase segmentation created by [23]. POP909 contains piano arrangements of 909 popular songs created by professional musicians, which is a great source of delicate piano textures. Each song has a separated melody, chord, and accompaniment MIDI track. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and quantize them at 16th notes (chords at 4th). This derives 857 songs segmented into 11032 phrases. As shown in Table 1, we have four-bar and

Table 1. Length Distribution of POP909 Phrase

bars	<4	4	5~7	8	>8
Phrases	1338	3591	855	3796	1402

eight-bar phrases in majority, which makes sense for popular songs. We also use POP909 to fine-tune our transition model, during which we randomly split the dataset (at song level) into training (95%) and validation (5%) sets.

At inference time, the query lead sheets come from the Nottingham Dataset [27], a collection of ~1000 British and American folk tunes. We also adopt $\frac{2}{4}$ and $\frac{4}{4}$ pieces quantized at 16th (chords at 4th). We label their phrase segmentation by hand, where four-bar and eight-bar phrases are also the most common ones.

4.2 Architecture Design

We develop our model based on the chord-texture disentanglement model proposed by [8], which comprises a texture encoder, a chord encoder, and a decoder. The texture encoder consists of a convolutional layer with kernel size 12×4 and stride 1×4 and a bi-directional GRU encoder [24]. The convolutional layer is followed by a ReLU activation [28] and max-pooling with kernel size 4×1 and stride 4×1 . The chord encoder is a bi-directional GRU encoder. The decoder is consistent with PianoTree VAE [9], a hierarchical architecture for polyphonic representation learning. The architecture of $Enc^{txt}(\cdot)$ in the proposed transition model is the same as the texture encoder illustrated above. We directly take the chord-texture disentanglement model with pre-trained weights as our style transfer model. We fine-tune the transition model with W_1 and W_2 in Eq (7) as trainable parameters.

4.3 Training

Our model is trained with a mini-batch of 128 piano-roll pairs for 50 epochs using Adam optimizer [29] with a learning rate from $1e-4$ exponentially decayed to $5e-6$. Note that each piano-roll pair contains 2 consecutive piano-rolls and 4 randomly sampled ones. We first pre-train a chord-texture disentanglement model and initialize $Enc^{txt}(\cdot)$ using weights of the texture encoder in the pre-trained model. Then we update all the parameters of the proposed transition model using contrastive loss \mathcal{L} in Eq (7). We set both α and β in Eq (4) to 0.5. During inference, we set δ and γ in Eq (9) to 0.3 and 0.7.

4.4 Generated Examples

To this end, we show two long-term accompaniment arrangement examples by the Accomontage system. The first one is illustrated in Figure 2, in which we show a whole piece (32-bar music) piano arrangement (the bottom two staves) based on the lead sheet (the top staff). We see that the generated accompaniment matches with the melody and has a natural flow on its texture. Moreover, it follows the A8A8B8B8 structure of the melody.



Figure 2. Accompaniment arrangement for *Castles in the Air* from Nottingham Dataset by AccoMontage. The 32-bar song has an A8A8B8B8 phrase structure which is captured during accompaniment arrangement. Second melodies and texture variations are also introduced to manifest music flow. Here we highlight some texture re-harmonization of 7th chords.

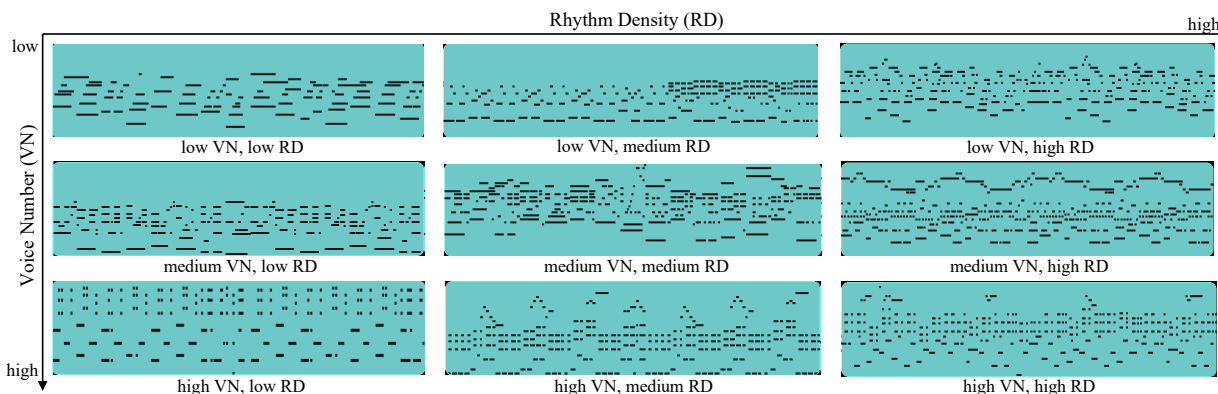


Figure 3. Pre-Filtering Control on Rhythm Density and Voice Number

The second example shows that our controls on rhythm density and voice number are quite successful. To better illustrate, we switch to a piano-roll representation in Figure 3, where 9 arranged accompaniments for the same lead sheet is shown in a 3 by 3 grid. The rhythm density control increases from left to right, while the voice number control increases from top to bottom. We can see that both controls have a significant influence on the generated results.

4.5 Evaluation

4.5.1 Baseline Models

The AccoMontage system is a generalized template-based model that leverages both rule-based optimization and deep learning to complement each other. To evaluate, we introduce a hard template-based and a pure learning-based baseline to compare with our model. Specifically, the base-

line model architectures are as follows:

Hard Template-Based (HTB): The hard template-based model also retrieves references from existing accompaniment, but directly applies them without any style transfer. It uses the same phrase selection architecture as our model while skipping the style transfer stage.

Pure Learning-Based (PLB): We adopt the accompaniment arrangement model in [8], a seq2seq framework combining Transformer [30] and chord-texture disentanglement. We consider [8] the current state-of-the-art algorithm for controllable accompaniment generation due to its tailored design of harmony and texture representations, sophisticated neural structure, and convincing demos. The input to the model is a lead sheet and its first four-bar accompaniment. The model composes the rest by predicting every four bars based on the current lead sheet and previous four-bar accompaniment.

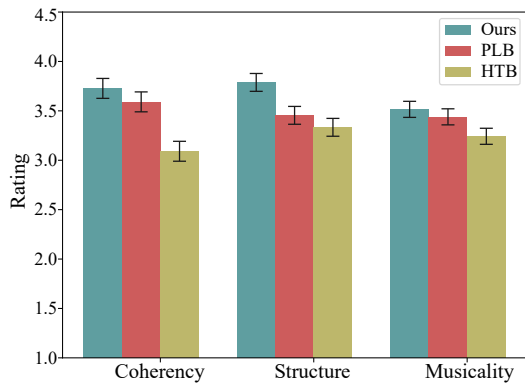


Figure 4. Subjective Evaluation Results.

4.5.2 Subjective Evaluation

We conduct a survey to evaluate the musical quality of the arrangement performance of all models. In our survey, each subject listens to 1 to 3 songs randomly selected from a pool of 14. All 14 songs are randomly selected from the Nottingham Dataset, 12 of which have 32 bars and the other two 24 and 16 bars. Each song has three accompaniment versions generated by our and the baseline models. The subjects are required to rate all three accompaniment versions of one song based on three metrics: coherence, structure, and musicality. The rating is base on a 5-point scale from 1 (very poor) to 5 (excellent).

- **Coherence:** If the accompaniment matches the lead melody in harmony and texture;
- **Structure:** If the accompaniment flows dynamically with the structure of the melody;
- **Musicality:** Overall musicality of accompaniment.

A total of 72 subjects (37 females and 35 males) participated in our survey and we obtain 67 effective ratings for each metric. As in Figure 4, the heights of bars represent the means of the ratings and the error bars represent the MSEs computed via within-subject ANOVA [31]. We report a significantly better performance of our model than both baselines in coherence and structure ($p < 0.05$), and a marginally better performance in musicality ($p = 0.053$).

4.5.3 Objective Evaluation

In the phrase selection stage, we leverage a self-supervised contrastive loss (Eq (7)) to enforce a smooth textural transition among reference phrases. We expect a lower loss for true adjacent phrase pairs than in other situations. Meanwhile, true consecutive pairs should hold a similar texture pattern with smaller differences in general properties.

We investigate the contrastive loss (CL) and the difference of rhythm density (RD) and voice number (VN) among three types of phrase pairs from the validation set. Namely, *Random*, *Same Song*, and *Adjacent*. Between totally randomly pairing and strict adjacency, *Same Song* refers to randomly selecting two phrases (not necessarily adjacent) from one song. Results are shown in Figure 5.

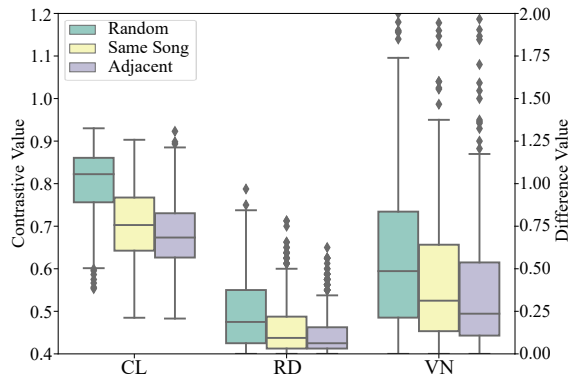


Figure 5. Evaluation of Transition Model. The contrastive loss (CL) and differences of RD and VN are calculated for three types of phrase pairs. A consistent decreasing trend illustrates reliable discernment of smooth transition.

Table 2. Ranking Accuracy and Mean Rank

Metric	Phrase Acc.	Song Acc.	Rank@50
Value	0.2425	0.3769	5.8003

For contrastive loss and each property, we see a consistent decreasing trend from *Random* to *Same Song* and to *Adjacent*. Specifically, we see the upper quartile of *Adjacent* is remarkably lower than the lower quartile of *Random* for CL, which indicates a reliable textural discernment that ensures smooth phrase transitions. This is also proved by the metric of ranking accuracy and mean rank [20], where the selection rank of the true adjacent phrase out of $k - 1$ randomly selected phrases (Rank@ k) is calculated. We follow [20] and adopt Rank@50, and the results are shown in Table 2. Phrase Acc. and Song Acc. each refers to the accuracy that the top-ranked phrase is *Adjacent* or belongs to the *Same Song*. The high rank of adjacent pairs illustrates our model’s reliability to explore smooth transitions.

5. CONCLUSION

In conclusion, we contribute a generalized template-based algorithm for the accompaniment arrangement problem. The main novelty lies in the methodology that seamlessly combines deep generation and search-based generation. In specific, searching is used to optimize the high-level structure, while neural style transfer is in charge of local coherency and melody-accompaniment fitness. Such a top-down hybrid strategy is inspired by how human musicians arrange accompaniments in practice. We aim to bring a new perspective not only to music generation, but to long-term sequence generation in general. Experiments show that our AccoMontage system significantly outperforms pure learning-based and template-based methods, being capable of rendering well-structured and fine-grained accompaniment for full-length songs.

6. ACKNOWLEDGEMENT

The authors wish to thank Yixiao Zhang for his contribution to figure framing and proofreading. We thank Liwei Lin and Junyan Jiang for providing feedback on initial drafts of this paper and additional editing.

7. REFERENCES

- [1] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," *arXiv preprint arXiv:1906.03626*, 2019.
- [2] K. Chen, G. Xia, and S. Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*, 2020, pp. 128–135.
- [3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.
- [4] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.
- [5] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.
- [6] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.
- [7] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.
- [8] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," *arXiv preprint arXiv:2008.07122*, 2020.
- [9] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "Pianotree vae: Structured representation learning for polyphonic music," *arXiv preprint arXiv:2008.07118*, 2020.
- [10] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 725–734.
- [11] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using blstm networks," *arXiv preprint arXiv:1712.01011*, 2017.
- [12] Y. Zhang, Z. Wang, D. Wang, and G. Xia, "Butter: A representation learning framework for bi-directional music-sentence retrieval and generation," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, 2020, pp. 54–58.
- [13] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," *arXiv preprint arXiv:2008.07142*, 2020.
- [14] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
- [15] H.-M. Liu and Y.-H. Yang, "Lead sheet generation and arrangement by conditional generative adversarial network," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 722–727.
- [16] B. Jia, J. Lv, Y. Pu, and X. Yang, "Impromptu accompaniment of pop music using coupled latent variable model with binary regularizer," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.
- [17] P.-C. Chen, K.-S. Lin, and H. H. Chen, "Automatic accompaniment generation to evoke specific emotion," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–6.
- [18] Y.-C. Wu and H. H. Chen, "Emotion-flow guided music accompaniment generation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 574–578.
- [19] C.-H. Liu and C.-K. Ting, "Polyphonic accompaniment using genetic algorithm with music theory," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–7.
- [20] M. Bretan, G. Weinberg, and L. Heck, "A unit selection methodology for music generation using deep neural networks," *arXiv preprint arXiv:1612.03789*, 2016.
- [21] G. Xia, "Expressive collaborative music performance via machine learning," Jul 2018. [Online]. Available: https://kithub.cmu.edu/articles/thesis/Expressive_Collaborative_Music_Performance_via_Machine_Learning/6716609/1
- [22] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," *arXiv preprint arXiv:1803.06841*, 2018.
- [23] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," *arXiv preprint arXiv:2010.07518*, 2020.

- [24] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4364–4373.
- [25] G. Bernardes, D. Cocharro, M. Caetano, C. Guedes, and M. E. Davies, "A multi-level tonal interval space for modelling pitch relatedness and musical consonance," *Journal of New Music Research*, vol. 45, no. 4, pp. 281–294, 2016.
- [26] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.
- [27] E. Foxley, "Nottingham database," [EB/OL], <https://ifdo.ca/~seymour/nottingham/nottingham.html> Accessed May 25, 2021.
- [28] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Icml*, 2010.
- [29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.
- [30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.
- [31] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

Author Index

Author Index

- A, Rohit M 19
 Aguilar, Aldo 220
 Ahlbäck, Sven 151, 389
 Akama, Taketo 27
 Alfaro-Contreras, María 35
 Alunno, Marco 818
 Antony, Ria 358
 Aravind, R 547
 Audebert, Nicolas 197
- Barthet, Mathieu 610
 Baró, Arnau 690
 Batliner, Anton 509
 Baumann, Stefan A 42
 Beauvais-Lacasse, Emmanuelle 555
 Benetatos, Christodoulos 786
 Benetos, Emmanouil 389, 500
 Berg-Kirkpatrick, Taylor 159, 167
 Berndt, Axel 50
 Bhattacharjee, Amitrajit 19
 Bhattacharya, Arpita 358
 Bientinesi, Paolo 818
 Bigo, Louis 58
 Brazier, Charles 66
 Bridge, Derek 213
 Brown, Dan 562
 Bryan, Nicholas J. 594, 777
 Burgoyne, John Ashley 135, 673
- Calvo-Zaragoza, Jorge 35, 81
 Cano, Estefania 237
 Cao, Yin 342
 Carr, Cj 610
 Carter-Enyi, Aaron 74
 Castellanos, Francisco J. 81
 Castellon, Rodrigo 88
 Chang, Chin-Jui 97
 Chanquion, Pierre 443
 Chen, Bo-Yu 310
 Chen, Yen-Hsing 105
 Chen, Yi-Wei 105
 Cheung, Vincent K.M. 113
 Ching, Joann 318
 Choi, Keunwoo 121, 342, 396, 769
 Chowdhury, Shreyan 128
 Condit-Schultz, Nathaniel 74
 Cornelissen, Bas 135
 Cwitkowitz, Frank 270
- Dai, Shuqi 143
 Dannenberg, Roger 143
 Demirel, Emir 151
 Dixon, Simon 151, 205, 389
 Doh, Seungheon 318
- Donahue, Chris 88, 159
 Dong, Hao-Wen 159, 167
 Duan, Zhiyao 270, 786
- Edirisooriya, Sachinda 167
 Elowsson, Anders 174
 Engel, Jesse 246, 468
 Ens, Jeffrey 182
- Fazekas, Gyorgy 254
 Finkensiep, Christoph 189
 Flexer, Arthur 531
 Fornés, Alicia 690
 Foscarin, Francesco 197
 Foster, Dave 205
 Fournier-S'Niehotta, Raphael 197
 Fujii, Shinya 500
 Fujinaga, Ichiro 404
 Fukatsu, Haruno 500
- Gabbolini, Giovanni 213
 Gallego, Antonio-Javier 81
 Garcia, Hugo F Flores 220
 Gomes, Celso 143
 Gomez, Emilia 237
 Gotham, Mark R H 229, 404
 Goto, Masataka 697, 705
 Gouyon, Fabien 350
 Gulz, Torbjörn L 460
 Guo, Dongsheng 665
 Gómez-Cañón, Juan S. 237
- Hadjeres, Gaëtan 579
 Hamasaki, Masahiro 697, 705
 Hawthorne, Curtis 246, 468
 Hayes, Ben 254
 He, Qiqi 682
 Hennequin, Romain 714
 Hentschel, Johannes 262
 Herrera, Perfecto 237
 Heydari, Mojtaba 270
 Hiramatsu, Yuki 278
 Holzapfel, Andre 301, 460
 Hsiao, Yo-Wei 285
 Hsu, Jui-Yang 293
 Huang, Rujing 301
 Hung, Hsiao-Tzu 318
 Hung, Tun Min 310
 Höger, Frank 366
- Inesta, Jose M. 35
 Ishida, Keisuke 697
 Ishizuka, Ryoto 493
- Jacoby, Nori 366

- Ji, Kevin 326, 810
Jiang, Junyan 381
Jiang, Tao 825
Jiang, Yiliang 825
Jin, Zeyu 143
- K, Sri Rama Murty 547
Kang, Lei 690
Kao, Hsuan-Kai 113
Keskar, Asawari 657
Kim, Keunhyoung 334
Kim, Nabin 318
Klauk, Stephanie 229
Kleinertz, Rainer 229
Kondo, Haruka 500
Kong, Qiuqiang 342, 381
Korzeniowski, Filip 350
Kosta, Katerina 443
Kum, Sangeun 334
Kumar, Adarsh 610
- Lalmas, Mounia 602
Lartillot, Olivier 174
Lattner, Stefan 484
Le, Anh 358
Lee, Chun-Yi 97
Lee, Harin 366
Lee, Hung-Shin 105
Lee, Jin Ha 358
Lee, Jongpil 334
Lenchitz, Jordan 374
Lerch, Alexander 517, 634
Li, Wei 682
Liang, Percy 88
Lin, Liwei 381
Lisena, Pasquale 412
Liu, Haohe 342
Lordelo, Carlos 389
Lu, Wei-Tsung 396, 730
López, Néstor Nápoles 404
- Madaghiele, Vincenzo 412
Manilow, Ethan 220, 246
Martin, Nicolas 58
Masclef, Ninon Lizé 420
Masuda, Naotake 428
Mcadams, Stephen 555
Mcauley, Julian 159, 167
Mcbride, John M 500
Mcleod, Andrew 435
Medeot, Gabriele 443
Mehrotra, Rishabh 602
Micchi, Gianluca 443
Mignot, Remi 762
Miguel, Martin A 452, 525
Miniotaité, Jūra 460
Misgeld, Olof 460
- Mittal, Gautam 468
Moss, Fabian C. 262, 569
Moussallam, Manuel 420, 714
Murthy, Hema A 547
Mysore, Gautham 777
Müller, Meinard 229, 626, 746
- Nadeem, Faraaz 476
Nakamura, Eita 278
Nam, Juhan 318, 334
Neuwirth, Markus 262
Nieto, Oriol 594
Nistal, Javier 484
- Oramas, Sergio 350
Oyama, Takehisa 493
Ozaki, Yuto 500
- Parada-Cabaleiro, Emilia 509, 618
Pardo, Bryan 220
Park, Minsu 366
Pasquier, Philippe 182
Pati, Ashis 517
Peeters, Geoffroy 539, 754
Peng, Hu 825
Pfordresher, Peter 500
Pironio, Nicolás 525
Praher, Verena 531
Prinz, Katharina 531
Proutskova, Polina 500
Prétet, Laure 539
- R, Gowriprasad 547
Rabinovitch, Gilad 74
Rao, Preeti 19, 657
Regnier, David 58
Reymore, Lindsey 555
Richard, Gael 714
Richard, Gaël 484, 539
Rizo, David 35
Robinson, Kyle 562
Roebel, Axel 762
Rohrmeier, Martin A 189, 262, 435, 569
Rosenzweig, Sebastian 626
Roth, Camille 642
Rouard, Simon 579
Rowe, Luke O 586
- Saitis, Charalampos 254
Saito, Daisuke 428
Sakai, Emi 500
Salamon, Justin 594, 777
Santero, Nicole 358
Saravanou, Antonia 602
Sarmiento, Pedro Pereira 610
Savage, Patrick E. 500
Schedl, Markus 509, 618

Schmitt, Maximilian 509
Schuberth, Florian 746
Schuller, Bjorn W. 509
Schweiger, Harald Victor 618
Schwär, Simon J 626
Schönwiesner, Marc 366
Serra, Xavier 769, 777
Seshadri, Pavan M 634
Shakespeare, Dougal 642
Shi, Zhengshan 650
Shikarpur, Nithya Nadig 657
Simon, Ian 246, 468
Six, Joren 500
Slezak, Diego Fernandez 452, 525
Smith, Bennett 555
Smith, Jordan B. L. 730
Song, Qingwei 665
Song, Xuchen 396, 730
Spijkervet, Janne 673
Sturm, Bob L. T. 301
Su, Li 113, 285, 293
Sun, Qiwei 665
Sun, Xiaoheng 682
Swavely, Rigel 246

Tierney, Adam 500
Tomasi, Federico 602
Torras, Pau 690
Troncy, Raphael 412
Tsai, Timothy 326, 802, 810
Tsukuda, Kosetsu 697, 705
Tzanetakis, George 586

V, Venkatesh 547
Vaglio, Andrea 420, 714

Wang, Hsin-Min 105
Wang, Ju-Chiang 396, 730
Wang, Yuxuan 121, 342
Wang, Ziyu 722
Wei, Shiqi 738
Weiss, Christof 229, 746, 754
Widmer, Gerhard 66, 128, 531
Wolff, Daniel 762
Won, Minz 396, 769, 777
Wuerkaixi, Abudukelimu 786

Xia, Gus 381, 722, 738, 833

Yamamoto, Kazuhiko 794
Yang, Daniel 326, 802, 810
Yang, Yi-Hsuan 97, 237, 310, 318, 610
Yeh, Yen Tung 310
Yongwei, Gao 682
Yoshii, Kazuyoshi 278, 493

Zehren, Mickael 818
Zeitler, Johannes 746
Zhang, Changshui 786
Zhang, Huan 825
Zhao, Jingwei 833
Zheng, Haiyong 665
Zuidema, Willem 135
Zukowski, Zack 610
Zunner, Tim 746