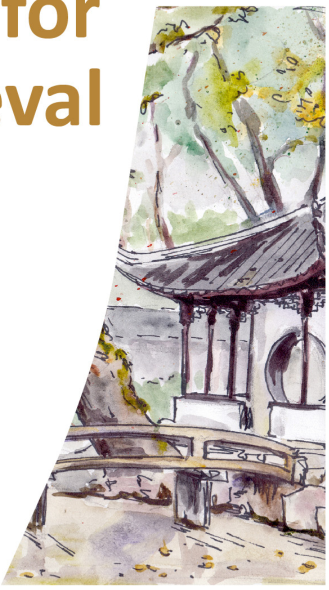
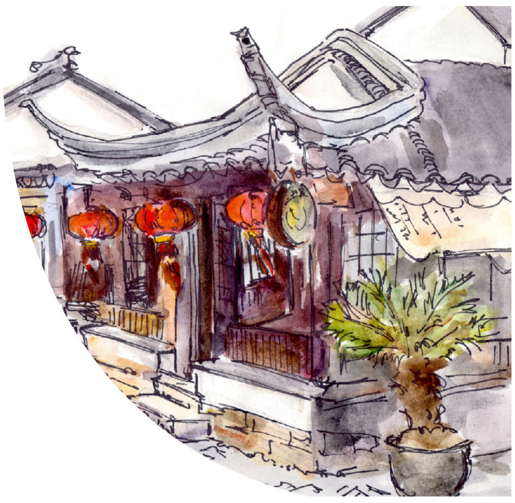
18th International Society for Music Information Retrieval Conference

October 23 - 27, 2017
Suzhou, China

ISMIR
2017, SUZHOU, CHINA





ISMIR 2017

Proceedings of the 18th International Society for Music Information Retrieval Conference

ISMIR

2017, SUZHOU, CHINA

October 23 - 27, 2017
Suzhou, China

Edited by
Xiao Hu, Sally Jo Cunningham, Doug Turnbull, and Zhiyao Duan

ISMIR 2017 is organized by National University of Singapore and the International Society for Music Information Retrieval.

Website: https://ismir2017.smcnus.org/

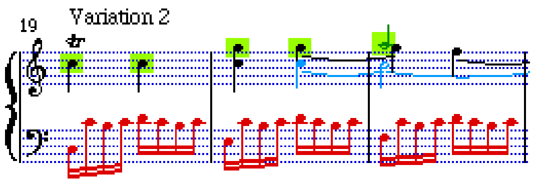
ISMIR 2017 Logo & Proceedings Cover Designed by: Zeyu He

Edited by:

Sally Jo Cunningham (University of Waikato, New Zealand)
Zhiyao Duan (University of Rochester, USA)
Xiao Hu (University of Hong Kong, Hong Kong)
Douglas Turnbull (Ithaca College, USA)

ISBN: 978-981-11-5179-8
Title: Proceedings of the 18th International Society for Music Information Retrieval Conference

Organizers



Sponsors

Platinum Sponsors

Gold Sponsors

DOLBY® pandora®

steinberg YOUSICIAN

Silver Sponsors

ACRCloud Adobe iZotope

Bronze Sponsors

Google

School of Information Sciences
The iSchool at Illinois

sonzn

Conference Organizing Committee

Program Committee

Jean-Julien Aucouturier, CNRS/IRCAM
Juan Pablo Bello, New York University
Emmanouil Benetos, Queen Mary University of London
Rachel Bittner, New York University
J. Ashley Burgoyne, University of Amsterdam
Sebastian Böck, Johannes Kepler University Linz
Ching-Hua Chuan, University of North Florida
Tom Collins, Lehigh University
Roger Dannenberg, Carnegie Mellon University
Bas de Haas, Chordify
Johanna Devaney, The Ohio State University
Christian Dittmar, International Audio Laboratories Erlangen
Simon Dixon, Queen Mary University of London
J. Stephen Downie, University of Illinois
Andreas Ehmann, Pandora
Sebastian Ewert, Queen Mary University of London
George Fazekas, Queen Mary University of London
Ben Fields, FutureLearn
Arthur Flexer, Austrian Research Institute for Artificial Intelligence (OFAI)
José Fornari, NICS / COCEN / UNICAMP
Ichiro Fujinaga, McGill University
Masataka Goto, AIST
Fabien Gouyon, Pandora
Maarten Grachten, Johannes Kepler University
Emilia Gómez, Music Technology Group, Universitat Pompeu Fabra
Andrew Hankinson, University of Oxford
Dorien Herremans, Singapore University of Technology and Design
André Holzapfel, KTH
Eric Humphrey, Spotify, Inc
Ozgur Izmirli, Connecticut College
Peter Knees, Vienna University of Technology
Paul Lamere, Spotify, Inc
Audrey Laplante, Université de Montréal
Jinha Lee, University of Washington
Cynthia Liem, Delft University of Technology
Michael Mandel, The Ohio State University
Brian McFee, New York University
Matt McVicar, The University of Bristol
Meinard Müller, International Audio Laboratories Erlangen
Oriol Nieto, Pandora
Nicola Orio, Department of Cultural Heritage

Kevin Page, University of Oxford
Hélène Papadopoulos, CNRS
Bryan Pardo, Northwestern University
Geoffroy Peeters, UMR STMS (IRCAM CNRS UPMC)
Matthew Prockup, Pandora
Marcelo Queiroz, University of São Paulo
Rafael Ramirez, Universitat Pompeu Fabra
Markus Schedl, Johannes Kepler University Linz
Erik Schmidt, Pandora
Jeffrey Scott, Gracenote, Inc.
Xavier Serra, Universitat Pompeu Fabra
Jordan Smith, AIST Japan
Mohamed Sordo, Pandora
Bob Sturm, Queen Mary University of London
Li Su, Academia Sinica
George Tzanetakis, University of Victoria
Julián Urbano, Delft University of Technology
Peter van Kranenburg, Meertens Institute, Amsterdam
Anja Volk, Utrecht University
Xinxi Wang, NUS Alumni
Kazuyoshi Yoshii, Kyoto University
Eva Zangerle, University of Innsbruck

Reviewers

Samer Abdallah
Jakob Abeßer
Stefanie Acevedo
Kat Agres
Anna Aljanaki
Nazareno Andrade
Manuel Anglada Tort
Tom Arjannikov
Andreas Arzt
Stefan Balke
Isabel Barbancho
Michael Barone
Christine Bauer
Alejandro Bellogín
Brian Bemman
Oded Ben-Tal
Gilberto Bernardes
Louis Bigo
Laura Bishop
Dmitry Bogdanov
Ciril Bohak
Juanjo Bosch
Baris Bozkurt
Maarten Brinkerink
Donald Byrd
Jorge Calvo-Zaragoza
Carlos Cancino Chacon
Estefania Cano
Julio Carabias
Rafael Caro
Mark Cartwright
Pritish Chadna
Tak-Shing Chan
Shuo Chen
Ching-Wei Chen
Tian Cheng
Keunwoo Choi
Chia-Hao Chung
Andrea Cogliati
Emanuele Coviello
Tim Crawford
Brecht De Man
David Martins de Matos
Reinier de Valk
Andrew Demetriou
Junqi Deng
Diego Di Carlo
Hamid Eghbal-Zadeh
Anders Elowsson
Slim Essid
Sebastian Ewert
Zhe-Cheng Fan
Ángel Faraldo
Xavier Favory
Flavio Figueireido
Thomas Fillon
Derry Fitzgerald
Eduardo Fonseca
Frederic Font
Dominique Fourer
Magdalena Fuentes
Satoru Fukayama
Martin Gasser
Mathieu Giraud
Aggelos Gkiokas
Nicolas Gold
Rong Gong
Ryan Groves
David Grunberg
Sankalp Gulati
Chun Guo
Swapnil Gupta
Masatoshi Hamanaka
Yun Hao
Tom Hedges
Jason Hockman
Amy Hoover
Yu-Hui Huang
Jose Inesta
Charles Inskip
Katsutoshi Itoyama
Berit Janssen
Tristan Jehan
Blair Kaneshiro
Andreas Katsiavalos
Rainer Kelz
Jaehun Kim
Jong Wook Kim
Minje Kim
Rainer Kleinertz
Vincent Koops
Filip Korzeniowski
Katerina Kosta
Amanda Krause
Nadine Kroher
Anna Kruspe
Lun-Wei Ku
Frank Kurth
Mathieu Lagrange
Robin Laney
Thibault Langlois
Stefan Lattner
Sang Won Lee
Bernhard Lehner
Florence Leve
Mark Levy
David Lewis
Bochen Li
Peter Li
Jialing Li
Dawen Liang
Thomas Lidy
Elad Liebman
Yuan-Pin Lin
Yi-Wen Liu
Antoine Liutkus
Vincent Lostanlen
Athanasios Lykartsis
Patricio López-Serrano
Akira Maezawa
José Pedro Magalhães
Leandro Balby Marinho
Matija Marolt
Alan Marsden
Matthias Mauch
James McInerney
Cory McKay
Andrew McLeod
Gabriele Medeot
Marius Miron
Saumitra Mishra
Nicola Montecchio
Josh Moore
Hema Murthy
Hidehisa Nagano
Eita Nakamura
Tomoyasu Nakano
Juhan Nam
Maria Navarro
Eric Nichols
Mitsunori Ogihara

Sergio Oramas
Ken O’Hanlon
Gabriel Parent
Tae Hong Park
Antonio Pertusa
Pedro Pestana
Martin Pichl
Aggelos Pikrakis
Phillip Popp
Alastair Porter
Elio Quinton
Colin Raffel
Zafar Rafii
Gang Ren
Iris Ren
David Rizo
Matthias Robine
Francisco Rodriguez
Christian Rohlfing
Gerard Roma
Robert Rowe
Gabriel Sargent
Flávio Schiavoni
Alexander Schindler
Jan Schlueter
Rodrigo Schramm
Prem Seetharaman
Marco Selvi
Sertan Şentürk
Daniel Shanahan
Zhengshan Shi
Diana Siwiak
Joren Six
Olga Slizovskaia
Lloyd Smith
Ajay Srinivasamurthy
Ryan Stables
Rebecca Stewart
Fabian-Robert Stoeter
Tiago Tavares
Florian Thalmann
Mi Tian
Marko Tkalcic
George Tourtellot
Kosetsu Tsukuda
Andreu Vall
Rafael Valle
Marnix van Berchum
Bastiaan van der Weij
Leigh VanHandel
Gissel Velarde
Amruta Vidwans
Gabriel Vigliensoni
Richard Vogl
Siying Wang
Hsin-Min Wang
Cheng-i Wang
Kevin Webster
David Weigl
Ron Weiss
Christof Weiß
Tillman Weyde
Daniel Wolff
Matthew Woolhouse
Chih-Wei Wu
Guangyu Xia
Karthik Yadati
Luwei Yang
Asteris Zacharakis
Frank Zalkow
Yichi Zhang

Preface

It is our great pleasure to welcome you to the 18th International Society for Music Information Retrieval Conference (ISMIR 2017). The annual ISMIR conference is the world's leading research forum on processing, analyzing, searching, organizing, and accessing music-related data. This year's conference takes place at the National University of Singapore Research Institute (NUSRI) in Suzhou, China, from October 23-27, 2017. It is organized by National University of Singapore.

Music Information Retrieval (Music-IR) is a highly interdisciplinary field, incorporating elements from the disciplines of signal processing, machine learning, psychology, musicology, electrical engineering, computer science, music librarianship, and many more. This conference aims to cover the broad range of Music-IR topics, enabling researchers, developers, students, educators, and other professionals to exchange ideas, share results, and gain new perspectives from each other. The conference in turn provides opportunities to foster collaborations and encourage new developments in the field.

The present volume contains the complete manuscripts of all peer-reviewed papers presented at ISMIR 2017. A total of 226 submissions were received before the deadline, of which 190 complete and well-formatted papers entered the review process. Special care was taken to assemble an experienced and interdisciplinary review panel comprising people from many different academic and industrial institutions worldwide. As in previous years, reviews were double-blinded (i.e., both the authors and the reviewers were anonymous) with a two-tier review model involving a pool of 275 reviewers, including a program committee (PC) of 63 members. Each paper was assigned to a PC member and three reviewers. Reviewer assignments were based on topic preferences, bidding on papers, and PC member assignments. Following the review phase, PC members and reviewers entered a discussion phase aiming to homogenize acceptance versus rejection decisions.

Handling four submissions on average, each PC member was asked to adopt an active role in the review process by conducting an intensive discussion phase with the other reviewers and providing a detailed meta-review. Final acceptance decisions were based on 758 reviews and meta-reviews. Of the 190 reviewed papers, 97 were accepted, resulting in an acceptance rate of 51.0%. The table shown on the next page summarizes the ISMIR publication statistics over the history of the conference.

The mode of presentation of the accepted papers was determined after the accept/reject decisions and has no relation to the quality of the papers or to the number of pages allotted in the proceedings. From the 97 accepted contributions, 24 papers were chosen for oral presentation to achieve a broad coverage of Music-IR topics, while the other 73 were chosen for poster presentation. Each oral presentation has a 20-minute slot (including setup and questions/answers from the audience) and each poster presentation takes place over two sessions on a given day, during lunch and in the afternoon, for a total of 3 hours.

The ISMIR 2017 conference runs for a 5-day period. Accepted papers are presented over a period of 3.5 days, preceded by a day of tutorials and followed by a half day of Late-Breaking & Demo (LBD) and Unconference sessions. The main academic program also includes the Women in Music-IR (WiMIR) annual meeting, keynote talks, panel sessions, and social events.

In addition, ISMIR 2017 provides a musical program. This program includes a diverse selection of music, from western classical standards to traditional music of the Suzhou area, and also focuses on music incorporating aspects of Music-IR. In this way we hope to encourage the use of such techniques in the creation of new music, as well as to explore music which can lead to novel research ideas in the field.

ISMIR 2017 also features four major social events, including the Welcome Reception, a Grand Canal cruise, a concert at the Master of Nets Garden, and the ISMIR 2017 Banquet with music performances from our community.

Finally, ISMIR 2017 offers three satellite events before and after the main conference: Hacking Audio and Music Research (HAMR), the China Conference on Sound and Music Technology (CSMT), and the Digital Libraries for Musicology (DLfM) workshop.

We believe this is an exciting and engaging program reflecting the breadth and depth of activities across our community.

Year	Location	Oral	Poster	Total Papers	Total Pages	Total Authors	Unique Authors	Pages/ Paper	Authors/ Paper	Unique Authors/ Paper
2000	Plymouth	19	16	35	155	68	63	4.4	1.9	1.8
2001	Indiana	25	16	41	222	100	86	5.4	2.4	2.1
2002	Paris	35	22	57	300	129	117	5.3	2.3	2.1
2003	Baltimore	26	24	50	209	132	111	4.2	2.6	2.2
2004	Barcelona	61	44	105	582	252	214	5.5	2.4	2
2005	London	57	57	114	697	316	233	6.1	2.8	2
2006	Victoria	59	36	95	397	246	198	4.2	2.6	2.1
2007	Vienna	62	65	127	486	361	267	3.8	2.8	2.1
2008	Philadelphia	24	105	105	630	296	253	6	2.8	2.4
2009	Kobe	38	85	123	729	375	292	5.9	3	2.4
2010	Utrecht	24	86	110	656	314	263	6	2.	2.4
2011	Miami	36	97	133	792	395	322	6	3	2.4
2012	Porto	36	65	101	606	324	264	6	3.2	2.6
2013	Curitiba	31	67	98	587	395	236	5.9	3	2.4
2014	Taipei	33	73	106	635	343	271	6	3.2	2.6
2015	Málaga	24	90	114	792	370	296	7	3.2	2.6
2016	New York	25	88	113	781	341	270	6.9	3.0	2.4
2017	**Suzhou**	**24**	**73**	**97**	**716**	**324**	**248**	**7.4**	**3.3**	**2.6**

Tutorials

Five tutorials take place on Monday, providing a balance between culture and technology. Two 3-hour tutorials are presented in parallel on Monday morning, and three 3-hour tutorials are presented in parallel on Monday afternoon.

Morning Sessions:

Tutorial 1: Bayes and Markov Listen to Music
George Tzanetakis, University of Victoria, Canada

Tutorial 2: Leveraging MIDI Files for Music Information Retrieval
Colin Raffel, Google Brain, USA

Afternoon Sessions:

Tutorial 3: A Basic Introduction to Audio-Related Music Information Retrieval
Meinard Müller, International Audio Laboratories Erlangen, Germany
Stefan Balke, International Audio Laboratories Erlangen, Germany
Christof Weiss, International Audio Laboratories Erlangen, Germany

Tutorial 4: So You Want to Conduct a User Study in MIR?
Andrew Demetriou, Delft University of Technology, Netherlands
Audrey Laplante, Université de Montréal, Canada
Sally Jo Cunningham, University of Waikato, New Zealand
Cynthia Liem, Delft University of Technology, Netherlands

Tutorial 5: Machine-Learning for Symbolic Music Generation
François Pachet, SONY CSL, France
Jean-Pierre Briot, Paris VI - SONY CSL, France

Keynote Speakers

We are honored to have three distinguished keynote speakers:

Structures: Performed, Perceived and Constructed

Prof. Elaine Chew

Professor of Digital Media, School of Electronic Engineering and Computer Science, Queen Mary University of London, UK

From Stanford to Smule: Reflections on the Nine-Year Journey of a Music Start-Up

Dr. Jeffrey C. Smith

Co-founder, Chairman, and CEO, Smule, Inc., USA

Does MIR Stop at Retrieval?

Prof. Roger B. Dannenberg

Professor of Computer Science, Art & Music, Carnegie Mellon University, USA

WiMIR Annual Meeting

Women in Music-IR (WiMIR) is a group of people in the Music-IR community dedicated to promoting the role of, and increasing opportunities for, women in the field. Participants meet to network, share information, and discuss in an informal setting the goal of building a community that supports women – and more broadly, diversity – in the field of Music-IR.

WiMIR has held annual meetings at the ISMIR conference since 2012, garnering a high turnout of both female and male attendees. For the first time in 2016, WiMIR has organized a mentoring program connecting female students, postdocs, and early-stage researchers to more senior females and male allies in the field, and has also received substantial financial support which enables more female researchers to attend the ISMIR conference.

This year's WiMIR session is hosted by Preeti Rao (Indian Institute of Technology Bombay, India) and Zhongzhe Xiao (Soochow University, Mainland China). Apart from a presentation on the ongoing diversity initiatives, the WiMIR session at ISMIR 2017 features a talk by Shawn Carney, Head of Global IT at Spotify on promoting diversity, with the competitive edge it provides in an increasingly interdependent and interconnected world.

Bye Bye Bias: Promoting Diverse Teams

Shawn Carney

Head of Global IT at Spotify, USA

Panels

ISMIR 2017 features two panels: the ISMIR Industry Panel, and the Future of Music-IR Research panel.

Industry Panel

The Industry Panel is moderated by Blair Kaneshiro (Stanford University, USA). The panel brings together five distinguished members of industry for a discussion of current topics as well as Q&A with the audience:

- Junqi Deng, KuGou, Mainland China
- Anssi Klapuri, Yousician, Finland
- Matthew McCallum, Gracenote, USA
- John Neuharth, Microsoft, USA
- Avery Wang, Shazam, USA

Future of Music-IR Research Panel

The Future of Music-IR Research panel focuses on novel and creative research directions (e.g., neuroscience, deep learning, Music-IR & health, Music-IR & education). It is moderated by the ISMIR 2017 General Chair, Ye Wang (National University of Singapore, Singapore). The panel features young researchers who have already made an impact on our community:

- Kat Agres, Agency for Science, Technology & Research (A*STAR), Singapore
- Cynthia Liem, Delft University of Technology, Netherlands
- Brian McFee, New York University, USA
- Gus (Guangyu) Xia, NYU Shanghai, Mainland China

Late-Breaking/Demo & Unconference

Friday afternoon is dedicated to late-breaking papers and Music-IR system demonstrations. Abstracts for these presentations are available online. Moreover, as in previous years, we have a special "unconference" session in which participants break up into smaller groups to discuss Music-IR issues of particular interest. This is an informal and informative opportunity to get to know peers and colleagues from around the world.

Music Program and Community Performance

On Wednesday, there is a special concert night in the Master of Nets Garden (网师园). Conference participants enjoy Chinese traditional music and performances in one of the finest gardens in China, and a UNESCO heritage site, along with a docent-led tour of the garden. The program includes Kun opera (昆曲), Pingtan – storytelling and ballad singing in Suzhou dialect (苏州评弹), traditional dances, and instrumental performances such as Qin (古琴), Zheng (古筝), Pipa (琵琶), Di (笛子), and Erhu (二胡).

On Thursday, there is a community performance night following the banquet dinner. The program includes both submitted and invited pieces, including original work that intersects Music-IR technology, traditional Chinese music, and music ensemble with sensors. This includes seven music pieces:

1. Etudes-Tableaux No. 1, by Dr. Jeffrey C. Smith.
2. Ancient Pipa Solo Piece “Ambush from Ten Sides”, by Shuqi Dai.
3. Fibonacci-sequence based composition: Carnatic Classical performance, by Venkata Subramanian Viraraghavan.
4. A Trip to Suzhou, by Roger B. Dannenberg, Gus Xia, and Shuqi Dai.
5. Informatics Philharmonic Concert, by Christopher Raphael.
6. Liuyang River, by Zhengshan Shi.
7. Motion Initiated Music Ensemble with Sensors (MIMES), by the NUS Sound and Music Computing Lab.

Social Events

In addition to the academic focus of ISMIR, we have aimed to provide a number of unique social events. The social program provides participants with an opportunity to relax after meetings, to experience Suzhou City, and to network with other ISMIR participants. The social program includes:

Welcome Reception

Welcome reception happens at the NUSRI cafe on Monday, October 23, from 6pm to 8pm. Hors d’oeuvres and drinks are provided.

WiMIR/Diversity Reception

Amazon Music presents a reception at the NUSRI cafe on Tuesday, October 24, from 6pm to 7:30pm to recognize WiMIR and the diversity initiatives of the ISMIR 2017 conference. All registered conference attendees are welcome. Hors d’oeuvres and drinks are provided.

Grand Canal Cruise

On Wednesday, October 25, from 7pm to 8pm, conference participants take a cruise on the Suzhou Ancient Grand Canal (苏州古运河). The Grand Canal is the oldest and longest man-made waterway in the world. Parts of this canal between Hangzhou and Beijing date back more than 2000 years. The canal starts from Beijing in the north and ends at Hangzhou in the south. The Grand Canal measures 1700 kilometers in length.

Garden Concert

After the Grand Canal Cruise, from 9pm to 10pm on Wednesday, October 25, conference participants visit the Master of Nets Garden (网师园) and enjoy Chinese traditional music and performances. Details of the concert can be found in the "Music Program and Community Performance" section above.

ISMIR Banquet + Community Performance

The Annual ISMIR Banquet takes place at Songhe Lou Culture and Arts Center Restaurant (松鹤楼科文中心饭店). There is a ISMIR community performance. The program comprises a selection of submitted and invited pieces, as detailed in the "Music Program and Community Performance" above.

Satellite Events

ISMIR 2017 has expanded its offering of satellite events to three, emphasizing two very important themes: the bridge between academia and industry and diversity in our community.

Hacking Audio and Music Research (HAMR) is an event series which applies the hackathon model to the development of new techniques for analyzing, processing, and synthesizing audio and music signals. This is in contrast to traditional hackathons and hack days, which generally emphasize prototyping commercial applications, but have proven to be an effective way for entrepreneurs and hobbyists to spend a concentrated period of time doing preliminary work on a new project.

The **China Conference on Sound and Music Technology** (CSMT) has the aim of promoting the research of Chinese traditional music in the field of sound and music technology, and the use of computational methods in Chinese traditional music research, as well as creating a space for dialog and debate among engineers, musicologists, and musicians who have interest in this music tradition.

The **Digital Libraries for Musicology (DLfM)** workshop presents a venue specifically for those working on, and with, Digital Library systems and content in the domain of music and musicology. This includes bibliographic and metadata for music, intersections with music Linked Data, and the challenges of working with the multiple representations of music across large-scale digital collections such as the Internet Archive and HathiTrust.

Host City

Suzhou has a rich history of over 2500 years. The city's old village canals, stone bridges, pagodas, and meticulously designed UNESCO World Heritage gardens draw millions of domestic and overseas tourists each year. Suzhou's culture is famous across China and beyond; Marco Polo himself once described Suzhou as the "Venice of the East".

The conference venues are all close to the National University of Singapore Research Institute (NUSRI) in Suzhou, which is the first oversea research institute of NUS. NUSRI is located at the Suzhou Industrial Park (SIP) within the Higher Education Town (HET) with many campuses from local and oversea universities which is rich with dining options. It offers convenient public transportation (e.g., subway) to many tourist attractions in the ancient Suzhou city.

Acknowledgments

We are very proud to present to you the proceedings of ISMIR 2017. The conference program was made possible thanks to the hard work of many people including the members of the organizing committee, the many reviewers and meta-reviewers from the program committee, as well as the leadership and administrative staff of National University of Singapore (NUS) and the National University of Singapore Research Institute (NUSRI) in Suzhou.

Special thanks go to this year's sponsors:

Platinum Sponsors:

- Amazon Music
- Gracenote
- KuGou
- Microsoft
- Native Instruments
- Shazam
- Smule
- Spotify

Gold Sponsors:

- Dolby
- Pandora
- Steinberg
- Yousician

Silver Sponsors:

- ACRCloud
- Adobe
- iZotope

Bronze Sponsors:

- Google
- The iSchool at Illinois
- SONZN

We also gratefully acknowledge the sponsors who contributed specifically to Women in Music Information Retrieval (WiMIR) initiatives and Student Travel awards:

- Amazon Music
- Gracenote
- iZotope
- Microsoft
- Smule
- Spotify
- Steinberg

Last but not least, the ISMIR program is possible only thanks to the excellent contributions of our community in response to our Call for Participation. The biggest acknowledgment goes to you, the authors, reviewers, researchers, and participants of this conference. We wish you a productive and memorable stay in Suzhou.

Sally Jo Cunningham, University of Waikato, New Zealand
Xiao Hu, University of Hong Kong, Hong Kong
Douglas Turnbull, Ithaca College, USA
Scientific Program Co-Chairs

Zhiyao Duan, University of Rochester, USA
Publications Chair

Ye Wang, National University of Singapore, Singapore
General Chair

Jyh-Shing Roger Jang, National Taiwan University, Taiwan
Wei Li, Fudan University, Mainland China
General Co-Chairs

Contents

Keynote Talks

Keynote Talk 1

Structures: Performed, Perceived and Constructed

Prof. Elaine Chew
Professor of Digital Media
School of Electronic Engineering and Computer Science
Queen Mary University of London

Abstract
Conventional understanding of music structure in musicology as well as music information research limits its definition to musical form and sectional structures. But structure is more than sonata form or ABA structure. Structure refers to all manners of musical coherence as generated by surface features and deeper ones, musical entities and boundaries, movements and arrivals. Thus, long-term modulation of intensity or tension creates a form of structure (coherence), as does local weighting of notes to indicate an upbeat or downbeat, or subtle changes of color (for example, vibrato and timbral) amidst a sustained note. The human mind is wired to perceive, use, and crave structure. Grappling with ways to construct coherence is central to the work of music making, and imagining new and convincing ways to formulate musical coherence lies at the heart of musical innovation. By considering music structures as emerging from musical sense making, we open up new ways to explore and understand the manifold forms of music structure.

With this broad definition of music structure in mind, I shall survey some of our recent work* in the scientific and computational modeling and analysis of music structure as performed, music structure as perceived, and music structure as applied to composition. Structures as perceived or communicated through prosody serve to shape the meaning of the musical text; when perceived or communicated structures serve as the given information rather than the end goal in an algorithm, this dual (reverse) approach leads to interesting insights into musical sense making; when perceived or communicated structures further serve as sources for crafting new compositions, they provide important seed material for generating coherence. In addition, the musical mind imputes structure on music information. Harking back to the medieval concept of music internal to the human body (musica humana), the presentation will conclude with applications of music structure extracted from arrhythmic heartbeats.

* The presentation includes joint work with Dorien Herremans, Isaac Schankler, Jordan Smith, Luwei Yang, Ashwin Krishna, Daniel Soberanes, and Matthew Ybarra.

Biography
Elaine Chew is Professor of Digital Media at Queen Mary University of London’s School of Electronic Engineering and Computer Science, where she is affiliated with the Centre for Digital Music. Her research centers on mathematical modeling of musical prosody, structure, cognition, and interaction. She was previously Associate Professor at the University of Southern California’s Viterbi School of Engineering and Thornton School of Music, where she founded the Music Computation and Cognition research laboratory. Her work has received recognition through the NSF CAREER/PECASE awards, and fellowships at the Radcliffe Institute for Advanced Study at Harvard. She earned Ph.D. and S.M. degrees in Operations Research from MIT, and a B.A.S. in Mathematical and Computational Sciences (honors) and in Music Performance (distinction) from Stanford. She holds Fellowship and

Licentiate diplomas in piano performance from Trinity College London. As a pianist, she has performed internationally as soloist and chamber musician, and she frequently collaborates with composers to commission, create, present, and record new music. Her work has been featured on Los Angeles Philharmonic’s Inside the Music series, and in an exhibit on Beautiful Science at the Huntington Library in California. She has served as a member of the MIT Music and Theater Arts Visiting Committee and the Georgia Institute of Technology School of Music External Review Committee. She is on the advisory/editorial boards of the Computer Music Journal, the Journal of Music and Mathematics, Music Theory Spectrum, and ACM Computers in Entertainment. This year, she is also a jury member of the Guthman Musical Instrument Competition and the Falling Walls Lab.

Keynote Talk 2

From Stanford to Smule: Reflections on the Nine-Year Journey of a Music Start-Up

Dr. Jeffrey C. Smith
Co-founder, Chairman, and CEO
Smule, Inc.

Abstract

Smule, a music start-up based in San Francisco, began as a conversation between Jeffrey Smith, a PhD student at Stanford's CCRMA, and Ge Wang, a newly hired professor. Nine years later, Smule has emerged as the leading platform for music discovery and collaboration. Smule's global community of 50 million people create 7 billion recordings each year, and upload over 36 terabytes of their music to the Smule network each day. The Smule catalog of 2 million songs, which doubled in size in the past six months, represents one of the largest corpuses of structured musical content on the Internet. This catalog includes musical backing tracks in MIDI and MP4 formats, lyrics, pitch data, timing, and musical structure. Smule generated $101M in sales in '16 and has 1.8 million paying subscribers to their service.

Tracing the nine-year arch of Smule from a student concept to a market leader, what can we learn about the potential symbiotic relationship between academic research and commercial innovation? What was the genesis of Smule? What was the role of students, research, Stanford, venture capital, and more broadly, Silicon Valley? What were the formative challenges the company confronted as it scaled from tens of thousands of users "blowing" air through their iPhone microphones with Smule's Ocarina (a flute-like instrument designed for the original iPhone) in '08 to today, where an active community sings and plays 20M songs each day on their mobile phones, often together in collaboration? How is a musical social graph different than a social graph built around other forms of media, such as photos, video, or text? Finally, what role did MIR play in the development of the Smule business model and technology stack?

Biography

Dr. Jeffrey C. Smith**,** PhD, is the co-founder, Chairman, and CEO of Smule. Jeff has a BS in Computer Science from Stanford University and a PhD in computer-based music theory and acoustics ("Correlation Analyses of Encoded Music Performance") from Stanford's CCRMA. Jeff has taught introductory computer science courses at Stanford in addition to serving as a teaching assistant in music theory and computer science. More recently, Jeff periodically teaches Music 264 at Stanford, a seminar with lab that analyzes large-scale industry data sources (including Stanford DAMP) to develop insights into musical engagement. Early in his career, Jeff worked as a software engineer at Hewlett-Packard's language lab and IBM's Scientific Research Center in Palo Alto. For the past twenty-five years, Jeff has served as a leader in businesses he co-founded, including Envoy (acquired by Novell '95), Tumbleweed (NASDAQ listing in '99), Simplify Media (acquired by Google in '10), and for the past nine years, Smule. Jeff is the co-author of twenty-seven patents in the fields of computer music and email security. Jeff enjoys writing and playing classical piano music – he is currently immersed in Brahms Op. 9 & Op. 10.

Keynote Talk 3

Does MIR Stop at Retrieval?

Prof. Roger B. Dannenberg
Professor of Computer Science, Art & Music
Carnegie Mellon University

Abstract

The Music Information Retrieval community that formed around this conference has moved swiftly from a narrow set of concerns and problems to a much wider exploration that I have long characterized as "Music Understanding." Music Understanding explores methods to find pattern and structure in music, ranging from low-level features, such as pitch and onsets, to high-level properties such as keys, transcriptions, and yes, even genre. I believe that "Music Understanding" and the MIR community must broaden their scope even further, turning to music composition, improvisation, and production, for at least three reasons. First, attempts to automate music generation have had very limited success, so music generation is a good measure of the gap between music formalisms and our human understanding of music. Second, music generation research might lead to a better understanding of creativity, learning, and the brain. Finally, music generation has practical applications, and I will discuss one: music generation for music therapy. I will also summarize the history of computer-assisted composition, describe the state-of-the-art, and provide a critique intended to spur new research.

Biography

Dr. Roger B. Dannenberg, PhD, is currently a Professor of Computer Science at Carnegie Mellon University with courtesy appointments in the Schools of Art and Music. Dr. Dannenberg studied at Rice University and Case-Western Reserve University before receiving a Ph.D. in Computer Science from Carnegie Mellon University. He also worked for Steve Jobs at NeXT as a member of the music group, MakeMusic's commercialization of Dannenberg's computer accompaniment research, and with Music Prodigy, an award-winning MIR-based music education start-up.

Dr. Dannenberg is an international leader in computer music and is well known especially for programming language design and real-time interactive systems including computer accompaniment. He and his students have introduced a number of innovations to the field: functional programming for sound synthesis and real-time interactive systems, spectral interpolation synthesis, score-following using dynamic programming, probabilistic formulations of score alignment, machine learning for style classification, score alignment using chromagrams, and bootstrap learning for onset detection. Dannenberg is also the co-creator of Audacity, an open-source audio editor used by millions.

Dr. Dannenberg is an active trumpet player and composer. His trumpet playing includes orchestral, jazz, and experimental music with electronics, and his opera, La Mare dels Peixos, co-composed with Jorge Sastre, premiered in Valencia, Spain in December 2016.

WiMIR Talk

WiMIR Talk

Bye Bye Bias: Promoting Diverse Teams

Shawn Carney
Head of Global IT
Spotify

Abstract
Globalization is leading to an increasingly interdependent, interconnected world and diversity is both a means of differentiation within that complex system as well as a competitive advantage. Bye Bye Bias: Promoting Diverse Teams examines the role of intersectional diversity and inclusion in the process of recruiting and retaining successful teams. Starting with explaining foundational concepts, such as a shared lexicon and the benefits of diversity, and then progressing to operational tips for busting bias, building diverse recruiting processes, and retaining inclusive teams, this talk will provide tactical tools towards building and maintaining high-performing teams.

Biography
Shawn Carney has been a leader in the music and technology field since 1993. Originally based in Washington DC, she went on to join technology teams at major music labels in LA and NY. Shawn has a passion for building diverse, inclusive teams that solve problems in creative ways. When she isn't working, she spends time bicycling, playing video games, illustrating, and documenting the antics of her two dogs (a Corgi and a Chihuahua) and her two cats (a Siamese and a Scottish Fold). Shawn now leads the IT organization for Spotify and lives in New York City.

Tutorials

Tutorial 1

Bayes and Markov Listen to Music

George Tzanetakis

Abstract
Music is a very complex signal with information spread across different hierarchical levels and temporal scales. In the last 15 years in the field of Music Information Retrieval (MIR) and Music Signal Processing there has been solid progress in developing algorithms for understanding music signals with applications such as music recommendation, classification, transcription and visualization. Probabilities and probabilistic modeling play an important role in many of these algorithms. The goal of this tutorial is to explore how probabilistic reasoning is used in the analysis of music signals.

The target audience is researchers and students interested in MIR but the tutorial would also be of interest to participants from other areas of signal processing as the techniques described have a wide variety of applications. More specifically the tutorial will cover how basic discrete probabilities can be used for symbolic music generation and analysis, followed by how classification can be cast as a probability density function estimation problem through Bayes theorem. Automatic chord detection and structure segmentation will be used as a motivating problems for probabilistic reasoning over time and Hidden Markov Models more specifically. Kalman and particle filtering will be described through real-time beat tracking and score following. More complex models such as Bayesian Networks and Conditional Random Fields and how the can be applied for music analysis will also be presented. Finally the tutorial will end with Markov Logic Networks a formalism that subsumes all previous models. Through the tutorial the central concepts of Bays Theorem, Markov assumptions and maximum likelihood estimation and expectation maximization will be described.

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 is Canada Research Chair (Tier II) in the Computer Analysis and Audio and Music and received the Craigdaroch research award in artistic expression at the University of Victoria in 2012. In 2011 he was Visiting Faculty at Google Research. He received his PhD in Computer Science at Princeton University in 2002 and was a Post-Doctoral 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 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.

Tutorial 2

Leveraging MIDI Files for Music Information Retrieval

Colin Raffel

Abstract
MIDI files are a widely-available digital score format which contain a bounty of valuable information about a given piece of music. A MIDI file which has been matched to a corresponding audio recording can provide a transcription, key and meter annotations, and occasionally lyrics for the recording. They are also useful in very large-scale metadata-agnostic corpus studies of popular music. Despite their potential utility, they remain underused in the music information retrieval community. The purpose of this tutorial is to expose attendees to the promise of leveraging MIDI files in MIR tasks.

The motivation for having this tutorial now is the release of the Lakh MIDI Dataset (LMD), a collection of 178,561 MIDI files of which many have been matched and aligned to corresponding entries in the Million Song Dataset. The tutorial will therefore include a mix of explanatory sessions on research involving MIDI files and hands-on demonstrations of utilizing the LMD. Attendees will leave the tutorial with a strong awareness of what MIDI files are, what sort of information we can extract from them, what steps are necessary for leveraging this information, practical knowledge of how to utilize MIDI files, and an idea of tantalizing prospects for future research.

Colin Raffel is a researcher focused on machine learning methods for sequences, with a particular interest in music data. He is currently a Research Scientist at Google Brain. In 2016, he completed a PhD in Electrical Engineering at Columbia University in LabROSA, supervised by Dan Ellis. His thesis focused on learning- based methods for comparing sequences, with the particular application of matching MIDI files to corresponding audio recordings. Prior to his PhD, he completed a Master’s at the Center for Computer Research in Music and Acoustics and a Bachelor’s at Oberlin College.

Tutorial 3

A Basic Introduction to Audio-Related Music Information Retrieval

Meinard Müller, Stefan Balke, and Christof Weiß

Abstract
The main goal of this tutorial is to give an introduction to Music Information Retrieval with a particular focus on audio-related analysis and retrieval tasks. Well-established topics in MIR are selected to serve as motivating application scenarios. Within these scenarios, fundamental techniques and algorithms that are applicable to a wide range of analysis and retrieval problems are presented in depth. Including numerous figures and sound examples, this tutorial is intended to suite for a wide and interdisciplinary audience with no particular background in MIR or audio processing. This tutorial consists of eight parts, each lasting between 20 and 25 minutes. The first two parts cover fundamental material on music representations and the Fourier transform—concepts that are required throughout the tutorial. In the subsequent parts, concrete MIR tasks serve as starting points for our investigations. Each part starts with a general description of the MIR scenario at hand and integrates the topic into a wider context. Motivated by a concrete scenario, each part discusses important techniques and algorithms that are generally applicable to a wide range of analysis, classification, and retrieval problems.

Meinard Müller studied mathematics (Diploma) and computer science (Ph.D.) at the University of Bonn, Germany. In 2002/2003, he conducted postdoctoral research in combinatorics at the Mathematical Department of Keio University, Japan. In 2007, he finished his Habilitation at Bonn University in the field of multimedia retrieval. From 2007 to 2012, he was a member of the Saarland University and the Max-Planck Institut fur Informatik. Since September 2012, Meinard Müller holds a professorship for Semantic Audio Processing at the International Audio Laboratories Erlangen, which is a joint institution of the Friedrich-Alexander-Universitat Erlangen-Nürnberg (FAU) and the Fraunhofer-Institut fur Integrierte SchaÏtungen IIS. His recent research interests include music processing, music information retrieval, audio signal processing, and motion processing. Meinard Muller has been a member of the IEEE Audio and Acoustic Signal Processing Technical Committee from 2010 to 2015 and is a member of the Board of Directors of the International Society for Music Information Retrieval (ISMIR) since 2009. He has co-authored more than 100 peer-reviewed scientific papers, wrote a monograph titled Information Retrieval for Music and Motion (Springer, 2007) as well as a textbook titled Fundamentals of Music Processing (Springer, 2015, www.music-processing.de).

Stefan Balke studied electrical engineering (Diplom) at the Leibniz Universitat Hannover, Germany. Since 2014, he has been working towards his Ph.D. degree in the Semantic Audio Processing Group headed by Prof. Meinard Muller at the International Audio Laboratories Erlangen. His research interests include music information retrieval, deep learning, and web-based multimedia retrieval. In summer 2016, he supported Martin Pfleiderer and Klaus Frieler as a teacher at the International Summer School on Computational Musicology in Weimar, Germany. The goal of this summer school was to provide a comprehensive introduction into methods, applications, and potentials of computational musicology and music information retrieval especially for jazz music. Together with the Nuremberg University of Music, he developed a web-based interface for jazz-piano education. He is an active contributor in several open-source projects (e. g. librosa), mainly dealing with audio signal processing with Python. In his spare time, he plays trumpet in several local jazz bands and projects.

Christof Weiß studied physics (Diplom) at the Julius-Maximilians-Universitat Wurzburg as well as composition (Diplom-Musik) at the Hochschule fur Musik Wurzburg, Germany. From 2012–2015, he worked as a Ph.D. student in the Semantic Music Technologies Group at the Fraunhofer Institute fur Digitale Medientechnologie (IDMT) Ilmenau, Germany. His Ph.D. thesis deals with computational methods for tonality and style analysis in music recordings and was supervised by Prof. Karlheinz Brandenburg. In 2014, he visited the Centre for Digital Music at the Queen Mary University of London for two extended research stays. Since 2015, Christof Weiß has been a member of the Semantic Audio Processing Group headed by Prof. Meinard Muller at the International Audio Laboratories Erlangen. He conducts research in a project on Wagner’s “Ring” cycle, which is a collaboration with the musicology department of the Universitat des Saarlandes, Saarbrucken. His work as a composer encompasses pieces for orchestra, ensemble, and choir, as well as chamber music. In 2013, he was awarded a second prize in the competition “Pablo Casals” in Prades, France. He was commissioned by the Mozartfest Wurzburg and the festival “Young Euro Classics” Berlin. In 2012, he received the Youth Cultural Advancement Award of the city of Amberg, Germany. From 2007–2015, Christof Weiß was a fellow of the Foundation of German Business in the study and the Ph.D. scholarship program.

Tutorial 4

So You Want to Conduct a User Study in MIR?

Andrew Demetriou, Audrey Laplante, Sally Jo Cunningham, and Cynthia Liem

Abstract
This tutorial will consist of three main parts. In the first one, we will provide an overview of the user studies in the ISMIR community as well as in other domains such as psychology, music sociology, musicology, library and information science, and HCI, highlighting the important scholars and summarizing the major themes. In the second part, we will present an overview of the different user research methods. We will cover the commonly used methods in social science research (e.g., interviews, surveys, focus groups, written/audio journals, task-based experiments, tracking biological data, etc.), discuss the suitability of each method for different research projects, and the strengths and weaknesses of each method.

The third part will consist of an interactive session during which the participants will be invited to brainstorm research questions that are relevant to their own MIR research projects and could be answered by conducting an interdisciplinary user study. Individual participants or group of participants will present their ideas and they will receive feedback from the presenters as well as from other participants.

Andrew Demetriou is a research assistant in the Multimedia Computing research group at TU Delft. He completed a research masters in social psychology at VU Amsterdam (2015), with a focus on biological data collection methods, and mate choice/romantic attraction. His research has been published in Letters on Evolutionary Behavioral Science, Journal of Crime and Delinquency, Proceedings of ISMIR 2016, and Proceedings of 10th ACM Conference on Recommender Systems. His research interests include social and romantic bonding, optimal mental/physiological states (e.g. “flow”, mindfulness), and how music, along with biological and sensor data, can be used to study these phenomena.

Audrey Laplante is an associate professor at the Université de Montréal’s Library and Information Science School. She is a member of the Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT). She received a PhD (2008) and a Master’s (2001) in information science from McGill University and Université de Montréal, respectively. Her research interests focus primarily on the information-seeking behaviour of music researchers and amateurs, and on systems for music information retrieval and discovery. Her research has been published in a variety of outlets, including Library & Information Science Research, the Proceedings of the ISMIR, Journal of Documentation, and the collective book New Directions in Children’s and Adolescents’ Information Behavior Research (Emerald Group Publishing, 2014).

Sally Jo Cunningham is an associate professor of Computer Science at the University of Waikato (Te Whare Wānanga o Waikato), in Hamilton, New Zealand. Her research focuses on everyday, authentic information behavior over a range of media (text, music, images, and video). Sally Jo was advised by her flute instructor to choose a major other than music as an undergraduate; now she enjoys experiencing music through the experiences of other people with her MIR research. She is active in the digital libraries and human-computer interaction research communities—a member of the steering committee for JCDL; program co-chair for ICADL 2008, JCDL 2014, DL 2015, ISMIR 2017; general chair for ICADL 2017; chair of the IEEE/CS TCDL (2016-2017)—and has over 120 refereed research publications in digital libraries, music information retrieval, human-computer interaction, and machine learning.

Cynthia Liem is an Assistant Professor in Computer Science in the Multimedia Computing Group of Delft University of Technology, and pianist of the Magma Duo. She initiated and co-coordinated the European research project PHENICX (2013-2016), focusing on technological enrichment of symphonic concert recordings with partners such as the Royal Concertgebouw Orchestra. Her research interests consider music and multimedia search and recommendation, and increasingly shift towards making people discover new interests and content which would not trivially be retrieved. Beyond her academic activities, Cynthia gained industrial experience at Bell Labs Netherlands, Philips Research and Google. She was a recipient of the Lucent Global Science and Google Anita Borg Europe Memorial scholarships, the Google European Doctoral Fellowship 2010 in Multimedia, and a finalist of the New Scientist Science Talent Award 2016 for young scientists committed to public outreach. She will serve as general co-chair of ISMIR 2018, to be hosted in Delft, The Netherlands.

Tutorial 5

Machine-Learning for Symbolic Music Generation

François Pachet and Jean-Pierre Briot

Abstract

The goal of this tutorial is to present in a comprehensive way the challenges and techniques of using computers for generating musical content. Various kind of techniques will be considered from Markov models to deep learning models, with the goal of presenting both the state of the art and the current limitations and open problems. The tutorial will cover essentially symbolic music generation, with an emphasis on leadsheets, seen as the primary form of mainstream music, as well as polyvocal music.

We will cover two classes of models: Markov models, and in particular Markov Constraints models, which have been particularly successful at modeling monophonic material as well as leadsheets. We'll describe the underlying models that can be learned efficiently, and will illustrate them with many examples of generated music in various styles.

We will also cover deep learning models and application to polyvocal music. After reviewing the basic components of deep architectures (neural layers, autoencoders, recurrent networks…), we will describe how they can be used in a direct way, e.g., to produce musical accompaniment, or in a more indirect way (by controlling sampling or unit selection and aggregation, etc.) for a finer control of generation. Various examples of architectures, experiments and approaches will be analyzed and compared.

François Pachet was trained in engineering, computer science and artificial intelligence (University of Paris 6) and is also a jazz musician. His research addresses issues in music interaction and production. In 2012 He obtained an ERC Advanced Grant to develop a new generation of music and text generation tools called Flow Machines. He was also trained in classical guitar, Baroque harmony and jazz. He plays jazz guitar and composes jazz and pop music, and published two albums. He initiated the series "Why X is interesting" at ISMIR, which was instantiated for many music genres.

Jean-Pierre Briot is a computer scientist trained in mathematics and computer science at Université Pierre et Marie Curie (Paris 6). He has a long interest in music and computer science, since his PhD conducted at IRCAM and Paris 6 in 1984. He is a CNRS Research Director at Laboratoire d'Informatique de Paris 6 (LIP6), research consultant for the Flow Machines Project at Sony CSL and permanent visiting researcher at PUC-Rio University in Brazil. He is also a regular musician (jazz and Brazilian music).

Oral Session 1

Musicology

MULTI-LABEL MUSIC GENRE CLASSIFICATION FROM AUDIO, TEXT, AND IMAGES USING DEEP FEATURES

Sergio Oramas[1]**, Oriol Nieto**[2]**, Francesco Barbieri**[3]**, Xavier Serra**[1]
[1]Music Technology Group, Universitat Pompeu Fabra
[2]Pandora Media Inc.
[3]TALN Group, Universitat Pompeu Fabra
{sergio.oramas, francesco.barbieri, xavier.serra}@upf.edu, onieto@pandora.com

ABSTRACT

Music genres allow to categorize musical items that share common characteristics. Although these categories are not mutually exclusive, most related research is traditionally focused on classifying tracks into a single class. Furthermore, these categories (e.g., Pop, Rock) tend to be too broad for certain applications. In this work we aim to expand this task by categorizing musical items into multiple and fine-grained labels, using three different data modalities: audio, text, and images. To this end we present *MuMu*, a new dataset of more than 31k albums classified into 250 genre classes. For every album we have collected the cover image, text reviews, and audio tracks. Additionally, we propose an approach for multi-label genre classification based on the combination of feature embeddings learned with state-of-the-art deep learning methodologies. Experiments show major differences between modalities, which not only introduce new baselines for multi-label genre classification, but also suggest that combining them yields improved results.

1. INTRODUCTION

Music genres are useful labels to classify musical items into broader categories that share similar musical, regional, or temporal characteristics. Dealing with large collections of music poses numerous challenges when retrieving and classifying information [3]. Music streaming services tend to offer catalogs of tens of millions of tracks, for which tasks such as music classification are of utmost importance. Music genre classification is a widely studied problem in the Music Information Research (MIR) community [40]. However, almost all related work is concentrated in multi-class classification of music items into broad genres (e.g., Pop, Rock), assigning a single label per item. This is problematic since there may be hundreds of more specific music genres [33], and these may not be necessarily mutually exclusive (i.e., a song could be Pop, and at the same time have elements from Deep House and a Reggae grove). In this work we aim to advance the field of music classification by framing it as multi-label genre classification of fine-grained genres.

To this end, we present *MuMu*, a new large-scale multimodal dataset for multi-label music genre classification. *MuMu* contains information of roughly 31k albums classified into one or more 250 genre classes. For every album we analyze the cover image, text reviews, and audio tracks, with a total number of approximately 147k audio tracks and 447k album reviews. Furthermore, we exploit this dataset with a novel deep learning approach to learn multiple genre labels for every album using different data modalities (i.e., audio, text, and image). In addition, we combine these modalities to study how the different combinations behave.

Results show how feature learning using deep neural networks substantially surpasses traditional approaches based on handcrafted features, reducing the gap between text-based and audio-based classification [29]. Moreover, an extensive comparative of different deep learning architectures for audio classification is provided, including the usage of a dimensionality reduction approach that yields improved results. Finally, we show how the late fusion of feature vectors learned from different modalities achieves better scores than each of them individually.

2. RELATED WORK

Most published music genre classification approaches rely on audio sources [2, 40]. Traditional techniques typically use handcrafted audio features, such as Mel Frequency Cepstral Coecients (MFCCs) [20], as input of a machine learning classifier (e.g., SVM) [39, 44]. More recent deep learning approaches take advantage of visual representations of the audio signal in form of spectrograms. These visual representations are used as input to Convolutional Neural Networks (CNNs) [5, 6, 8, 9, 34], following approaches similar to those used for image classification.

Text-based approaches have also been explored for this task. For instance, in [13, 29] album customer reviews are used as input for the classification, whereas in [4, 22] song lyrics are employed. By contrast, there are a limited number of papers dealing with image-based genre classification [18]. Most multimodal approaches for this task found in the literature combine audio and song lyrics as text [16, 27]. Moreover, the combination of audio and video has also been explored [37]. However, the authors are not aware of published multimodal approaches for music genre classification that involve deep learning.

Multi-label classification is a widely studied problem [14, 43]. Despite the scarcity in terms of approaches for multi-label classification of music genres [36, 46], there is a long tradition in MIR for tag classification, which is a multi-label problem [5, 46].

3. MULTIMODAL DATASET

To the best of our knowledge, there are no publicly available large-scale datasets that encompass audio, images, text, and multi-label annotations. Therefore, we present *MuMu*, a new Multimodal Music dataset with multi-label genre annotations that combines information from the Amazon Reviews dataset [23] and the Million Song Dataset (MSD) [1]. The former contains millions of album customer reviews and album metadata gathered from Amazon.com. The latter is a collection of metadata and precomputed audio features for a million songs.

To map the information from both datasets we use MusicBrainz [1]. For every album in the Amazon dataset, we query MusicBrainz with the album title and artist name to find the best possible match. Matching is performed using the same methodology described in [30], following a pair-wise entity resolution approach based on string similarity. Following this approach, we were able to map 60% of the Amazon dataset. For all the matched albums, we obtain the MusicBrainz recording ids of their songs. With these, we use an available mapping from MSD to MusicBrainz [2] to obtain the subset of recordings present in the MSD. From the mapped recordings, we only keep those associated with a unique album. This process yields the final set of 147,295 songs, which belong to 31,471 albums.

The song features provided by the MSD are not generally suitable for deep learning [45], so we instead use in our experiments audio previews between 15 and 30 seconds retrieved from `7digital.com`. For the mapped set of albums, there are 447,583 customer reviews in the Amazon Dataset. In addition, the Amazon Dataset provides further information about each album, such as genre annotations, average rating, selling rank, similar products, cover image url, etc. We employ the provided image url to gather the cover art of all selected albums. The mapping between the three datasets (Amazon, MusicBrainz, and MSD), genre annotations, data splits, text reviews, and links to images are released as the *MuMu* dataset [3]. Images and audio files can not be released due to copyright issues.

[1] https://musicbrainz.org/
[2] http://labs.acousticbrainz.org/million-song-dataset-echonest-archive
[3] https://www.upf.edu/web/mtg/mumu

3.1 Genre Labels

Amazon has its own hierarchical taxonomy of music genres, which is up to four levels in depth. In the first level there are 27 genres, and almost 500 genres overall. In our dataset, we keep the 250 genres that satisfy the condition of having been annotated in at least 12 albums. Every album in Amazon is annotated with one or more genres from different levels of the taxonomy. The Amazon Dataset contains complete information about the specific branch from the taxonomy used to classify each album. For instance, an album annotated as Traditional Pop comes with the complete branch information *Pop / Oldies / Traditional Pop*. To exploit either the taxonomic and the co-occurrence information, we provide every item with the labels of all their branches. For example, an album classified as *Jazz / Vocal Jazz* and *Pop / Vocal Pop* is annotated in *MuMu* with the four labels: Jazz, Vocal Jazz, Pop, and Vocal Pop. There are in average 5.97 labels for each song (3.13 standard deviation).

Table 1. Top-10 most and least represented genres

Genre	% of albums	Genre	% of albums
Pop	84.38	Tributes	0.10
Rock	55.29	Harmonica Blues	0.10
Alternative Rock	27.69	Concertos	0.10
World Music	19.31	Bass	0.06
Jazz	14.73	European Jazz	0.06
Dance & Electronic	12.23	Piano Blues	0.06
Metal	11.50	Norway	0.06
Indie & Lo-Fi	10.45	Slide Guitar	0.06
R&B	10.10	East Coast Blues	0.06
Folk	9.69	Girl Groups	0.06

The labels in the dataset are highly unbalanced, following a distribution which might align well with those found in real world scenarios. In Table 1 we see the top 10 most and least represented genres and the percentage of albums annotated with each label. The unbalanced character of the genre annotations poses an interesting challenge for music classification that we also aim to exploit. Among the multiple possibilities that this dataset may offer to the MIR community, we focus our work on the multi-label classification problem, described next.

4. MULTI-LABEL CLASSIFICATION

In multi-label classification, multiple target labels may be assigned to each classifiable instance. More formally: given a set of n labels $L = \{l_1, l_2, \ldots, l_n\}$, and a set of m items $I = \{i_1, i_2, \ldots, i_m\}$, we aim to model a function f able to associate a set of c labels to every item in I, where $c \in [1, n]$ varies for every item.

Deep learning approaches are well-suited for this problem, as these architectures allow to have multiple outputs in their final layer. The usual architecture for large multi-label classification using deep learning ends with a logistic regression layer with sigmoid activations evaluated with the cross-entropy loss, where target labels are encoded as high-dimensional sparse binary vectors [42]. This method, which we refer as LOGISTIC, implies the assumption that the classes are statistically independent (which is not the case in music genres).

A more recent approach [7], relies on matrix factorization to reduce the dimensionality of the target labels. This method makes use of the interrelation between labels, embedding the high-dimensional sparse labels onto lower-dimensional vectors. In this case, the target of the network is a dense lower-dimensional vector which can be learned using the cosine proximity loss, as these vectors tend to be $l2$-normalized. We denote this technique as COSINE, and we provide a more formal definition next.

4.1 Labels Factorization

Let M be the binary matrix of items I and labels L where $m_{ij} = 1$ if i_i is annotated with label l_j and $m_{ij} = 0$ otherwise. Using M, we calculate the matrix X of Positive Pointwise Mutual Information (PPMI) for the set of labels L. Given L_i as the set of items annotated with label l_i, the PPMI between two labels is defined as:

$$X(l_i, l_j) = max\left(0, \log \frac{P(L_i, L_j)}{P(L_i)P(L_j)}\right) \qquad (1)$$

where $P(L_i, L_j) = |L_i \cap L_j|/|I|$ and $P(L_i) = |L_i|/|I|$.

The PPMI matrix X is then factorized using Singular Value Decomposition (SVD) such that $X \approx U\Sigma V$, where U and V are unitary matrices, and Σ is a diagonal matrix of singular values. Let Σ_d be the diagonal matrix formed from the top d singular values, and let U_d be the matrix produced by selecting the corresponding columns from U, the matrix $C_d = U_d \cdot \sqrt{\Sigma_d}$ contains the label factors of d dimensions. Finally, we obtain the matrix of item factors F_d as $F_d = C_d \cdot M^T$. Further information on this technique may be found in [17].

Factors present in matrices C_d and F_d are embedded in the same space. Thus, a distance metric such as cosine distance can be used to obtain distance measures between items and labels. Similar labels are grouped in the space, and at the same time, items with similar sets of labels are near each other. These properties can be exploited in the label prediction problem.

4.2 Evaluation Metrics

The evaluation of multi-label classification is not necessarily straightforward. Evaluation measures vary according to the output of the system. In this work we are interested in measures that deal with probabilistic outputs, instead of binary. The Receiver Operating Characteristic (ROC) curve is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. Thus, the area under the ROC curve (AUC) is often taken as an evaluation measure to compare such systems. We selected this metric to compare the performance of the different approaches as it has been widely used for genre and tag classification problems [5, 9].

The output of a multi-label classifier is a label-item matrix. Thus, it can be evaluated either from the labels or the items perspective. We can measure how accurate the classification is for every label, or how well the labels are ranked for every item. In this work, the former point of view is evaluated with the AUC measure, which is computed for every label and then averaged. We are interested in classification models that strengthen the diversity of label assignments. As the taxonomy is composed of broad genres which are over-represented in the dataset (see Table 1), and more specific subgenres (e.g., Vocal Jazz, Britpop), we want to measure whether the classifier is focusing only on over-represented genres, or on more fine-grained ones. To this end, catalog coverage (also known as aggregated diversity) is an evaluation measure used in the extreme multi-label classification [14] and the recommender systems [32] communities. Coverage@k measures the percentage of normalized unique labels present in the top k predictions made by an algorithm across all test items. Values of $k = 1, 3, 5$ are typically employed in multi-label classification.

5. ALBUM GENRE CLASSIFICATION

In this section we exploit the multimodal nature of the *MuMu* dataset to address the multi-label classification task. More specifically, and since each modality on this set (i.e., cover image, text reviews, and audio tracks) is associated with a music album, our task focuses on album classification.

5.1 Audio-based Approach

A music album is composed by a series of audio tracks, each of which may be associated with different genres. In order to learn the album genre from a set of audio tracks we split the problem into three steps: (1) track feature vectors are learned while trying to predict the genre labels of the album from every track in a deep neural network. (2) Track vectors of each album are averaged to obtain album feature vectors. (3) Album genres are predicted from the album feature vectors in a shallow network where the input layer is directly connected to the output layer.

It is common in MIR to make use of CNNs to learn higher-level features from spectrograms. These representations are typically contained in $\mathbb{R}^{\mathcal{F} \times N}$ matrices with $\mathcal{F}$ frequency bins and N time frames. In this work we compute 96 frequency bin, log-compressed constant-Q transforms (CQT) [38] for all the tracks in our dataset using `librosa` [24] with the following parameters: audio sampling rate at 22050 Hz, hop length of 1024 samples, Hann analysis window, and 12 bins per octave. In addition, log-amplitude scaling is applied to the CQT spectrograms. Following a similar approach to [45], we address the variability of the length N across songs by sampling one 15-seconds long *patch* from each track, resulting in the fixed-size input to the CNN.

To learn the genre labels we design a CNN with four convolutional layers and experiment with different number of filters, filter sizes, and output configurations (see Section 6.1).

5.2 Text-based Approach

In the presented dataset, each album has a variable number of customer reviews. We use an approach similar to [13, 29] for genre classification from text, where all reviews from the same album are aggregated into a single text. The aggregated result is truncated at 1000 characters, thus balancing the amount of text per album, as more popular artists tend to have a higher number of reviews. Then we apply a Vector Space Model approach (VSM) with tf-idf weighting [47] to create a feature vector for each album. Although word embeddings [25] with CNNs are state-of-the-art in many text classification tasks [15], a traditional VSM approach is used instead, as it seems to perform better when dealing with large texts [31]. The vocabulary size is limited to 10k as it was a good balance of network complexity and accuracy.

Furthermore, a second approach is proposed based on the addition of semantic information, similarly to the method described in [29]. To semantically enrich the album texts, we adopted Babelfy, a state-of-the-art tool for entity linking [26], a task to associate, for a given textual fragment candidate, the most suitable entry in a reference KB. Babelfy maps words from a given text to Wikipedia [4]. In Wikipedia, categories are used to organize resources. We take all the Wikipedia categories of entities identified by Babelfy in each document and add them at the end of the text as new words. Then a VSM with tf-idf weighting is applied to the semantically enriched texts, where the vocabulary is also limited to 10k terms. Note that either words or categories may be part of this vocabulary.

From this representation, a feed forward network with two dense layers of 2048 neurons and a Rectified Linear Unit (ReLU) after each layer is trained to predict the genre labels in both LOGISTIC and COSINE configurations.

5.3 Image-based Approach

Every album in the dataset has an associated cover art image. To perform music genre classification from these images, we use Deep Residual Networks (ResNets) [11]. They are the state-of-the-art in various image classification tasks like Imagnet [35] and Microsoft COCO [19]. ResNet is a common feed-forward CNN with *residual learning*, which consists on bypassing two or more convolution layers. We employ a slightly modified version of the original ResNet [5]: the scaling and aspect ratio augmentation are obtained from [41], the photometric distortions from [12], and weight decay is applied to all weights and biases. The network we use is composed of 101 layers (ResNet-101), initialized with pretrained parameters learned on ImageNet. This is our starting point to finetune the network on the genre classification task. Our ResNet implementation has a logistic regression final layer with sigmoid activations and uses the binary cross entropy loss.

[4] http://wikipedia.org

[5] https://github.com/facebook/fb.resnet.torch/

5.4 Multimodal approach

We aim to combine all of these different types of data into a single model. There are several works claiming that learning data representations from different modalities simultaneously outperforms systems that learn them separately [10, 28]. However, recent work in multimodal learning with audio and text in the context of music recommendation [31] reflects the contrary. We have observed that deep networks are able to find an optimal minimum very fast from text data. However, the complexity of the audio signal can significantly slow down the training process. Simultaneous learning may under-explore one of the modalities, as the stronger modality may dominate quickly. Thus, learning each modality separately warrants that the variability of the input data is fully represented in each of the feature vectors.

Therefore, from each modality network described above, we separately obtain an internal feature representation for every album after training them on the genre classification task. Concretely, the input to the last fully connected layer of each network becomes feature vector for its respective modality. Given a set of feature vectors, $l2$-regularization is applied on each of them. They are then concatenated into a single feature vector, which becomes the input to a simple Multi Layer Perceptron (MLP), where the input layer is directly connected to the output layer. The output layer may have either a LOGISTIC or a COSINE configuration.

6. EXPERIMENTS

We apply the architectures defined in the previous section to the *MuMu* dataset. The dataset is divided as follows: 80% for training, 10% for validation, and 10% for test. We first evaluate every modality in isolation in the multi-label genre classification task. Then, from each modality, a deep feature vector is obtained for the best performing approach in terms of AUC. Finally, the three modality vectors are combined in a multimodal network. All results are reported in Table 2. Performance of the classification is reported in terms of AUC score and Coverage@k with $k = 1, 3, 5$. The training speed per epoch and number of network hyperparameters are also reported. All source code and data splits used in our experiments are available on-line [6].

The matrix of album genre annotations of the training and validation sets is factorized using the approach described in Section 4.1, with a value of $d = 50$ dimensions. From the set of album factors, those annotated with a single label from the top level of the taxonomy are plotted in Figure 1 using t-SNE dimensionality reduction [21]. It can be seen how the different albums are properly clustered in the factor space according to their genre.

6.1 Audio Classification

We explore three network design parameters: convolution filter size, number of filters per convolutional layer,

[6] https://github.com/sergiooramas/tartarus

Table 2. Results for Multi-label Music Genre Classification of Albums

Modality	Target	Settings	Params	Time	AUC	C@1	C@3	C@5
Audio	LOGISTIC	TIMBRE-MLP	0.01M	1s	0.792	0.04	0.14	0.22
Audio	LOGISTIC	LOW-3X3	0.5M	390s	0.859	0.14	0.34	0.54
Audio	LOGISTIC	HIGH-3X3	16.5M	2280s	0.840	0.20	0.43	0.69
Audio	LOGISTIC	LOW-4X96	0.2M	140s	0.851	0.14	0.32	0.48
Audio	LOGISTIC	HIGH-4X96	5M	260s	0.862	0.12	0.33	0.48
Audio	LOGISTIC	LOW-4X70	0.35M	200s	0.871	0.05	0.16	0.34
Audio	LOGISTIC	HIGH-4X70	7.5M	600s	0.849	0.08	0.23	0.38
Audio	COSINE	LOW-3X3	0.33M	400s	0.864	0.26	0.47	0.65
Audio	COSINE	HIGH-3X3	15.5M	2200s	0.881	0.30	0.54	0.69
Audio	COSINE	LOW-4X96	0.15M	135s	0.860	0.19	0.40	0.52
Audio	COSINE	HIGH-4X96	4M	250s	0.884	0.35	0.59	0.75
Audio	COSINE	LOW-4X70	0.3M	190s	0.868	0.26	0.51	0.68
Audio (A)	**COSINE**	**HIGH-4X70**	**6.5M**	**590s**	**0.888**	**0.35**	**0.60**	**0.74**
Text	LOGISTIC	VSM	25M	11s	0.905	0.08	0.20	0.37
Text	LOGISTIC	VSM+Sem	25M	11s	0.916	0.10	0.25	0.44
Text	COSINE	VSM	25M	11s	0.901	0.53	0.44	0.90
Text (T)	**COSINE**	**VSM+Sem**	**25M**	**11s**	**0.917**	**0.42**	**0.70**	**0.85**
Image (I)	LOGISTIC	ResNet	1.7M	4009s	0.743	0.06	0.15	0.27
A + T	LOGISTIC	MLP	1.5M	2s	0.923	0.10	0.40	0.64
A + I	LOGISTIC	MLP	1.5M	2s	0.900	0.10	0.38	0.66
T + I	LOGISTIC	MLP	1.5M	2s	0.921	0.10	0.37	0.63
A + T + I	**LOGISTIC**	**MLP**	**2M**	**2s**	**0.936**	**0.11**	**0.39**	**0.66**
A + T	COSINE	MLP	0.3M	2s	0.930	0.43	0.74	0.86
A + I	COSINE	MLP	0.3M	2s	0.896	0.32	0.57	0.76
T + I	COSINE	MLP	0.3M	2s	0.919	0.43	0.74	0.85
A + T + I	COSINE	MLP	0.4M	2s	0.931	0.42	0.72	0.86

Number of network hyperparameters, epoch training time, AUC-ROC, and catalog coverage at $k = 1, 3, 5$ for different settings and modalities.

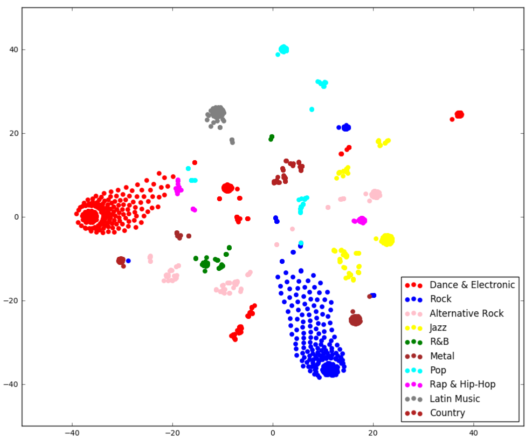


Figure 1. t-SNE of album factors.

and target layer. For the filter size we compare three approaches: square 3x3 filters as in [5], a filter of 4x96 that convolves only in time [45], and a musically motivated filter of 4x70, which is able to slightly convolve in the frequency domain [34]. To study the width of the convolutional layers we try with two different settings: HIGH with 256, 512, 1024, and 1024 in each layer respectively, and LOW with 64, 128, 128, 64 filters. Max-pooling is applied after each convolutional layer. Finally, we use the two different network targets defined in Section 4, LOGISTIC and COSINE. We empirically observed that dropout regularization only helps in the HIGH plus COSINE configurations. Therefore we applied dropout with a factor of 0.5 to these configurations, and no dropout to the others.

Apart from these configurations, a baseline approach is added. This approach consists in a traditional audio-based approach for genre classification based on the audio descriptors present in the MSD [1]. More specifically, for each song we aggregate four different statistics of the 12 timbre coefficient matrices: mean, max, variance, and $l2$-norm. The obtained 48 dimensional feature vectors are fed into a feed forward network as the one described in Section 5.4 with LOGISTIC output. This approach is denoted as TIMBRE-MLP.

The results show that CNNs applied over audio spectrograms clearly outperform traditional approaches based on handcrafted features. We observe that the TIMBRE-MLP approach achieves 0.792 of AUC, contrasting with the 0.888 from the best CNN approach. We note that the LOGISTIC configuration obtains better results when using a lower number of filters per convolution (LOW). Configurations with fewer filters have less parameters to optimize, and their training processes are faster. On the other hand, in COSINE configurations we observe that the use of a higher number of filters tends to achieve better performance. It seems that the fine-grained regression of the factors benefits from wider convolutions. Moreover, we observe that 3x3 square filter settings have lower performance, need more time to train, and have a higher number of parameters to optimize. By contrast, networks using time convolutions only (4X96) have a lower number of parameters, are faster to train, and achieve comparable performance. Furthermore, networks that slightly convolve across the frequency bins (4X70) achieve better results with only a slightly higher number of parameters and training time. Finally, we observe that the COSINE regression approach achieves better AUC scores in most configurations, and also their results are more diverse in terms of catalog coverage.

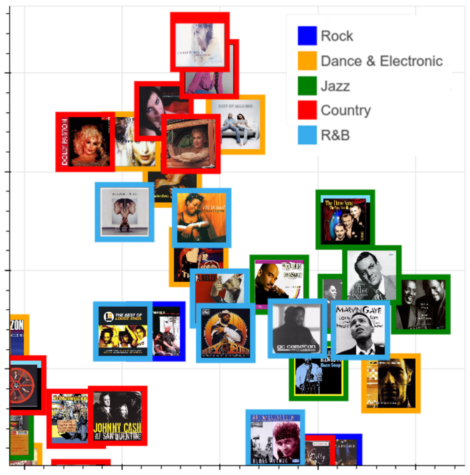


Figure 2. Particular of the t-SNE of randomly selected image vectors from five of the most frequent genres.

6.2 Text Classification

For text classification, we obtain two feature vectors as described in Section 5.2: one built from the texts VSM, and another built from the semantically enriched texts VSM+SEM. Both feature vectors are trained in the multi-label genre classification task using the two output configurations LOGISTIC and COSINE.

Results show that the semantic enrichment of texts clearly yields better results in terms of AUC and diversity. Furthermore, we observe that the COSINE configuration slightly outperforms LOGISTIC in terms of AUC, and greatly in terms of catalog coverage. The text-based results are overall slightly superior to the audio-based ones.

We also studied the information gain of words in the different genres. We observed that genre labels present in the texts have important information gain values. However, it is remarkable that *band* is a very informative word for Rock, *song* for Pop, and *dope*, *rhymes*, and *beats* are discriminative features for Rap albums. Place names have also important weights, as *Jamaica* for Reggae, *Nashvile* for Country, or *Chicago* for Blues.

6.3 Image Classification

Results show that genre classification from images has lower performance in terms of AUC and catalog coverage compared to the other modalities. Due to the use of an already pre-trained network with a logistic output (ImageNet [35]) as initialization of the network, it is not straightforward to apply the COSINE configuration. Therefore, we only report results for the LOGISTIC configuration.

In Figure 2 a set of cover images of five of the most frequent genres in the dataset is shown using t-SNE over the obtained image feature vectors. In the left top corner the ResNet recognizes women faces on the foreground, which seems to be common in Country albums (red). The jazz albums (green) on the right are all clustered together probably thanks to the uniform type of clothing worn by the people of their covers. Therefore, the visual style of the cover seems to be informative when recognizing the album genre. For instance, many classical music albums include an instrument in the cover, and Dance & Electronics covers are often abstract images with bright colors, rarely including human faces.

6.4 Multimodal Classification

From the best performing approaches in terms of AUC of each modality (i.e., AUDIO / COSINE / HIGH-4X70, TEXT / COSINE / VSM+SEM and IMAGE / LOGISTIC / RESNET), a feature vector is obtained as described in Section 5.4. Then, these three feature vectors are aggregated in all possible combinations, and genre labels are predicted using the MLP network described in Section 5.4. Both output configurations LOGISTIC and COSINE are used in the learning phase, and dropout of 0.7 is applied in the COSINE configuration.

Results suggest that the combination of modalities outperforms single modality approaches. As image features are learned using a LOGISTIC configuration, they seem to improve multimodal approaches with LOGISTIC configuration only. Multimodal approaches that include text features tend to improve the results. Nevertheless, the best approaches are those that exploit the three modalities of *MuMu*. COSINE approaches have similar AUC than LOGISTIC approaches but a much better catalog coverage, thanks to the spatial properties of the factor space.

7. CONCLUSIONS

An approach for multi-label music genre classification using deep learning architectures has been proposed. The approach was applied to audio, text, image data, and their combination. For its assessment, *MuMu*, a new multimodal music dataset with over 31k albums and 135k songs has been gathered. We showed how representation learning approaches for audio classification outperform traditional handcrafted feature based approaches. Moreover, we compared the effect of different design parameters of CNNs in audio classification. Text-based approaches seem to outperform other modalities, and benefit from the semantic enrichment of texts via entity linking. While the image-based classification yielded the lowest performance, it helped to improve the results when combined with other modalities. Multimodal approaches appear to outperform single modality approaches, and the aggregation of the three modalities achieved the best results. Furthermore, the dimensionality reduction of target labels led to better results, not only in terms of accuracy, but also in terms of catalog coverage.

This paper is an initial attempt to study the multi-label classification problem of music genres from different perspectives and using different data modalities. In addition, the release of the *MuMu* dataset opens up a number of unexplored research possibilities. In the near future we aim to modify the ResNet to be able to learn latent factors from images as we did in other modalities and apply the same multimodal approach to other MIR tasks.

8. ACKNOWLEDGMENTS

This work was partially funded by the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502).

9. REFERENCES

[1] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, 2011.

[2] Dmitry Bogdanov, Alastair Porter, Perfecto Herrera, and Xavier Serra. Cross-collection evaluation for music classification tasks. In *ISMIR*, 2016.

[3] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[4] Kahyun Choi, Jin Ha Lee, and J Stephen Downie. What is this song about anyway?: Automatic classification of subject using user interpretations and lyrics. In *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 453–454. IEEE Press, 2014.

[5] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *ISMIR*, 2016.

[6] Keunwoo Choi, George Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. *arXiv preprint arXiv:1609.04243*, 2016.

[7] François Chollet. Information-theoretical label embeddings for large-scale image classification. *CoRR*, pages 1–10, 2016.

[8] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen. Audio-based music classification with a pretrained convolutional network. In *ISMIR*, 2011.

[9] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6964–6968. IEEE, 2014.

[10] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Towards score following in sheet music images. *ISMIR*, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Andrew G Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013.

[13] Xiao Hu, J Stephen Downie, Kris West, and Andreas F Ehmann. Mining music reviews: Promising preliminary results. In *ISMIR*, 2005.

[14] Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 935–944. ACM, 2016.

[15] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751, 2014.

[16] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. Multimodal music mood classification using audio and lyrics. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, pages 688–693. IEEE, 2008.

[17] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[18] Janis Libeks and Douglas Turnbull. You can judge an artist by an album cover: Using images for music annotation. *IEEE MultiMedia*, 18(4):30–37, 2011.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[20] Beth Logan et al. Mel frequency cepstral coefficients for music modeling. In *ISMIR*, 2000.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[22] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Rhyme and style features for musical genre classification by song lyrics. In *ISMIR*, 2008.

[23] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[24] Brian Mcfee, Colin Raffel, Dawen Liang, Daniel P W Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and Music Signal Analysis in Python. *Proc. of the 14th Python in Science Conf.*, (Scipy):1–7, 2015.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013.

[26] Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity Linking meets Word Sense Disambiguation: A Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244, 2014.

[27] Robert Neumayer and Andreas Rauber. Integration of text and audio features for genre classification in music information retrieval. In *European Conference on Information Retrieval*, pages 724–727. Springer, 2007.

[28] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[29] Sergio Oramas, Luis Espinosa-Anke, Aonghus Lawlor, et al. Exploring customer reviews for music genre classification and evolutionary studies. In *ISMIR*, 2016.

[30] Sergio Oramas, Francisco Gómez, Emilia Gómez, and Joaquín Mora. Flabase: Towards the creation of a flamenco music knowledge base. In *ISMIR*, 2015.

[31] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. A Deep Multimodal Approach for Cold-start Music Recommendation. *ArXiv e-prints*, June 2017.

[32] Sergio Oramas, Vito Claudio Ostuni, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):21, 2016.

[33] François Pachet and Daniel Cazaly. A taxonomy of musical genres. In *Content-Based Multimedia Information Access-Volume 2*, pages 1238–1245, 2000.

[34] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on*, pages 1–6. IEEE, 2016.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[36] Chris Sanden and John Z. Zhang. Enhancing multi-label music genre classification through ensemble techniques. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 705–714, New York, NY, USA, 2011. ACM.

[37] Alexander Schindler and Andreas Rauber. An audio-visual approach to music genre classification through affective color features. In *European Conference on Information Retrieval*, pages 61–67. Springer, 2015.

[38] Christian Schörkhuber and Anssi Klapuri. Constant-Q transform toolbox for music processing. *7th Sound and Music Computing Conference*, (JANUARY):3–64, 2010.

[39] Klaus Seyerlehner, Markus Schedl, Tim Pohle, and Peter Knees. Using block-level features for genre classification, tag classification and music similarity estimation. *Submission to Audio Music Similarity and Retrieval Task of MIREX*, 2010, 2010.

[40] Bob L Sturm. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, pages 29–66. Springer, 2012.

[41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[43] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3), 2006.

[44] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[45] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 2643–2651, 2013.

[46] Fei Wang, Xin Wang, Bo Shao, Tao Li, and Mitsunori Ogihara. Tag integrated multi-label music style classification with hypergraph. In *ISMIR*, 2009.

[47] Justin Zobel and Alistair Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1):18–34, 1998.

THE SIGNIFICANCE OF THE LOW COMPLEXITY DIMENSION IN MUSIC SIMILARITY JUDGEMENTS

Jeff Ens
Simon Fraser University
jeffe@sfu.ca

Bernhard E. Riecke
Simon Fraser University
ber1@sfu.ca

Philippe Pasquier
Simon Fraser University
pasquier@sfu.ca

ABSTRACT

Previous research has demonstrated that similarity judgements are context specific, as they are shaped by cultural exposure, familiarity, and the musical aesthetic of the content being compared. Although such research suggests that the criterion for similarity judgement varies with respect to the musical style of the content being compared, the specific musical factors which shape this criterion are unknown. Since dimensional complexity differentiates musical genres, and has been shown to affect similarity judgements following lifelong exposure, this experiment investigates the short-term influence of dimensional complexity on similarity judgements. Rhythmic and pitch sequences with two levels of complexity were factorially combined to create four distinct types of prototype melodies. 51 participants rated the similarity of each type of prototype melody (M) to two variations, one in which the pitch content was modified $(\bar{M}_p)$, and another in which the rhythmic content was modified $(\bar{M}_r)$. The results indicate that rhythm and pitch complexity both play a significant role, influencing the perceived similarity of $\bar{M}_p$, and $\bar{M}_r$. The dimension bearing low complexity information was found to be the predominant factor in similarity judgements, as participants found modifications to this dimension to significantly decrease perceived similarity.

1. INTRODUCTION

Similarity directly informs our experience of music, enabling the perception of cohesion within a musical work, and the categorization of musical works. Consequently, developing models that encapsulate the manner in which similarity is perceived, is of critical importance within the areas of Musicology, Music Cognition and Music Theory [30]. In particular, the search for robust and flexible similarity measures has dominated research in the Music Information Retrieval (MIR) domain, as large digital databases of music information necessitate content-based querying and retrieval, and classification. Although there is a large body of research that explores similarity perception within music, many aspects of similarity perception are not yet fully understood. The current study corroborates previous evidence that similarity criterion vary with respect to the musical content being compared [9], demonstrating that the complexity of pitch and rhythmic content influence similarity perception.

Since pitch and rhythm are the two most prominent musical dimensions in the context of symbolic notation, the current study will manipulate complexity along these dimensions and observe the effects on similarity perception. Although no musical dimensions are completely orthogonal, as a modification in a particular dimension may affect the perception of other dimensions, the complexity of pitch and rhythmic content can be measured independently, and there is evidence that these dimensions are processed separately in cognition [13, 27]. Therefore, pitch and rhythm complexity were considered to be independent for the purposes of this study. *Pitch content* refers to the sequence of pitches encapsulated in a particular melody, and *rhythm content* refers to the sequence of durations. *Dimensional complexity* refers to the absolute level of complexity along a particular musical dimension. In this study we measure the dimensional complexity of pitch and rhythm content.

2. RELATED WORK

Previous work examining the perception of musical similarity, has focused on establishing a hierarchy of musical dimensions, ranking their observed contributions to similarity perception. On a whole, most research claims that rhythmic information is the most important. Halpern [7] constructed 16 melodies — a factorial combination of two pitch sequences, two rhythmic sequences, two tonal structures and forward and reversed versions — and found that rhythm was the most important distinguishing factor, followed by pitch, direction and tonal structure. Similarly, Rosner and Meyer [19] found rhythm to be the strongest determinant of melodic similarity. Despite the general consensus that rhythm plays a dominant role in similarity judgements, pitch still plays a considerable role. Dowling [2] demonstrated that a modified imitation of a prototype melody is often misidentified as the prototype when it has a similar pitch contour.

Given the multidimensional nature of music, many researchers have found it useful to make the distinction between surface-level and structural features. In general, surface-level attributes include contour, loudness and tempo while structural attributes denote aspects of form, thematic development and patterns. In short term contexts, where participants are unfamiliar with the musical material being compared, surface-level features are a strong predictor of both melodic [15, 19, 22] and polyphonic [9] similarity. Prince [15] found that rhythm was the dominant aspect informing perceived melodic similarity, followed by contour, meter, and tonal structure.

However, there is increasing evidence which questions the generality of these results, as contextual factors including familiarity, cultural exposure, and the aesthetic of the musical content being compared, have been shown to have a considerable effect on similarity perception. Pollard-Gott found that with repeated listening, surface level features became less influential and thematic material became more important [14]. Similarly, the long term analysis of a collection of folk melodies by a panel of experts, placed emphasis on thematic and motivic similarity above all other factors [31]. Schubert and Stevens [22] found that contour is more important than harmonic structure for making similarity comparisons, but with musical expertise, harmonic structure also has an effect.

Other research has shown that cultural exposure affects similarity perception. Hannon and Trehub [8] found the metrical bias of North American adults to be the result of an enculturation processes, with no evidence of a natural predisposition for the simple meters which characterize much of western music. Goldstone [6] suggests that humans learn by focusing on perceptual features that are more informative, at the cost of decreased attention towards other dimensions. This phenomenon has been observed in a musical context, where the voice that consists of immediate and exact repetitions of a short musical fragment tends to perceptually decrease in salience for the listener over time [24]. Instead, the listener is naturally drawn to focus on the high complexity voice. Since distinct rhythmic durations occur at a relatively higher frequency than distinct pitches in western music, they demand less attention than pitch content. After years of exposure, this likely results in an increased sensitivity to the pitch content in a melody [17]. Notably, Eerola et al. [3] demonstrated that musical complexity perceptions are shaped by exposure to different musical culture, which likely results from the mechanisms described above.

In addition to the factors mentioned above, music aesthetic has been shown to influence how similarity is perceived. Lamont and Dibben [9] examined similarity relationships in two contrasting musical styles, requiring participants to rate the similarity of extracts from a Beethoven sonata (op. 10, no. 1, first movement) and a dodecaphonic work composed by Schoenberg (Klavierstück op. 33a). Nine polyphonic excerpts were selected from each piece, each approximately eight measures long, and the similarity of each possible combination was rated by participants, resulting in 36 similarity ratings for each piece. Notably, both pieces are composed for solo piano, and have more than one theme which is developed throughout the duration of each work. They found that similarity judgements were primarily based on surface level features, however, the similarity judgements for each piece were predominantly influenced by different surface features. These results suggested that each piece establishes a different similarity criterion within which listeners make appropriate similarity judgements. Although Lamont and Dibben demonstrated that the criterion for similarity judgements varies with respect to the musical aesthetic of the stimuli being compared, the specific musical factors which caused this phenomenon are still unknown, directly motivating our experiment.

3. MOTIVATION

As evidenced by the brief overview in section 2, numerous studies have demonstrated the prevalent influence of contextual factors on musical similarity judgements [8, 9, 14, 17, 31], directly motivating further study in this area. Since contextual factors like cultural exposure and familiarity are difficult to integrate into a similarity measure, this study examines the third contextual factor, the role of the musical content itself in shaping a criterion for similarity judgements. The phenomenon that Lamont and Dibben [9] observed, provides evidence that musical content influences the manner in which music is compared, as participants used different musical dimensions to make comparisons depending on the nature of the musical content. In light of this evidence, it is worthwhile to examine how specific musical characteristics of the content being compared shape similarity judgements, which does not appear to have been examined previously. Due to the fact that dimensional complexity differentiates musical genres [3], and affects similarity judgements following lifelong exposure [8], this experiment investigates the short-term influence of dimensional complexity on melodic similarity judgements. More specifically, this study investigates the role of dimensional complexity in shaping awareness to modifications in that particular dimension, effectively establishing a criterion for melodic similarity judgements.

Previous research has shown that limitations on the human capacity for musical memory, have an effect on musical perception. Participants found it more difficult to retain melodies with complex contours, which were devoid of any repetition, and were often unable to distinguish them from another complex contour [18]. Moreover, complexity was one of four variables which collectively predicted the recognizability of melodies when presented a second time [20]. In these cases, it seems likely that working memory limitations make it difficult to encapsulate all aspects of a complex melody on first exposure. In summarizing recent research on working memory limitations, Cowan [1] proposes that there is a capacity of three to five chunks in working memory for young adults. According to these findings, modifications to the musical dimension bearing the least complex musical material should be the easiest to detect, which suggests that this musical dimension would have a predominant influence on similarity judgements. Collectively, this research supports the following hypothesis: modifications to the musical dimension bearing low complexity information will result in a significant decrease in similarity, in comparison to similar modifications to the musical dimension bearing high complexity information.

4. METHODOLOGY

4.1 Participants

The participants were recruited online using the Crowdflower [1] crowdsourcing platform, and required to pass a test before participating in the experiment. Participants were paid $0.02 USD for each question they answered, in accordance with the typical compensation offered to Crowdflower users. Of the 96 participants who took the test, 76 passed (79.2%) and 63 completed the experiment. 12 participants responses were deemed ineligible based on the inconsistent responses to an identical question. In total, 51 participants came from 25 different countries.

4.2 Stimuli

4.2.1 Measuring Complexity

Given the multifaceted nature of complexity, it is necessary to make the distinction between the entropy based complexity measures proposed by Eerola et al. [3], and the notion of complexity which grounds the current study. Shannon Entropy quantifies the disorder or uncertainty inherent in an information source based on a representative probability distribution [23]. Eerola et al. calculate entropy using the marginal probability of each symbol in a sequence. This type of complexity will be referred to as *entropy*$_m$. Although entropy$_m$ has been shown to correlate with the percieved complexity of musical sequences [16], this measurement of complexity does not provide the necessary resolution to make comparisons between many musical sequences. For an explicit example, consider the following pitch sequences, $s_1 = \{$c, d, e, f, c, d, e, f$\}$, and $s_2 = \{$c, f, e, d, e, c, d, f$\}$. Even though s_1 exhibits less complexity than s_2, both s_1 and s_2 have the same entropy$_m$, as this measurement does not take the repetition of longer phrases into consideration. Clearly, it is necessary to take the repetition of phrases into consideration when measuring complexity.

Admittedly, this can be accomplished by calculating the entropy rate of an n-th order markov chain derived from the musical sequence being measured, however there are still issues with this approach. In contrast to the manner in which humans percieve musical content, and by extension musical complexity, the entropy rate is not designed to distiguish between repetition which occurs within the prevailing metric structure, and repetition which spans metrical boundaries. Research suggests that humans perceive music by breaking it into a series of chunks [5], and have a natural tendency to project metre onto sequences of sound, despite the absence of acoustic cues for metric organization [4]. In addition, when listening to music, humans naturally extract motivic patterns [32], and larger formal structures [12]. Since humans segment music in accordance with metrical boundaries, it is likely that humans are less sensitive to repetition which is obscured by these boundaries. Consequently, a true measure of musical complexity must take this distinction into account.

Furthermore, an entropy based model of complexity is not capable of taking similarity into consideration, as entropy is based on the lossless encoding of an information source [23]. This becomes more of an issue when entropy is being measured with respect to larger subsequences, as is the case when measuring the n-th order entropy rate. This formulation of complexity cannot make the distinction between a collection of subsequences which share the same contour, and a collection that does not. As a result, it seems most reasonable to take the collective dissimilarity of subsequences segmented with respect to the prevailing metric structure, as a measure of complexity. Consequently, a homogenous collection of segments would be perceived as having a low complexity, while a diverse collection of segments would be perceived as having a high complexity. We use the term *redundancy* to refer to this type of complexity throughout the paper.

In order to quantify redundancy, two different measures were used. Thul's [28] adaptation of Tanguiane's [25, 26] algorithm, measures redundancy by counting the number of *root patterns*, at several hierarchical levels. This will be referred to as *Tanguiane's Rhythmic Complexity* (*TRC*). The other measure of redudancy is calculated using Eqn (1), where (S) is a set of subsequences, derived by segmenting a sequence of symbols into measures. Notably, Eqn (1) also requires a distance metric (D). Chronotonic distance [29] is used to measure *Rhythmic Sequence Complexity* (*RSC*), and a similarity measure proposed by Maidín [10] is used to measure *Pitch Sequence Complexity* (*PSC*). Admittedly, segmenting a pitch sequence according to metre means that *PSC* is dependant on the rhythmic content, however, within-measure rhythmic patterns have no bearing on *PSC* in this paradigm, and the metric structure is not being manipulated in this study. Although *PSC* does not account for the complexity of invidual segments, section 4.2.2 describes how complexity is restricted in this experiment, effectively mitigating the variance of segment complexity in the current study.

$$f(S) = \frac{1}{|S|}\sum_{i=1}^{|S|} \min\{D(S_i, S_j) : j \neq i; 1 \leq j \leq |S|\} \tag{1}$$

4.2.2 Prototype Melodies

In this experiment, there were four types of melodies; *rhythm*$_s$-*pitch*$_s$, *rhythm*$_s$-*pitch*$_c$, *rhythm*$_c$-*pitch*$_s$, and *rhythm*$_c$-*pitch*$_c$, where s denotes a simple or low complexity sequence, and c denotes a complex sequence [2]. In addition, eight versions of each melody type were constructed, resulting in 32 (4×8) prototype melodies of equal

[1] https://www.crowdflower.com/

[2] The melodies used in this experiment can be found at https://mlab%2Dexperiments.iat.sfu.ca/ismir2017/audio.

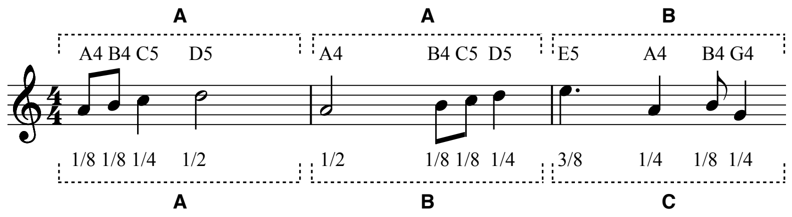


Figure 1. A melody with complex rhythm and simple pitch, using letters to show the form of each dimension.

length (three measures). As mentioned in section 4.2.1, redundancy quantifies the degree to which an information source is self similar and contains periodic repetition in conjunction with the prevailing metrical structure. In light of this aim, melodies were comprised of three measure-length phrases, with phrase repetition varied to create two distinct levels of complexity. Low complexity sequences had a formal pattern AAB, where a pattern is repeated in the first two measures, and a new pattern is introduced in the last measure. High complexity sequences had a formal pattern ABC, where each measure is dissimilar. This construction process is demonstrated in Figure 1, which shows a high complexity rhythm sequence and a low complexity pitch sequence.

Care was taken to restrict the variability of entropy$_m$ based complexity, using measures proposed by Eerola et al. [3]. Since the pitch sequences were constructed from scales consisting of five distinct pitch classes, *Entropy of pitch class distribution* and *Entropy of interval distribution* did not vary significantly. Similarly, rhythm sequences were constructed from four distinct durations, limiting the variance of *Entropy of note duration distribution* and *Rhythmic variability*. Notably, it seemed reasonable to have fewer distinct durations than pitch classes, as research has demonstrated that most listeners are able to perceive pitch diversity more readily [17]. A One-Way Analysis of Variance (ANOVA) across all four prototype melody types demonstrated that none of these entropy$_m$ based complexity measures were a significant source of variance, while *PSC*, *RSC* and *TRC* varied significantly. Furthermore, the entropy rate – calculated using a first order markov chain – did not vary significantly across melody type. This verified that our experiment measured the effect of variations in redundancy in relative isolation.

In order to restrict the variance of segment complexity, *Mean interval size* and *Note density* were restricted, which Eerola et al. [3] found to be a significant source of complexity. Each melody was constrained to an octave range, restricting the *Mean interval size*. The *Note density*, was invariant for each constructed melody, as each melody had four notes per measure, and was three measures long.

4.2.3 Modified Melodies

For each prototype melody (M), two modified versions were constructed for the main experiment: a version in which the pitch is modified $(\bar{M}_p)$, and a version in which the rhythm is modified $(\bar{M}_r)$. This process involved reversing the order of the measures in the dimension which is to be modified. As a result, regardless of the nature of the prototype melody, the first and last measures of the modified melody were different. Since test questions required a ground truth answer, three additional types of modified melodies were constructed: a melody in which the pattern form of M was transformed from AAB to ABA in the pitch dimension $(\bar{M}_{r\bar{p}})$, a melody in which the pattern form of M was transformed from AAB to ABA in the rhythm dimension $(\bar{M}_{p\bar{r}})$, and a melody in which both dimensions were modified $(\bar{M}_b)$.

4.3 Experimental Design

The experiment consisted of two independent variables, rhythm and pitch content complexity. Both rhythm and pitch complexity had two levels, low and high. This resulted in a 2×2 repeated measures experimental design, with four distinct types of prototype melodies. Participants were presented with a series of questions, consisting of a prototype melody (M) and two modified melodies (*melody A*, *melody B*). There were two types of test questions, which were developed using the modified melodies described above. The first type of question, compared either $\bar{M}_{r\bar{p}}$ and M against the prototype M, or $\bar{M}_{p\bar{r}}$ and M against M. This had an indisputable answer, as one of the modified melodies was in fact an exact replica of the prototype. The second type of question, compared $\bar{M}_p$ and $\bar{M}_b$ to the prototype, or compared $\bar{M}_r$ and $\bar{M}_b$ to the prototype. Given the manner in which these melodies were constructed, $\bar{M}_p$ and $\bar{M}_r$ are more similar to the prototype, as they are identical to the prototype along a single dimension, while $\bar{M}_b$ is dissimilar in both dimensions.

For the actual experiment itself, there was a single type of question, in which $\bar{M}_r$ and $\bar{M}_p$ were compared against the prototype. Irregardless of the type of question, the two modified melodies were randomly assigned to be *melody A* or *melody B*. For each question, participants rated the similarity of *melody A* to M, and *melody B* to M, on a Likert scale from 1 to 20, where 20 indicates maximal similarity. In the analysis below, the difference $(D = S(M, \bar{M}_r) - S(M, \bar{M}_p))$ between the perceived similarity of $\bar{M}_r$ to M $(S(M, \bar{M}_r))$, and the perceived similarity of $\bar{M}_p$ to M $(S(M, \bar{M}_p))$, is taken as the dependent variable. As a result, a positive value of D indicates that modifications to the rhythm dimension have less of an effect on similarity than modifications to the pitch dimension, while a negative value of D indicates the opposite.

4.4 Procedure

Before participating in the experiment, participants were required to complete 10 test questions with a minimum accuracy of 80%. The test questions served two purposes, eliminating those who were not taking the task seriously, and familiarizing participants with the similarity domain within which they were being asked to make comparisons. Once the test was successfully completed, participants were presented with 10 randomly ordered questions, consisting of eight different experiment questions (representing each of the eight different types of prototype melodies), a test question, and a repeated experiment question. The repeated experiment question was used to determine if participants were answering the questions consistently. For each question, the prototype melody was selected randomly from a collection of eight versions, and the key was randomly transposed so that the content varied from question to question. After listening to all three melodies, participants were asked to indicate which of the two modified versions was more similar to the prototype, and rate the similarity of *melody A* and *melody B* on a Likert scale from 1 to 20.

5. RESULTS

Since the ANOVA is relatively robust to violations of normality [21], the 2-Way ANOVA was conducted without transforming the data, despite the violation of the assumption of normality. A 2-Way ANOVA revealed the main effect of rhythm complexity ($F(50) = 9.17, p = .004, \eta_p^2 = .155$) and pitch complexity ($F(50) = 5.31, p = .025, \eta_p^2 = .096$), while the interaction between rhythm complexity and pitch complexity was insignificant ($p = .657$). To be thorough, an Aligned Rank Transform was performed on the data, correcting for the effects of the non-normal distributions of the data [33]. Using the transformed data, a 2-Way ANOVA revealed main effect of rhythm complexity ($F(50) = 9.82, p = .003, \eta_p^2 = .164$) and pitch complexity ($F(50) = 6.26, p = .016, \eta_p^2 = .111$), while the interaction between rhythm complexity and pitch complexity was insignificant ($p = .601$). These results corroborate the analysis of the untransformed data, indicating that 16.4% of the variability in similarity ratings were explained by changes in rhythm complexity, and 11.1% of the variability was explained by changes in pitch complexity.

As predicted, there was a main effect of rhythm complexity and pitch complexity, both shown in Figure 2b. Melodies containing low complexity rhythmic content ($M = 0.451, SD = 5.26$) were significantly lower than those containing high complexity rhythmic content ($M = 2.49, SD = 5.81$), which indicates that participants were more sensitive to pitch modifications when pitch sequences were less complex. This effect was pronounced in cases where the rhythmic sequence was more complex, as participants found pitch modified melodies ($\bar{M}_p$) to be significantly less similar to *rhythm$_c$-pitch$_s$* prototype melodies than rhythm modified melodies ($\bar{M}_r$). Conversely, melodies containing low complexity pitch content ($M = 2.26, SD = 5.43$) were significantly higher than those containing high complexity pitch content ($M = 0.676, SD = 5.73$), which indicates that participants were more sensitive to rhythmic modifications when rhythmic sequences were less complex. Similarly, this effect was pronounced in cases where the pitch sequence was more complex, as participants found rhythm modified melodies ($\bar{M}_r$) to be significantly less similar to *rhythm$_s$-pitch$_c$* prototype melodies than pitch modified melodies ($\bar{M}_p$). Therefore, the dimension bearing low complexity musical content was found to play a significant role in similarity judgements, as modifications to that dimension significantly decreased perceived similarity.

An analysis of the individual prototype melody conditions revealed that the *rhythm$_s$-pitch$_c$* condition ($M = -0.235, SD = 5.21$) was significantly less than the *rhythm$_c$-pitch$_s$* condition ($M = 3.39, SD = 5.39$), as pitch modified melodies were the most similar to *rhythm$_s$-pitch$_c$* prototypes, and rhythm modified melodies were the most similar to *rhythm$_c$-pitch$_s$* prototypes. The *rhythm$_s$-pitch$_s$* condition ($M = 1.14, SD = 5.27$) and the *rhythm$_c$-pitch$_c$* condition ($M = 1.59, SD = 6.12$) were roughly equivalent, and participants did not find a particular type of modified melody to be more similar, relative to the two other conditions. Collectively, these results indicate that melodies which are modified in the dimension bearing low complexity information are perceived as significantly less similar than melodies which are modified in the dimension bearing high complexity information.

6. DISCUSSION

As evidenced by the results presented above, modifications to the dimension bearing low complexity information result in a significant decrease in perceived similarity, demonstrating that the dimension bearing low complexity information plays a more significant role in melodic similarity judgments. On a whole, the values for all four conditions were positively skewed (Figure 2a), indicating that modifications to the pitch content of a melody had a greater influence on perceived similarity. Since there is no benchmark with which to compare rhythmic sequence complexity and pitch sequence complexity, it was not possible to equate the complexity across dimensions. Consequently, some skew in either direction was expected. The positive skew may indicate that the rhythmic content of the melodies in this experiment was on average more complex, and participants had difficulty noticing modifications in the rhythm dimension. Alternatively, due to the enculturation process that Hannon and Trehub [8] observed, participants may have paid more attention to the pitch content, resulting in the slight positive skew. When these factors are considered, it is arguably most meaningful to interpret the conditions in relation to each other, as some skew in either direction was inevitable. Viewed from this perspective, the hypothesis is directly corroborated, as the *rhythm$_s$-pitch$_c$*

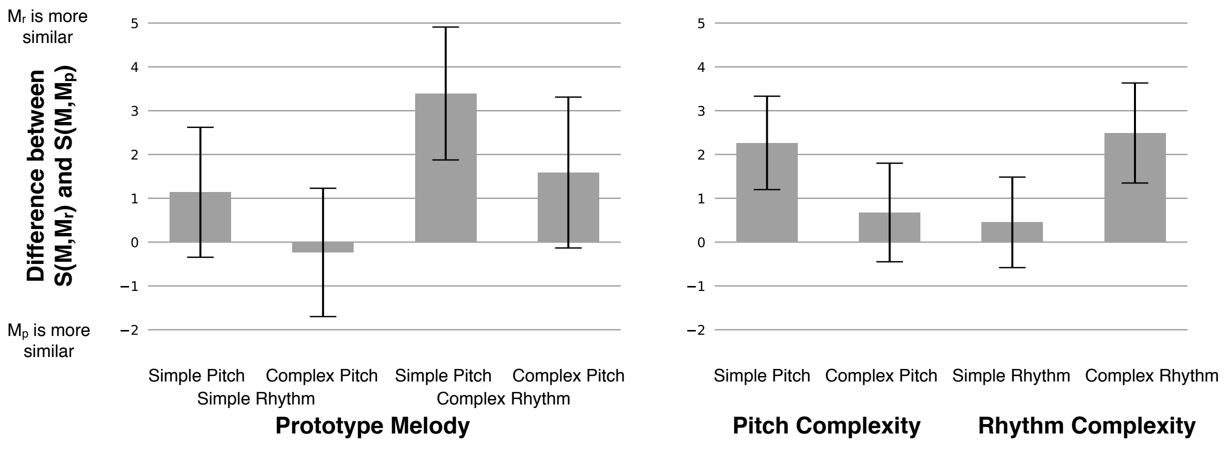


Figure 2. (a) The difference between the perceived similarity of the modified rhythm melody and the perceived similarity of the modified pitch version for each prototype melody complexity category, with 95% confidence intervals. (b) The main effects of pitch and rhythm complexity with 95% confidence intervals

condition is the lowest, the *rhythm*$_c$-*pitch*$_s$ is the highest, and the *rhythm*$_s$-*pitch*$_s$ and *rhythm*$_c$-*pitch*$_c$ conditions are in the middle.

Further analysis reveals that previous experiments are likely a special case of the generalized theory proposed in this paper. Monahan et al. [11] and Halpern [7] both make the claim that rhythm contributes more significantly to similarity perception, however, the rhythmic component of their stimuli is predominantly low complexity, and the pitch component of their stimuli is relatively higher on average. Notably, this was measured using *PSC*, *RSC*, and *TRC*. Although Halpern and Monahan et al. attribute their results to an inherent bias towards rhythm, the results of this experiment suggest that the relative complexity of the rhythm and pitch content provides a more robust explanation.

Admittedly, there are several limitations to the generalization of the results of this study. First and foremost, the observed relationship between dimensional complexity and similarity judgements may manifest itself quite differently when working with longer melodies, or polyphonic music. Secondly, due to the fact that musical complexity is multifaceted and far from understood, determining the relatively low complexity dimension may be quite difficult in some contexts. Despite the aforementioned limitations, the limited variance of Eerola et al.'s entropy based complexity measures provides substantial support for the generalization of these findings, as most western music makes use of the same limited collection of distinct note durations and pitch classes [16]. As a result, although this form of entropy based complexity is the source of some variability within the musical cannon, redundancy arguably accounts for more of this variation. Consequently, the results of this study are not restricted to a particular genre, and are relevant across musical genres.

7. CONCLUSION

Similarity is shaped by several factors, including familiarity, and cultural conditioning. This study asserts the significance of another factor – the nature of the musical content which is being compared – by examining the effects of dimensional complexity on similarity judgements. The general notion that characteristics of the musical content being compared have some bearing on the criterion used to make similarity judgements, is not new, and has been observed in past experiments [9]. However, the manner in which musical content establishes a criterion for similarity judgements has not been explored previously. The results of this study provide evidence that pitch and rhythmic complexity are factors which shape the criterion used in similarity judgements, as the dimension bearing relatively low complexity information has a greater influence on similarity perception. Furthermore, the results of this experiment are corroborated by previous experiments [7, 11], offering a general explanation for these previous findings.

Developing robust and flexible similarity measures continues to be a dominant area of research in the MIR domain, as large digital databases of music information necessitate accurate methods for comparison and categorization. As a result, adapting existing similarity measures to take dimensional complexity into account, is a possible application of the findings of this study. Future research is also necessary to investigate the role of complexity along other dimensions, including dynamics, articulation and timbre. Furthermore, the manner in which complexity is percieved along a single dimension is in need of continued exploration, as several issues with pre–existing methods for measuring complexity have been discussed in section 4.2.1. Clearly, musical similarity is a complex phenomenon which is deserving of continued exploration, as the results of this experiment have explicitly demonstrated that similarity judgements are dependant on another contextual factor, the complexity of pitch and rhythm content in the musical material being compared.

8. REFERENCES

[1] N. Cowan. The Magical Mystery Four: How Is Working Memory Capacity Limited, and Why? *Current Directions in Psychological Science*, 19(1):51–57, 2010.

[2] W. J. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354, 1978.

[3] T. Eerola, T. Himberg, P. Toiviainen, and J. Louhivuori. Perceived complexity of western and African folk melodies by western and African listeners. *Psychology of Music*, 34(3):337–371, 2006.

[4] P. Fraisse. Rhythm and Tempo. In Diana Deutsch, editor, *The Psychology of Music*, pages 149–181. Academic Press, New York, 1982.

[5] R. I. Godøy, A. R. Jensenius, and K. Nymoen. Chunking in music by coarticulation. *Acta Acustica united with Acustica*, 96(4):690–700, 2010.

[6] R. L. Goldstone. Learning to perceive while perceiving to learn. In R Kimchi, M Behrmann, and C Olson, editors, *Perceptual Organization in Vision: Behavioural and Neural Perspectives*, pages 233–278. Lawrence Erlbaum Associates, New Jersey, 2003.

[7] A. R. Halpern. Perception of structure in novel music. *Memory & cognition*, 12(2):163–170, 1984.

[8] E. E. Hannon and S. E. Trehub. Metrical categories in infancy and adulthood. *Psychological Science*, 16(1):48–55, 2005.

[9] A. Lamont and N. Dibben. Motivic Structure and the Perception of Similarity. *Music Perception: An Interdisciplinary Journal*, 18(3):245–274, 2001.

[10] D. O. Maidín. A geometrical algorithm for melodic difference. *Computing in musicology: a directory of research*, (11):65–72, 1998.

[11] C. B. Monahan and E. C. Carterette. Pitch and Duration as Determinants of Musical Space. *Music Perception: An Interdisciplinary Journal*, 3(1):1–32, 1985.

[12] C. Neuhaus, T. R. Knösche, and A. D. Friederici. Similarity and repetition: An ERP study on musical form perception. *Annals of the New York Academy of Sciences*, 1169:485–489, 2009.

[13] I. Peretz and M. Coltheart. Modularity of music processing. *Nature neuroscience*, 6(7):688–691, 2003.

[14] L. Pollard-Gott. Emergence of Thematic Concepts in Repeated Listening to Music. *Cognitive psychology*, 15(1):66–94, 1983.

[15] J. B. Prince. Contributions of pitch contour, tonality, rhythm, and meter to melodic similarity. *Journal of experimental psychology. Human perception and performance*, 40(6):2319–37, 2014.

[16] J. B. Prince and P. Q. Pfordresher. The role of pitch and temporal diversity in the perception and production of musical sequences. *Acta Psychologica*, 141(2):184–198, 2012.

[17] J. B. Prince, M. A. Schmuckler, and W. F. Thompson. The effect of task and pitch structure on pitch-time interactions in music. *Memory & cognition*, 37(3):368–381, 2009.

[18] T. W. Reiner. Pitch-distance and contour complexity in the recognition of short melodies. *Journal of Scientific Psychology*, (September):27–36, 2011.

[19] B. S. Rosner and L. B. Meyer. The perceptual roles of melodic process, contour, and form. *Music Perception*, 4(1):1–39, 1986.

[20] P. A. Russell. Memory for music: A study of musical and listener factors. *British Journal of Psychology*, 78(3):335–347, 1987.

[21] E. Schmider, M. Ziegler, E. Danay, L. Beyer, and M. Bühner. Is It Really Robust?: Reinvestigating the robustness of ANOVA against violations of the normal distribution assumption. *Methodology*, 6(4):147–151, 2010.

[22] E. Schubert and C. Stevens. The effect of implied harmony, contour and musical expertise on judgments of similarity of familiar melodies. *Journal of New Music Research*, 35(2):161–174, 2006.

[23] C.E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(July):379–423, 1948.

[24] C. Taher, R. Rusch, and S. McAdams. Effects of Repetition on Attention in Two-Part Counterpoint. *Music Perception*, 33(3):306–318, 2016.

[25] A. S. Tanguiane. *Artificial Perception and Music Recognition*. Springer-Verlag, Berlin, 1993.

[26] A. S. Tanguiane. A Principle of Correlativity of Perception and Its Application to Music Recognition. *Music Perception: An Interdisciplinary Journal*, 11(4):465–502, 1994.

[27] W. F. Thompson, M. D. Hall, and J. Pressing. Illusory conjunctions of pitch and duration in unfamiliar tone sequences. *J Exp Psychol Hum Percept Perform*, 27(1):128–140, 2001.

[28] E. Thul. *Measuring the Complexity of Musical Rhythm*. PhD thesis, McGill University, 2008.

[29] G. T. Toussaint. A comparison of rhythmic similarity measures. *Proc. International Conference on Music Information Retrieval (ISMIR 2004)*, pages 242–245, 2004.

[30] A. Volk, E. Chew, E. Hellmuth Margulis, and C. Anagnostopoulou. Music Similarity: Concepts, Cognition and Computation. *Journal of New Music Research*, 45(3):207–209, 2016.

[31] A. Volk and P. Kranenburg. Melodic similarity among folk songs: An annotation study on similarity-based categorization in music. *Musicae Scientiae*, 16(3):1–23, 2012.

[32] R. L. Welker. Abstraction of Themes from Melodic Variations. *Journal of Experimental Psychology. Human Perception and Performance*, 8(3):435–447, 1982.

[33] J. O. Wobbrock, L. Findlater, D. Gergle, and J. J. Higgins. The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 143–146, 2011.

TOWARDS COMPUTATIONAL MODELING OF THE UNGRAMMATICAL IN A RAGA PERFORMANCE

Kaustuv Kanti Ganguli **Preeti Rao**
Department of Electrical Engineering,
Indian Institute of Technology Bombay, Mumbai, India.
{kaustuvkanti,prao}@ee.iitb.ac.in

ABSTRACT

Raga performance allows for considerable flexibility in interpretation of the raga grammar in order to incorporate elements of creativity via improvisation. It is therefore of much interest in pedagogy to understand what ungrammaticality might mean in the context of a given raga, and possibly develop means to detect this in an audio recording of the raga performance. One prominent notion is that ungrammaticality is considered to occur only when the performer “treads” on another, possibly allied, raga in a listener’s perception. With this view, we consider modeling the technical boundary of a raga as that which separates it from another raga that is closest to it in its distinctive features. We wish to find computational models that can indicate ungrammaticality using a data-driven estimation of the model parameters; i.e. the raga performances of great artists are used to obtain representations that discriminate most between same and different raga performances. We choose a well-known pair of allied ragas (Deshkar and Bhupali in north Indian classical music) for an empirical study of computational representations for the distinctive attributes of tonal hierarchy and melodic shape of a chosen common descending phrase.

1. INTRODUCTION

The melodic framework in Indian art music is governed by the system of ragas. A raga can be viewed as falling somewhere between a scale and a tune in terms of its defining grammar which includes the tonal material, tonal hierarchy, and characteristic melodic phrases [26, 31]. Description of the raga grammar, as found in text resources or as verbalized in pedagogy, typically comprises of a listing of the allowed notes (svara) of the 12-tone scale, ascending and descending svara patterns, the mention of the most important svaras and a list of common phrases (svara sequences). While the texts do not explicitly describe the precise svara intonations or the actual melodic shapes of the phrases in terms of the transitions between svaras, the audio analyses of raga performances has demonstrated what is well-known to practitioners, i.e. the shape of the continuous pitch contour corresponding to the phrase is characteristic of the raga and therefore relatively invariant across performances in a given raga [28, 29].

The raga grammar can be viewed as a set of constraints within which creativity is given a free hand to realise whatever is aesthetically pleasing in the all-important melodic improvisation component, so characteristic of the genre. It is therefore of interest to understand, and possibly model, the technical boundary of a raga in terms of what might constitute ungrammaticalilty in a performance. The technical boundary would be specified in terms of the defining attributes such as tonal material and hierarchy, and phrase shapes. Such an exercise could lead to the development of computational tools for assessing performance accuracy in pedagogy together with the complementary aspect of creative skill. A popular notion of grammaticality in performance occurs around the notion of preserving a raga’s essential distinctiveness in terms of the knowledgeable listener’s perception [1–3, 5, 9, 24, 27, 30]. Thus, a performance with possibly many creative elements can still be considered not to transgress the raga grammar as long as it does not “tread on another raga” [24, 27, 35]. The technical boundary of a raga should therefore ideally be specified in terms of limits on the defining attributes where it is expected that the limit depends on the proximity of other ragas with respected to the selected attribute.

The computational modeling of the distinctive attributes of a raga has been the subject of previous research motivated by the task of raga recognition from audio given a large training dataset of performances across several ragas. The tonal material has been represented by a variety of first order pitch distributions and experimental outcomes based on recognition performance have been used to comment on the relative superiority of a given representation [4, 6–8, 11, 17, 22]. Motivated by the pitch-continuous nature of the melody, finely-binned histograms of octave-folded instantaneous pitch values have been used as templates in raga recognition tasks [8, 22]. Alternately, 12-bin distributions of pitch values within detected stable svara regions have been used to represent a raga’s tonal content [7, 11]. Melodic shape invariance of phrases, on the other hand, has been used in the modeling of similarity measures for the task of melodic motif detection within and across performances of a given raga [12,19–21,28,29]. Pitch contour shape is typically represented by a tonic-normalized time-series or reduced to a symbol string of the raga notes [14, 16].

In this work, we consider the computational modeling of tonal hierarchy and phrase shape based on maximizing the discrimination of 'close' ragas with respect to the given attribute. Such an approach has not been used in previous work. The notion of "allied ragas" is helpful here where we consider ragas that share the same grammar in major attributes while differing in a few. For example, the pentatonic ragas Deshkar and Bhupali have the same set of svaras (S, R, G, P, D corresponding to 0, 200, 400, 700, and 900 cents respectively) and common phrases in terms of svara sequences, e.g. the descending phrase GRS. Learners are typically introduced to the two ragas together and warned against confusing them [24, 30, 34]. Recently, subjective experiments on perceived similarity by musicians of synthetically manipulated raga phrases clearly demonstrated the existence of a sharp boundary between valid variants of a given raga phrase from variants of the same phrase (i.e. in terms of svara sequence) from an allied raga [15].

In the present work, we consider the pair of allied ragas, Deshkar and Bhupali, and use the performances of eminent Hindustani vocalists as proxy for creatively expressed, but grammatically accurate, examples of the stated raga. The performances in the allied raga are likewise considered to be examples of the corresponding ungrammatical renderings. We evaluate known, as well as some new, representations in terms of the achieved discrimination on a dataset of performances across the two ragas. Although the scope of the experiments is restricted to the given pair of ragas and chosen attributes, we expect our outcomes to be generalizable.

In the next section, we introduce our dataset along with the necessary musicological background, and describe the audio processing required to obtain the continuous pitch track that forms the basis for the computational representations under study. This is followed by sections that discuss potential representations for tonal hierarchy and phrase shape with associated distance measures. We next present an experimental study of the discrimination performance followed by our conclusions.

2. DATASET AND AUDIO PROCESSING

Table 1 presents a comparison of the melodic attributes corresponding to the grammars of the allied ragas as compiled from musicology texts. These cover the aspects of duration and intonation of the tonal material that includes ascending (*Ar*) and descending (*Av*) scales, dominant (*Vadi*) and subdominant (*Samvadi*), and characteristic phrases. 'Natural shruti' (last row) refers to the Just Intonation tuning, but there is no quantification of the term 'higher'. Also, there is some indication of a duration constraint on R (as a short or passing svara relative to its neighbours) in the form of braces (e.g. G(R)S in raga Deshkar).

2.1 Dataset and Annotation

The audio recordings used in this study are drawn from the Hindustani music corpus from 'Dunya' [1] compiled as a representative set of the vocal performances in the genre [33]. The editorial metadata for each audio recording is publicly available on the metadata repository MusicBrainz [2]. The Dunya corpus for raga Deshkar comprises 5 concerts of which 4 are selected for the current study, omitting the drut (fast tempo) concert due to the distinctly different style of realising phrases associated with such tempi [24]. Similarly, we selected 5 concerts for the Bhupali test set. We augmented the overall dataset, as described in Table 2, by additional concerts from personal collections.

Next, we annotate the occurrences of the chosen phrase, GRS. As discussed earlier, the GRS phrase is common to the two ragas and a frequently used descending motif. The GRS phrases are distributed across three octaves (upper, middle, and lower octaves), although lower octave instances are fewer. The segmentation of the phrases from the alap and vistar (i.e. improvised sections of the concert) is carried out semi-automatically as follows. A musician indicated the coarse location of each instance of the chosen phrase; this was then refined to obtain segmentation boundaries by automatic onset and offset detection methods described later. A count of the phrases used in this study is presented per concert in Table 2. The phrase-level annotation of the remaining concerts is underway for future work.

2.2 Pitch Time-series Extraction from Audio

Predominant-F0 detection is implemented by an algorithm proposed by [32] that exploits the spectral properties of the voice with temporal smoothness constraints on the pitch contour. The pitch is detected at 10 ms intervals with zero pitch assigned to the detected purely instrumental regions. The pitch values in Hz are converted to the cents scale by normalizing with respect the concert tonic determined automatically using a multi-pitch approach [18]. The final pre-processing step is to interpolate short silence regions

[1] https://dunya.compmusic.upf.edu/Hindustani
[2] https://musicbrainz.org/

Deshkar	Bhupali
Tonal material: SRGPD	Tonal material: SRGPD
Ar: SGPD, SPDS *Av*: S, PDGP, DPG(R)S	*Ar*: SRG, PDS *Av*: SDP, GDP, GRS
Vadi: D, *Samvadi*: G	*Vadi*: G, *Samvadi*: D
Phrases: SG, G(P)DPD, P(D)SP, DGP, DPG(R)S	Phrases: RDS, RPG, PDS, SDP, GDP, GRS
Higher shrutis of R, G, D	Natural shrutis of R, G, D

Table 1. Specification of raga grammar for the two allied ragas of the present study [1, 5, 25, 30]

Raga	# Concerts	Duration (hours)	# Artists	# GRS phrases
Deshkar	6	2:16:50	5	52
Bhupali	6	3:22:00	5	107

Table 2. Description of the test dataset.

below a threshold (250 ms which is empirically chosen as proposed by [13]) indicating musically irrelevant breath pauses or unvoiced consonants, by cubic spline interpolation, to ensure the integrity of the melodic shape. A median filtering with a 50 ms window is performed to get rid of spurious pitch excursions. Eventually, we obtain a continuous time-series of pitch values representing the melody line throughout the vocal regions of the concert.

3. MELODIC REPRESENTATIONS

Our goal is to propose computational representations that robustly capture particular melodic characteristics of the raga in a performance while being sensitive enough to the differences between allied ragas. Given that tonal material and hierarchy of svaras are an important component of the raga grammar, we consider representations of tonal hierarchy computable from the melody extracted from the audio recording of the performance. We also consider the representation of the melodic shape of a selected characteristic phrase.

3.1 Representation of Tonal Hierarchy

Tonal hierarchies, manifested in the relative frequencies and durations with which the tones are sounded in a musical piece, have been linked to key identification in Western art music. In a widely known work, Krumhansl [23] used a 12-element vector to code the total duration (in terms of number of beats) of each note of the chromatic scale in a piece and correlated it with each of 24 templates representing the major and minor keys to obtain accurate predictions of key. It is therefore logical to consider the same representation for raga discrimination. However, given the pitch-continuous nature of the music, we are faced with multiple competing options in the definition of a tonal representation. Closest to the tonal hierarchy vector of Krumhansl is the 12-element histogram of the total duration of each of the "stable notes" detected from the melodic contour. Considering the importance of the transitions connecting stable notes as well as microtonal differences in intonation between the same svara in different ragas, a histogram derived from all the pitch values in the melodic contour would seem more suitable. The bin width for such a pitch continuous distribution is also a design choice we must make. Finally, we need a distance measure computable between the histogram representations that correlates well with closeness of the compared performances in terms of raga identity.

3.1.1 Pitch Salience Histogram

The input to the system is tonic normalized pitch contour (cents vs time). The pitch values are octave-folded (0 - 1200 cents) and quantized to p bins of equal width (the bin resolution is $\frac{1200}{p}$). The bin centres are arithmetic mean of the adjacent bin edges. The salience of each bin is proportional to the accumulated duration of the pitch value corresponding to that bin. The normalization is to construct a probability distribution fuction (pdf) where the area under the histogram sums to unity. Given the number of bins, the histogram is computed as:

$$H_k = \sum_{n=1}^{N} \mathbb{1}_{[c_k \le F(n) \le c_{k+1}]} \tag{1}$$

where H_k is the salience of the k^{th} bin, $F(n)$ is the array of pitch values, (c_k, c_{k+1}) are the bounds of the k^{th} bin and $\mathbb{1}$ is an indicator random variable [3]. Figure 1 shows the pitch salience histogram for $p = 1200$ (1 cent bin resolution). For a bin resolution of 100 cents, the representation is equivalent to the conventional pitch class distribution (PCD) [7].

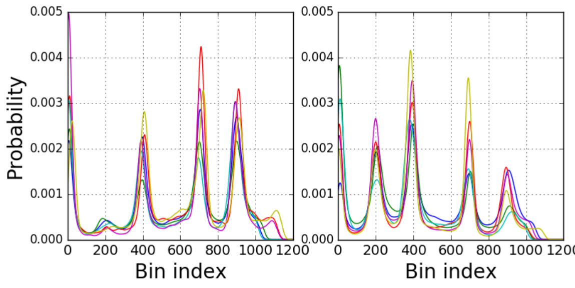


Figure 1. Pitch salience histograms (octave folded, 1 cent bin resolution) of 6 concerts each in ragas Deshkar (left) and Bhupali (right).

3.1.2 Svara Salience Histogram

The svara salience histogram is not equivalent to the PCD. The input to the system is segmented stable svaras which is a subset of the pitch contour. We use a previously proposed algorithm [16] that obtains a simple melodic transcription retaining only the stable svara regions of a pitch contour while discarding the transitory pitch regions. The stable svara regions are segmented by identifying the fragments of pitch contour that are within T_{tol} (35 cents) of the svara frequencies that are located via the peaks of a continuous pitch histogram. Next, the svara fragments that are smaller than T_{dur} (250 ms) in duration are filtered out, as they are too short to be considered as perceptually meaningful held svaras [28]. This leaves a string of fragments each labeled by a svara. Fragments with the same svara value that are separated by gaps less than 100 ms are merged. The svara salience histogram is obtained as:

[3] A random variable that has the value 1 or 0, according to whether a specified event occurs or not is called an indicator random variable for that event.

$$H_k = \sum_{n=1}^{N} \mathbb{1}_{[F(n) \in S_k, k \in (1,2,\cdots,12)]} \tag{2}$$

where H_k is the salience of the k^{th} bin, $F(n)$ is the array of pitch values, and S_k is the k^{th} svara of the octave. H_k is always a 12-element vector. Figure 2 shows the tonal hierarchy in the form of svara salience histogram. One major difference between pitch salience histogram and svara salience histogram is that the precise intonation information is lost in the latter.

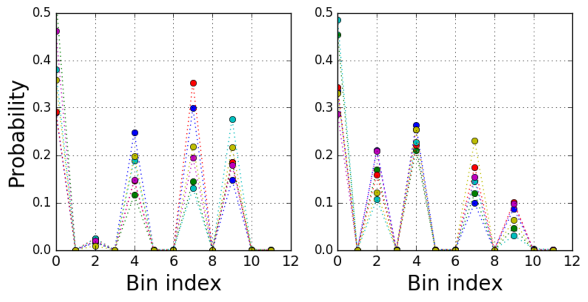


Figure 2. Svara salience histograms (octave folded) of 6 concerts each in ragas Deshkar (left) and Bhupali (right).

3.2 Representation of Phrase Shape

The phrase is a sequence of svara whose melodic realization includes specific intonations and transitions to/from neighboring svaras [24]. While computational models for measuring melodic similarity between phrases have employed distance measures between time-series of pitch values of the phrase segments, we might expect that a more discriminative representation is possible by explicitly incorporating features that contrast the two ragas.

Figure 3 shows a representative GRS phrase from each of the ragas. Distinctive features suggested by the comparison are: (i) durations of each of the stable svara regions, (ii) the durations of the glides connecting the svaras, and (iii) the pitch interval of the svara G. The implementation of these features would involve decisions on segmentation of stable svaras, and determining the pitch interval value from the pitch continuum in the region. Further, it is important to figure out the kind of normalization that is needed to reduce possible variability due to the tempo of the performance.

We describe a phrase as a sequence of melodic 'events' that can each be described by the chosen features. For the GRS phrase in question, we consider the following five events, i.e. svaras $\mathrm{G, R, S}$, and the $\mathrm{G - R}$ and $\mathrm{R - S}$ transitions. The selected features are: (i) $Start\_time$: onset of an event, (ii) End_time : offset of an event, (iii) $Duration$: difference of the two, (iv) $Intonation$: precise pitch interval location of a stable svara in the octave obtained as the median pitch value over the duration of the svara, and (v) $Slope$: gradient between the mean of last 20% and the first 20% pitch samples of a stable svara segment.

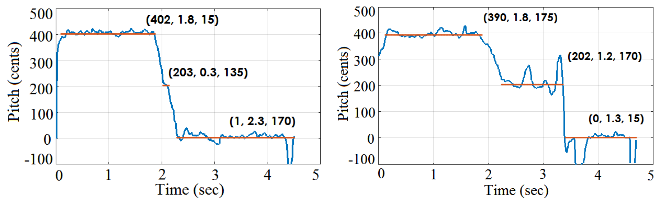


Figure 3. Two representative GRS phrases from ragas Deshkar (left) and Bhupali (right). The tuple corresponding to each svara denotes the extracted features $(Intonation, Duration, Slope)$ for that event.

4. EXPERIMENTAL RESULTS AND DISCUSSION

We present experiments that help us identify the aspects of optimal representation, for each of tonal hierarchy and phrase shape, that discriminate the two ragas maximally based on our labeled dataset of 12 concerts across two ragas. Our common approach for both attributes is to use unsupervised clustering (k-means with k=2) of the feature vectors and optimise the separation between the clusters over the considered choices for feature implementation. Performance in unsupervised clustering can be measured via the 'cluster purity' which is obtained by assigning each obtained cluster to the underlying class that is most frequent in that cluster, and computing the resulting classification accuracy.

We also use the receiver operated characteristic (ROC) curve, with respect to detecting similarity within a raga pair given the feature vectors and distance measure, to evaluate the tuning parameters of tonal hierarchy. An objective function to compare ROCs across configurations is the area under curve (AUC) measure. Closer the AUC value to 1, better is the performance. For the evaluation of phrase shape, we additionally use 'feature selection' to estimate the most significant features.

4.1 Tonal Hierarchy

There are three main aspects of tonal hierarchy that need to be addressed in order to maximize the separation between the two clusters, these are: (i) representation, (ii) distance measure, and (iii) time-scale of analyses. The experiments apply to both pitch salience and svara salience histograms. We describe them in order with discussions on the insights obtained.

4.1.1 Optimal Representation

Our base representation is the octave folded pitch salience histogram normalized to be a pdf. We try different bin resolutions ranging from 1 to 100 cents, with a denser sampling between 20 and 40 cents, motivated from a previous study by Datta et al. [10]. We perform an unsupervised clustering (euclidean distance) to obtain labels for each element in the 2 classes. Figure 4 shows the cluster purity values, where we note that no degradation in the evaluation measure is observed for 1 through 30 cents bin resolution. Each value on the curve is obtained by an average of 5 runs of the clustering algorithm to nullify the effect of any local minima. For the case of svara salience histograms, the average cluster purity value is obtained as 0.96.

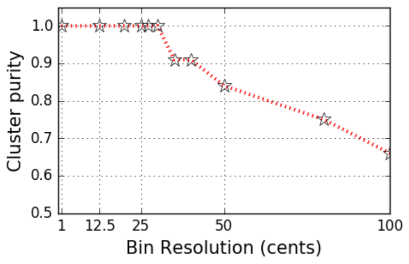


Figure 4. Cluster purity values obtained for different values of bin resolution for the pitch salience histograms.

4.1.2 *Comparison of Distance Measures*

In order to determine a suitable distance metric between the histograms representing raga tonal hierarchy, we test different metrics on the 25 cent binned pitch salience histogram (given that no degradation was observed at this resolution in the previous experiment) and on the svara salience histogram. As the distribution is a pdf, a natural distance measure suggested in the literature is the (symmetric) KL divergence [22]. We also try Cityblock (L-1 norm), Euclidean (L-2 norm), and Correlation distances. The last one is strongly motivated by the cognitive model of Krumhansl [23]. We present ROC curves (and AUC) for four distance measures in Figure 5. We find that the best performing distance measure is the KL divergence for svara salience histograms. The performance of cityblock distance is observed to be comparable to that of correlation distance for pitch salience histograms. We use the latter in the following experiments since it is more favored in previous work [6, 23].

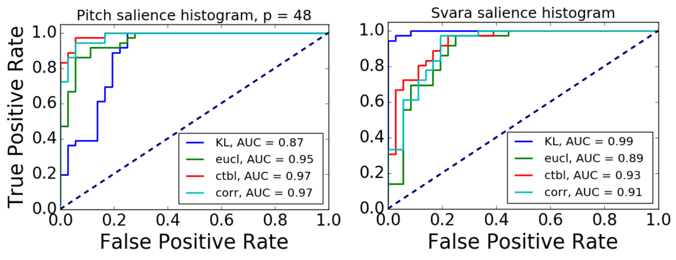


Figure 5. ROCs obtained for four different distance measures from pitch salience (left) and svara salience (right) histograms.

4.1.3 *Time-scale for a Stable Tonal Hierarchy*

It is of interest to determine at what time-scale, the measured tonal hierarchy qualifies as a stable representation. We therefore carry out the previous discrimination experiments on segmented concerts. We divide each concert to its $\left(\frac{1}{n}\right)^{th}$ (n = 1, $\cdots$, 5) portion and construct a distance matrix with each $\left(\frac{1}{n}\right)^{th}$ part. The goal is to find out the minimum proportion of the full concert that is necessary to robustly discriminate between the two classes. The ROC obtained from $\left(\frac{1}{4}\right)^{th}$ (and below this) of a concert results in an AUC < 0.5 which indicates that the time-scale is too small to constitute a stable tonal hierarchy. We augment our test dataset by considering each half and $\left(\frac{1}{3}\right)^{rd}$ of each concert, making the dataset size 24 and 36 respectively. Figure 6 shows a comparison of best performing systems for the full (n=1) and partial (n=2,3) concerts. We additionally investigate if a finer binned pitch salience histogram shows improvement. A finer bin resolution of 12.5 cents (p=96) is observed to perform better that the 25 cent binned pitch salience histogram (p=48). This suggests that the microtonal differences in intonation become more important when the concert segment duration is not long enough to capture 12-tone hierarchy in a stable manner. While the improvement (in terms of AUC value) might seem rather small, it was observed to be consistent with each distance measure under test. Moreover with full-concerts (n=1) for p=96, the maximum true positive rate for zero false positive rate is higher ($\approx$ 0.95) than that for p=48 ($\approx$ 0.75) indicating an improved performance. We also separately observed that considering only the beginning segments in the $\left(\frac{1}{n}\right)^{th}$ portions shows better performance compared to other locations. The first few minutes of each concert spans the alap and bandish (composition) that play a crucial role in raga delineation thus adhering closely to raga grammar.

4.2 Phrase Shape

We use the svara segmentation method outlined in Section 3.1.2 to obtain the components of the phrase shape corresponding to the sequence of stable svaras as well as the transient regions. In this section we present a statistical description of the features corresponding to the different events. We also compare the discrimination powers of the different features via a clustering experiment.

4.2.1 *Distribution of Event Durations*

Given the $Duration$ values of each event in the GRS phrases (52 instances in raga Deshkar and 107 instances in raga Bhupali), we present the distributions in Figure 7 of the event $Durations$ in the form of boxplots of the raw measurements in seconds. We observe distinctions between the two ragas in nearly all the duration parameters, most notably in the R $Durations$. That the R duration is small and constrained in raga Deshkar is supported by the raga grammar specification in Table 1 which indicates R in parantheses, suggesting a “weak note that is never sustained” [30]. Overall, the dispersion in the parameters is smaller in the phrase in raga Deshkar compared with Bhupali, consistent with the fact that it is a grammatically more constrained raga [1, 5, 30]. An exception is the realisation of the S svara with several outlying values of duration due to its location at phrase end, where a number of contextual considerations influence the note offset.

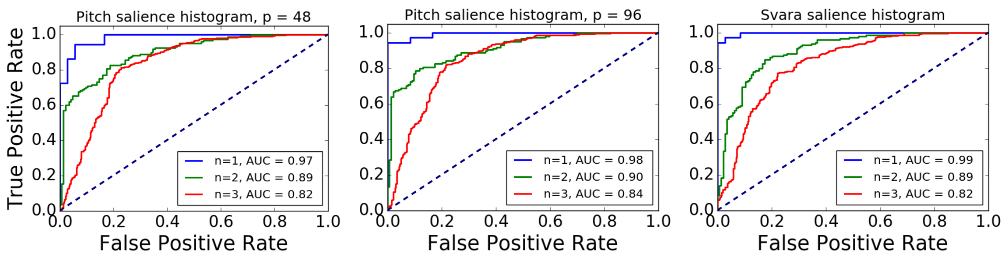


Figure 6. Comparison of ROCs obtained with correlation distance (for pitch salience histograms) and KL divergence (for svara salience histogram) for different time-scales (n=1,2,3) of the concerts.

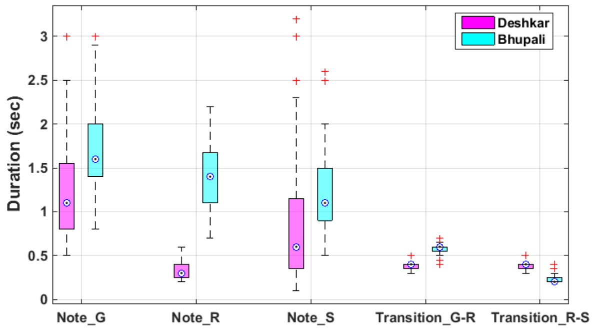


Figure 7. Distributions of event $Duration$s across the GRS phrase instances in the two ragas.

4.2.2 *Feature Selection and Evaluation*

To compare the predictive powers of the measured acoustic features, we perform 'Feature Selection' using Weka [4] datamining tool. We use the "InfoGainAttributeEval" as the attribute selector that evaluates the value of an attribute by measuring the information gain with respect to the class, in conjunction with the "Ranker" search method that ranks attributes by their individual performances. We construct a feature vector for each instance of the GRS phrase with 5 $Duration$ features, one for each event, and $Intonation$ and $Slope$ features corresponding to each of the three stable svaras as implemented in Section 3.2. Of these 11 features, we obtained the most significant features in terms of predictive power as the following: (i) R $Duration$, and (ii) G $Intonation$, with the third feature in the list placed considerably lower. This outcome is consistent with the raga grammars where these two aspects are considered distinctive properties of raga Deshkar.

Next, we perform an unsupervized clustering of the 159 phrases into two classes using the Euclidean distance between the 2-element vectors of the two selected features. The achieved cluster purity value is 0.99 (i.e. only 2 instances of the 159 are misclassified). As a next step, we investigate whether duration normalization is helpful. Given that overall phrase duration is correlated with tempo [34], it is natural to expect that the variability of phrase event durations may be reduced by normalization by the overall phrase duration. However, it turned out the the cluster purity with the duration-normalized $Duration$ (of R svara) feature coupled with the previous $Intonation$ (of G svara) feature reduced to 0.95 (i.e. 8 instances were misclassified). This indicates that musicians interpret the raga grammar in terms of raw durations rather than relative to the tempo.

[4] http://www.cs.waikato.ac.nz/ml/weka/

5. CONCLUSION AND FUTURE WORK

Based on the notion of grammaticality in raga performance, we examined computational representations for some of the key attributes of raga grammar based on discriminating allied ragas. In particular, both the pitch salience histograms and the stable-note based svara salience histograms were considered for tonal hierarchy with a variety of distance measures to derive a combination of histogram parameters and distance metric that best separated same-raga pairs from the allied-raga pairs. It was found that svara salience histograms worked best at the time-scale of full concerts whereas finer bin-widths of pitch salience histograms were superior for segmented concerts. Overall, full concerts with KL divergence as distance measure between 12-tone svara histograms performed best. A phrase level representation that considered only the discriminating elements of the same-phrase variants across the ragas in terms of absolute duration and pitch interval of key events (i.e. for R and G svaras respectively) was able to achieve a high degree of separation between the two allied ragas. Our results suggest that a pedagogy tool that measures ungrammaticality can indeed be designed based on modeling the raga attributes for any raga with the methodology presented here. Future work involves: (i) validation on other allied raga sets, (ii) correlating predicted ungrammaticality with perceived ungrammaticality by expert listeners, and (iii) determining the relative weighting of the different raga attributes for an overall rating, possibly at different time-scales, based on the expert judgments.

6. ACKNOWLEDGEMENT

This work received partial funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement 267583 (CompMusic).

7. REFERENCES

[1] Music in Motion: The automated transcription for Indian music (AUTRIM) project by NCPA and UvA. url: https://autrimncpa.wordpress.com/. Last accessed: April 26, 2017.

[2] S. Bagchee. *Nād: Understanding Raga Music*. Business Publications Inc, 1998.

[3] S. Bagchee. *Shruti: A Listener's Guide to Hindustani Music*. Rupa Co, 2006.

[4] S. Belle, R. Joshi, and P. Rao. Raga identification by using swara intonation. *Journal of ITC Sangeet Research Academy*, 23, 2009.

[5] Distinguishing between Similar Ragas. url: http://www.itcsra.org/Distinguishing-between-Similar-Ragas. Last accessed: April 26, 2017.

[6] B. Bozkurt. An automatic pitch analysis method for Turkish Maqam music. *Journal of New Music Research (JNMR)*, 37(1):1–13, 2008.

[7] P. Chordia and A. Rae. Raag recognition using pitch-class and pitch-class dyad distributions. In *Proc. of Int. Soc. for Music Information Retrieval Conf.*, pages 431–436, 2007.

[8] P. Chordia and S. Şentürk. Joint recognition of raag and tonic in north Indian music. *Computer Music Journal*, 37(3):82–98, 2013.

[9] A. Danielou. *The ragas of Northern Indian music*. Munshiram Manoharlal Publishers, 2010.

[10] A. K. Datta, R. Sengupta, N. Dey, and D. Nag. *Experimental analysis of Shrutis from performances in Hindustani music*. Scientific Research Department, ITC Sangeet Research Academy, 2006.

[11] P. Dighe, H. Karnick, and B. Raj. Swara histogram based structural analysis and identification of Indian classical ragas. In *Proc. of Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 35–40, 2013.

[12] S. Dutta, S. PV Krishnaraj, and H. A. Murthy. Raga verification in Carnatic music using longest common segment set. In *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 605–611, 2015.

[13] K. K. Ganguli, S. Gulati, X. Serra, and P. Rao. Data-driven exploration of melodic structures in Hindustani music. In *Proc. of the International Society for Music Information Retrieval (ISMIR)*, pages 605–611, August 2016. New York, USA.

[14] K. K. Ganguli, A. Lele, S. Pinjani, P. Rao, A. Srinivasamurthy, and S. Gulati. Melodic shape stylization for robust and efficient motif detection in hindustani vocal music. In *National Conference on Communications (NCC)*, 2017.

[15] K. K. Ganguli and P. Rao. Exploring melodic similarity in hindustani classical music through the synthetic manipulation of raga phrases. In *Cognitively-based Music Informatics Research*, 2016.

[16] K. K. Ganguli, A. Rastogi, V. Pandit, P. Kantan, and P. Rao. Efficient melodic query based audio search for Hindustani vocal compositions. In *Proc. of the International Society for Music Information Retrieval (ISMIR)*, pages 591–597, October 2015. Malaga, Spain.

[17] A. C. Gedik and B. Bozkurt. Pitch-frequency histogram-based music information retrieval for Turkish music. *Signal Processing*, 90(4):1049–1063, 2010.

[18] S. Gulati, A. Bellur, J. Salamon, H. G. Ranjani, V. Ishwar, H. A. Murthy, and X. Serra. Automatic tonic identification in Indian art music: Approaches and Evaluation. *Journal of New Music Research (JNMR)*, 43(1):53–71, 2014.

[19] S. Gulati, J. Serrà, V. Ishwar, S. Şentürk, and X. Serra. Phrase-based rāga recognition using vector space modeling. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 66–70, 2016.

[20] S. Gulati, J. Serrà, V. Ishwar, and X. Serra. Mining melodic patterns in large audio collections of Indian art music. In *Int. Conf. on Signal Image Technology & Internet Based Systems (SITIS-MIRA)*, pages 264–271, 2014.

[21] S. Gulati, J. Serrà, and X. Serra. Improving melodic similarity in Indian art music using culture-specific melodic characteristics. In *Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, pages 680–686, 2015.

[22] G. K. Koduri, S. Gulati, P. Rao, and X. Serra. Rāga recognition based on pitch distribution methods. *Journal of New Music Research*, 41(4):337–350, 2012.

[23] C. L. Krumhansl. *Cognitive Foundations of Musical Pitch*, chapter 4: A key-finding algorithm based on tonal hierarchies, pages 77 – 110. Oxford University Press, New York, 1990.

[24] S. Kulkarni. *Shyamrao Gharana*, volume 1. Prism Books Pvt. Ltd., 2011.

[25] V. Oak. 22 shruti. url: http://22shruti.com/. Lat accessed: April 26, 2017.

[26] H. S. Powers and R. Widdess. *India, subcontinent of*, chapter III: Theory and practice of classical music. New Grove Dictionary of Music. Macmillan, London, 2nd edition, 2001. Contributions to S. Sadie (ed.).

[27] D. Raja. *The Raga-ness of Ragas: Ragas Beyond the Grammar*. D.K. Print World Ltd., 2016.

[28] P. Rao, J. C. Ross, and K. K. Ganguli. Distinguishing raga-specific intonation of phrases with audio analysis. *Ninaad*, 26-27(1):59–68, December 2013.

[29] P. Rao, J. C. Ross, K. K. Ganguli, V. Pandit, V. Ishwar, A. Bellur, and H. A. Murthy. Classification of melodic motifs in raga music with time-series matching. *Journal of New Music Research (JNMR)*, 43(1):115–131, April 2014.

[30] S. Rao, J. Bor, W. van der Meer, and J. Harvey. *The Raga Guide: A Survey of 74 Hindustani Ragas*. Nimbus Records with Rotterdam Conservatory of Music, 1999.

[31] S. Rao and P. Rao. An overview of Hindustani music in the context of Computational Musicology. *Journal of New Music Research (JNMR)*, 43(1), April 2014.

[32] V. Rao and P. Rao. Vocal melody extraction in the presence of pitched accompaniment in polyphonic music. *IEEE Trans. on Audio, Speech & Language Processing*, 18(8), 2010.

[33] X. Serra. Creating research corpora for the computational study of music: the case of the Compmusic project. In *Proc. of the 53rd AES Int. Conf. on Semantic Audio*, London, 2014.

[34] W. van der Meer. *Hindustani Music in the 20th Century*. Martinus Nijhoff Publishers, 1980.

[35] K. G. Vijaykrishnan. *The Grammar of Carnatic Music*. De Gruyter Mouton, 2007.

A COLLECTION OF MUSIC SCORES FOR CORPUS BASED JINGJU SINGING RESEARCH

Rafael Caro Repetto, Xavier Serra
Music Technology Group, Universitat Pompeu Fabra, Barcelona
rafael.caro@upf.edu, xavier.serra@upf.edu

ABSTRACT

The MIR research on jingju (also known as Beijing or Peking opera) music has taken audio as the main source of information. Music scores are an important resource for the musicological research of this tradition, but no machine readable ones have been available for computational analysis. In order to explore the potential of symbolic score data for jingju music research, we have expanded the CompMusic Jingju Music Corpus, which contains mostly audio, with a collection of 92 machine readable scores, for a total of 897 melodic lines. Since our purpose is the study of jingju singing in terms of its musical system elements, we have selected the arias used as examples in reference jingju music textbooks. The collection is accompanied by scores metadata, curated annotations per score and melodic line, and a set of software tools for extracting statistical information from it. All the gathered data and developed software are available for research purposes. In this paper we first discuss the culture specific concepts that are needed for understanding the contents of the collection, followed by a detailed description of it. We then present a series of computational analyses performed on the scores and discuss some musicological findings.

1. INTRODUCTION

In recent years, jingju (also known as Beijing or Peking opera) music received an increasing attention from MIR researchers for tasks such as mood recognition [7], pitch contour analysis [8, 12, 13, 21–23], lyrics to audio alignment [10], timbre analysis [11, 17], percussion analysis [16, 19], and structural segmentation [18], all of them taking audio as the main source of information. In order to carry out these tasks, several corpora for jingju music research were gathered. The Jingju Music Corpus gathered in the CompMusic project [14] contains a collection of commercial audio recordings and their metadata, as well as some datasets for specific tasks. [1] The corpus built in [18] also consists of commercial recordings, annotated for structural segmentation analysis. Only the corpus created in [7] [2] contains a cappella recordings made by its authors, as well as annotations for the task of mood recognition.

The core component of jingju music is singing. In order to study the musical system developed in this tradition (see Section 1.1) it is therefore essential to analyse the sung melodic line. However, extracting its pitch contour from commercial recordings did not produce fully satisfactory results [13]. On the other hand, the a cappella recordings contained in the corpus gathered in [7] do not cover the whole jingju musical system and are not annotated for this goal, since it was created for a different purpose. Music scores are a meaningful alternative to audio recordings. The lack of MIR research based on jingju scores could be possibly motivated for two reasons: the secondary role that music scores play in this tradition (see Sections 1.1 and 1.2), and the lack of music scores in machine readable format available for computational analysis.

The goal of the Jingju Music Scores Collection (JMSC) presented in this paper is to offer a comprehensive and complete resource for the study of jingju singing in terms of its musical system. That is to say, jingju singing, as opposed to instrumental accompaniment, is taken as our research object with the aim of understanding not its acoustic or intonation characteristics, but the elements that build the jingju musical system.

1.1 Jingju Musical System

When talking about music in jingju, there are two important general considerations to take into account. First, jingju is a theatrical genre, in which the main structural element is the dramatic plot. Music, as the rest of the disciplines that integrate this comprehensive art form, is at the service of this dramatic goal, which implies that lyrics determine the music structure. Secondly, jingju music originated as folk music, [3] a fact that carries two implications. This music was not traditionally created by a composer, but arranged by the actors and musicians from a repository of folk tunes to fit the lyrics of new plays. Secondly, it has been orally transmitted, and even nowadays written material takes a secondary role.

In order to convey the expressive needs of the dramatic plot, music in jingju has been deeply convention-

[1] http://compmusic.upf.edu/corpora

[2] http://isophonics.net/SingingVoiceDataset

[3] During the 20th century it evolved into composed music.

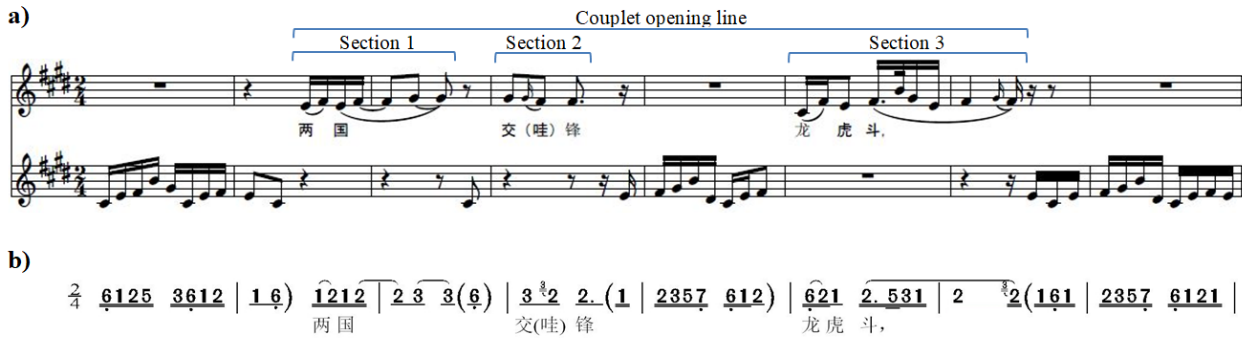


Figure 1. Example of a couplet opening line and its corresponding melodic line, both in its original *jianpu* notation (b) and the transnotated version (a) in JMSC. The three sections of the line are marked.

alised according to three elements that form its musical system, namely *shengqiang*, *banshi*, and role type. Each of the original folk tunes upon which jingju music was built evolved into a melodic framework called *shengqiang*, and it was associated to a specific emotional atmosphere. Although jingju uses many of them, *xipi* and *erhuang* are the ones that give this genre its musical identity.

More precise expressive functions are achieved by rhythmic transformations of the *shengqiang*'s melodic framework. Each of these transformations is codified in a pattern called *banshi*. There are about 12 commonly used ones, although no fixed number is agreed in the literature. The *banshi* in which the melody is conceived as the closest form to the *shengqiang*'s original tune, in medium tempo and 2/4 metre, is called *yuanban*. *Manban* is conceived as the transformation of *yuanban*'s melody to the slowest tempo range and 4/4 metre, *kuaiban* is conceived as its transformation to the fastest tempo range and 1/4 metre, and each intermediate tempo range forms a different *banshi*. Besides, there is a set of *banshi* in which the melody is rendered in different forms of free metre.

The third element is the role type (that can be understood as an acting profile), which the singing character belongs to. There are four broad categories of role types, with different degrees of subdivision. In terms of the musical system however, all of them can be classified in either the male or female styles of singing, represented respectively by the role types *laosheng* and *dan*.

A last important factor to consider is the lyrics structure, which determines the musical structure of the arias.[4] The lyrics of jingju arias are arranged in couplets, and each of the two lines of the couplet is usually subdivided in three sections. The melodic unit in jingju corresponds to the line of lyrics, so that each *shengqiang* defines a melodic line for the opening line of the couplet, and another one for the closing line, both of them subdivided into three melodic sections corresponding to the poetic ones (see Figure 1). One single aria is usually set to only one *shengqiang*, but according to its expressive content, it can contain different *banshi*. When dialogues are set to music, characters of the same or different role types can sing in the same aria.[5]

1.2 The Concepts of Work and Score in Jingju

Before describing the collection, two more considerations are needed to be taken in order to completely understand its usability and representativeness of the repertory. As stated in Section 1.1, jingju music was originally created and transmitted orally. Professional performers rarely rely on music scores, but try to convey what they learnt from their teachers, although inevitable changes occur in this transmission line. Therefore, music scores are notations of preexisting music, either for documentation purposes, or as learning material for amateurs. Most scores do not reference a source, what might indicate that the author is notating what he or she recalls as a standard version. Consequently, different score editions of the same work present noticeable discrepancies, and very rarely any of them perfectly matches a commercial audio recording of that work, although the melodic core generally is common to all.

The final remark has to do with the very concept of work.[6] The plots of jingju plays are rarely original, they are taken from well-known historical events, literary works or legends. Hence, a jingju play is just a passage from a greater encompassing story already known by the audience. Furthermore, since jingju plays did not have an author, but were adapted from these preexisting stories, performers have the possibility of adapting those passages to the needs of a specific performance. As a consequence, the title of a play always refers to the same plot, characters and usually the same set of arias, but specific performances can omit, extend or modify certain elements.

In a similar way, arias are sung passages of greater encompassing dramatic plots. Although most of them have musical signals indicating their start and end, they are usually intermingled with acting and reciting sections, what blurs their limits. When recorded or notated for commercial releases, the arias are named after its first line, but different performers or transcribers might have different criteria for deciding how to delimit the aria or how to deal

[4] Although translating the original term *changduan* as aria might cause some misconceptions, for the sake of clarity we will use this term throughout the paper to refer to sung sections in a jingju play.

[5] For a more detailed description in English of the jingju musical system, please refer to [20].

[6] We take the term work from MusicBrainz's terminology as "a distinct intellectual or artistic creation" (https://musicbrainz.org/doc/Work).

with non musical interpolations. For example, an aria set to more than one *banshi* can be recorded or transcribed in one release as a unit, whilst in another release each *banshi* can be listed separately. All these considerations pose important challenges for organizing our collection and integrating it in our Jingju Music Corpus (see Section 2.3).

The remaining of the paper is structured as follows. In the next section the jingju music scores collection is described in detail. In Section 3 we present computational analyses performed on it, and some musicological findings are discussed in Section 4. In the last section we present some concluding remarks and point out future work.

2. DESCRIPTION OF THE COLLECTION

2.1 Sources and Generation Process

In order to ensure the representativity of the selected scores, we have taken as sources three jingju music textbooks [3–5]. These textbooks have been suggested as reference works for the study of the jingju musical system by professors from the National Academy of Chinese Theatre Arts (NACTA), where the first author did field work. In these textbooks, the explanations of the music elements presented in Section 1.1 are illustrated with specific arias. In [5] and [3], the scores were given directly in the textbook. In the case of [4], some scores were given, while others were referenced by title.

	laosheng	*dan*	*laosheng+dan*	**total**
erhuang	20	17	1	38
xipi	24	27	3	54
total	44	44	4	**92**

Table 1. Content of JMSC, according to role type and *shengqiang*

To build JMSC, we have taken the arias used to illustrate the main metred *banshi*, namely *yuanban*, *manban*, and *kuaiban* (see Section 1.1), as they are used in the two main *shengqiang*, that is, *xipi* and *erhuang*, for the two main role types, *laosheng* and *dan*. When other related *banshi* are explained together with these, their sample arias have been also included in our collection. Those only referenced by title in [4] have been looked for in two printed scores collections that accompany our Jingju Music Corpus [1, 2]. Only eight have not been found. If different scores for the same aria occur, all versions have been included. As a result, JMSC contains 92 scores covering 80 arias. Table 1 shows the distribution of scores per role type and *shengqiang*. *Banshi* is not included here because some arias contain more than one. However, since the main melodic unit is the line, the information in Table 2 is a better representation of JMSC's potential for the study of the jingju musical system elements. All the arias have been included in its full original form, resulting in some samples of other instances of the musical system elements also being present in the collection, as shown in Table 2.

The original sources use a style of cypher notation called *jianpu* (see Figure 1.b). We transtonated them manually using MuseScore 2.1 [7] to staff notation, and exported them to MusicXML format. Since both notation systems are based in the same principles, the transnotation was straightforward. The main decision taken was the key, not given in the sources. [8] We unified all the scores in E major, a common key for the two role types considered, as stated in the reference textbooks.

	daeh	*daxp*	*lseh*	*lsxp*	**total**	*ldeh*
manban	72	50	66	17	205	
sanyan			12	17	29	2
zhongsanyan	6				6	
kuaisanyan	14		26	6	46	
yuanban	54	55	112*	47	268	
erliu		12			12	
liushui		121		78	199	
kuaiban		47		85	132	
total	146	285	216	250	**897**	
daoban		1				
sanban		2	2	3		
yaoban		1	1	8		
**kutou*			4			

Table 2. Content of JMSC per melodic line, according to role type, *shengqiang* (columns), and *banshi* (rows). On the upper heading, *da* stands for *dan*, *ls* for *laosheng*, *ld* for *laodan*, *eh* for *erhuang* and *xp* for *xipi*. Gray background indicates samples of instances not considered as focus of our research.

Jingju singing is accompanied by an instrumental ensemble in heterophony. Therefore, in order to represent the instrumental accompaniment it is customary to only notate the lead instrument in the ensemble, namely the jinghu, in an independent staff. In these cases, both the voice and the instrumental lines are notated in different staves in our transnotation. In other cases, since the voice and the instrumental ensemble conceptually play the same melody, the original *jianpu* notation represent both in one single staff with the instrumental sections in brackets, as shown in Figure 1.b. In those cases, our transnotation divides the original single staff into two, one for the voice and one for the accompaniment, as shown in Figure 1.a. Of the 92 scores, 53 (57.61 %) of them contain full accompaniment.

2.2 Coverage, Completeness and Reusability

Serra proposes in [15] five criteria for building research corpora which we followed for creating JMSC. Among them, *purpose* has been described in detail in Section 1, and we argue that the *quality* of the scores is assured insofar as our transnotation, as described in Section 2.1, maintains all the information contained in the original sources. Therefore, we discuss here the remaining criteria.

In terms of *coverage*, the collection includes the two main *shengqiang*, *xipi* and *erhuang*, and the two main role types, *laosheng* and *dan*. In order to evaluate their relevance for jingju music research, we have measured their occurrence in the recordings collection of the CompMusic Jingju Music Corpus, as published in [14]. *Xipi* and

[7] https://musescore.org/

[8] The specific tuning is chosen, within a certain range, according to the performer's needs.

erhuang stand for 80.25 % of the *shengqiang* present in the recordings collection, and *laosheng* and *dan* stand for 73.66 % of the role types. The whole range of metred *banshi* is represented in the collection, with special focus on *yuanban*, *manban*, *liushui* and *kuaiban* (see Table 2), as the most relevant ones (see Section 1.1). There are two main reasons for excluding the non-metred ones: metred *banshi* present more direct relations between them, what facilitates the study of the transformation processes. On the other hand, we are still looking for a satisfactory method to process non metred melodies. There also exist a few auxiliary *banshi*, whose occurrence is very occasional, and therefore no representative of the norm.

Considering *completeness*, the collection contains the metadata of the scores and annotations both at the score and the line level, organised in separate spreadsheets. For the scores, the metadata contain the title of the work in Chinese, role type, *shengqiang*, *banshi*, whether it contains full accompaniment (see Section 2.1), the reference of the original score, and if existing, the list of the MusicBrainz IDs of the recordings in our Jingju Music Corpus corresponding to the same work as the score. As for the lines, each of them is annotated with the role type, *shengqiang*, *banshi*, line type, that is, opening—which in the case of *erhuang* is divided in two types—or closing, the lyrics for the whole line and for each of its sections, the linguistic tones of the lyrics, and the starting and ending offsets of the line and each of its sections. Annotations have been done by the first author, linguistic tones have been extracted automatically [9] and corrected manually by the first author and by a Chinese native speaker knowledgeable about jingju, and specific doubts about line segmentation have been solved by two professors in NACTA.

Regarding *reusability*, all the scores, including metadata and annotations, are available for research purposes. Due to copyright issues, they are only available on demand, by contacting the authors.

2.3 Integration in the Jingju Music Corpus

JMSC was gathered as part of the Jingju Music Corpus from the CompMusic project, also created with the purpose of studying jingju singing in terms of its musical system elements. To exclude the influence of academic compositional techniques applied to jingju during the 20th century, only plays from the traditional repertoire were considered for the corpus, including JMSC. Consequently, two samples of revolutionary plays from [5] were discarded.

The scores are integrated in the corpus via the work they represent. If a particular recording in the corpus contains a performance of a work to which a particular score is related, due to the circumstances described in Section 1.2, it can not be assumed that the recording contains a performance of the score. Consequently, scores and recordings are indirectly related through works. Taking this into account, 63 of the 92 scores (68.48 %) are related to works associated to one or more audio recordings.

[9] The software used for extracting the linguistic tones can be obtained here: https://github.com/zangsir/Native-to-Pinyin

2.4 Potential of the Collection

JMSC has great potential for the computational research of jingju singing in terms of its musical system elements. In the following section we present statistical analyses with that aim. However, it is also suitable for other research tasks. Since the variety of melodic material is conceived as transformations of original tunes, these scores offer an excellent opportunity for pattern discovery and similarity analysis. These tasks can benefit from the accompanying annotations to develop culture specific heuristics. The annotated linguistic tones for the lyrics allow the research of their relationship with melody using symbolic data, whilst such studies have been carried out so far using only audio [22, 23]. Since the scores contain full or partial notation of the accompaniment, they are a good resource for the analysis of the relationship between the singing and instrumental lines. Finally, although scores and recordings are not directly related, the similarity between those which share a common work still allows combined analyses.

3. ANALYSES PERFORMED ON THE JINGJU MUSIC SCORES COLLECTION

In order to take advantage of JMSC, we have extracted statistical information with the aim of testing musicological claims made in the reviewed literature [3–6, 20]. To process the scores we have used the `music21` toolkit [9], and the developed code is openly available. [10] We introduce now the four types of features considered for analysis, together with their musicological motivations:

Pitch histograms. Jingju music is based on an anhemitonic pentatonic scale, each of whose degrees can be the finalis of a mode (*diao*), a defining characteristic of each *shengqiang*. Hence, *xipi* is associated to the *gong* mode, which has the first degree as finalis, and *erhuang* to the *shang* mode, whose finalis is the second degree. The 4th and 7th degrees, usually omitted, can be used as expressive notes. Male and female styles of singing, represented by the *laosheng* and *dan* role types are defined by different pitch registers. Pitch histograms can help to gain a deeper understanding of pitch distributions for each *shengqiang* and pitch register for each role type, as well as to evaluate the role of the expressive notes.

Interval histograms. One of the melodic features given in the literature for distinguishing *xipi* from *erhuang* is that it uses larger intervals. Analysing intervals through histograms will shed light upon this claim.

Cadential notes. One of the more common characteristics given in the literature when comparing *xipi* and *erhuang*, and closely related to their modal associations, is a schema of cadential notes (*laoyin*) for each line of the couplet and each of their three sections.

Melodic density. Understood as the proportion of notes per syllable, this feature is used for characterizing *banshi*, arguing that the slower the *banshi* is, the more notes are used for singing each syllable.

[10] https://github.com/MTG/Jingju-Scores-Analysis/releases/tag/v1.0

Our code computes the aforementioned features for any combination of role type, *shengqiang*, *banshi* and line type. Histograms show blue bars for *laosheng* and orange ones for *dan*, whilst *shengqiang* is indicated by the hatch, \ for *xipi* and / for *erhuang* (see Tables 3, 4, and 5). A solid red line marks the first degree (E4) in pitch histograms, and a dashed one its higher octave (see Tables 3, and 4). Interval analysis can consider interval direction or not. Cadential notes are computed for each section of the line, distinguishing opening and closing lines (see Table 6). Melodic density box plots show results for individual scores and for the average of all of them (see Table 7).

For this paper we computed the four aforementioned features for the whole collection considering all the combinations of role type, *shengqiang*, and *banshi*. For a first approach to the results, we grouped the *banshi* in three groups: *yuanban* and *erliu*, in 2/4 metre and medium tempo ranges; *manban*, *sanyan*, *zhongsanyan* and *kuaisanyan*, in 4/4 metre and slower tempo ranges than the previous group; and *kuaiban* and *liushui*, in 1/4 metre and faster tempo ranges than the first group. To ease readability, each group is referred to in the next section by the first *banshi* of the group. All the resulting plots are available. [10]

4. DISCUSSION OF THE RESULTS

From the results of the aforementioned analyses we have obtained the following musicological findings. Pitch histograms in Table 3 show that the role types *laosheng* and *dan* are well defined in terms of pitch range. The predominance of the 5th and 6th degrees for *dan* (B4 and C#4) is explained in the literature as a transposition of the modal center a fifth higher. In terms of pitch distribution, it can be observed that the finalis of each of the modes associated to each *shengqiang*, namely 1st (E4) for *xipi* and 2nd (F#4) for *erhuang*, or their transpositions a fifth higher for *dan*, are in fact the most predominant pitches, but not very far from other degrees. The exception is *laosheng erhuang*, whose most predominant pitch is the lower 6th degree (C#3), and that can be characterised by a relatively major importance

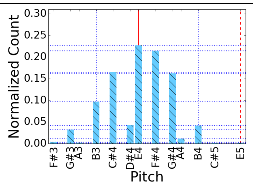
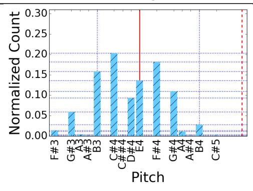
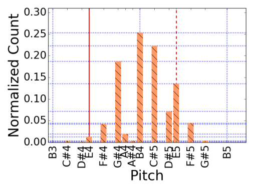
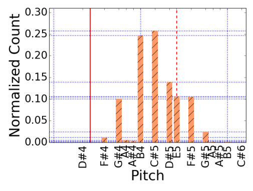
	xipi	*erhuang*
laosheng		
dan		

Table 3. Pitch histograms for the role types *laosheng* and *dan*, and the *shengqiang xipi* and *erhuang*. All *banshi* included

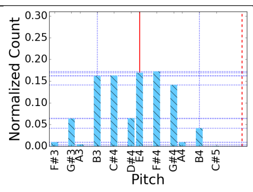
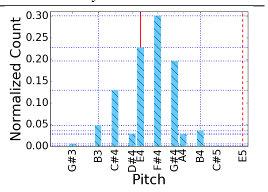
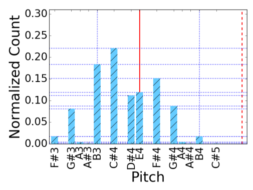
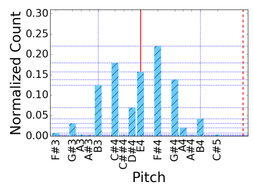
	manban	*yuanban*
laosheng xipi		
laosheng erhuang		

Table 4. Pitch histograms for the role type *laosheng*, comparing the *banshi* groups *manban* and *yuanban* for the *shengqiang xipi* and *erhuang*

of the lower region of its register. The use of the 7th degree (D#4) is also remarkable, specially prominent for *dan*, what can be explained by the transportation of the modal center, resulting in that this pitch acts as a 3rd degree. The 4th degree (A4) appears, but very rarely, and also its sharpened version (A#4). The appearance of these two versions could be due to the fact that its absolute tuning differs from the equal temperament [8], and transcribers use different approaches to notate this pitch.

We have also found that pitch histograms are useful for characterizing different *banshi*. In Table 4 it can be observed how slow *banshi* (*manban* group) explore lower pitch regions than the ones closer to the original tune (*yuanban* group). That is especially relevant in *xipi*. It can also be observed how the former make a more frequent use of the expressive tone D#4, the 7th degree.

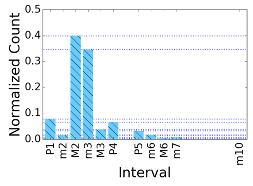
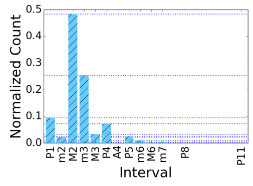
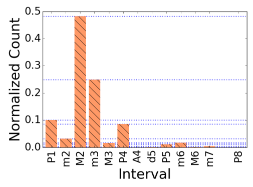
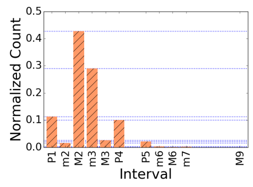
	xipi manban	*erhuang manban*
laosheng		
dan		

Table 5. Interval histogram for the role types *laosheng* and *dan* and the *shengqiang xipi* and *erhuang*, considering only the *banshi* group *manban*

Our findings on interval distributions support the musicological claims stated previously, but also help to observe some nuances. Table 5 shows that, in the case of *laosheng*, diatonic steps are more frequent than minor third

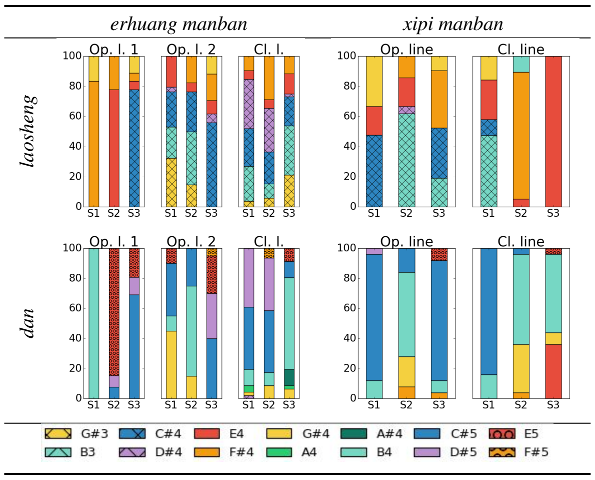


Table 6. Cadential notes in the *banshi* group *manban* for the role types *laosheng* and *dan* and the *shengqiang erhuang* and *xipi*, for opening (Op.) and closing (Cl.) lines in each of their three sections (S1, S2, and S3)

steps in *erhuang* than in *xipi*. However, in the case of *dan*, the results do not support the claim. In all cases, it can be observed that intervals larger than the minor 3rd are rare in both *shengqiang*, with the exception of the perfect 4th, whose distribution is similar in the four cases. Consequently, the major use of large intervals in *xipi* as compared with *erhuang* can be nuanced in light of these plots.

It is an agreement in jingju music literature to define *laosheng xipi* as cadencing in the 2nd degree for the opening line of the couplet, and in the 1st degree for the closing line, and *erhuang* as the opposite, 1st degree for the opening line and 2nd degree for the closing line. In the case of *dan*, cadences will be 6th and 5th degrees for opening and closing lines in *xipi*, and higher octave 1st and 5th degrees for *erhuang*. Different sources refer with different degrees of precision to cadential notes for each section of the line and to exceptions to these general rules. Focusing now only in line cadential notes, that is, those for section 3 (S3), Table 6 shows that only closing lines in *laosheng xipi manban* completely match the rules given previously, presenting the 1st degree (E4) as cadential note in all cases. As for *laosheng erhuang manban*, the cadential notes established in theory occupy a very small percentage, especially noticeable in opening line 1. That is also the case for *dan erhuang*, which presents a more varied range of possibilities for cadential notes than *dan xipi*, and where those established in the literature stand only for a minimum percentage in opening lines.

Finally, Table 7 shows how different *banshi* groups can be characterized in terms of melodic density. The duration unit for measuring a syllable length is the crotchet. Two aspects are interesting in these plots. The median for each group shows, as expected, that sung syllables are longer as the tempo decreases, and compared with *laosheng*, the *dan* role type shows slightly higher median values in *yuanban*. But the outliers also provide meaningful information. These correspond to a singing feature known as *tuoqiang*,

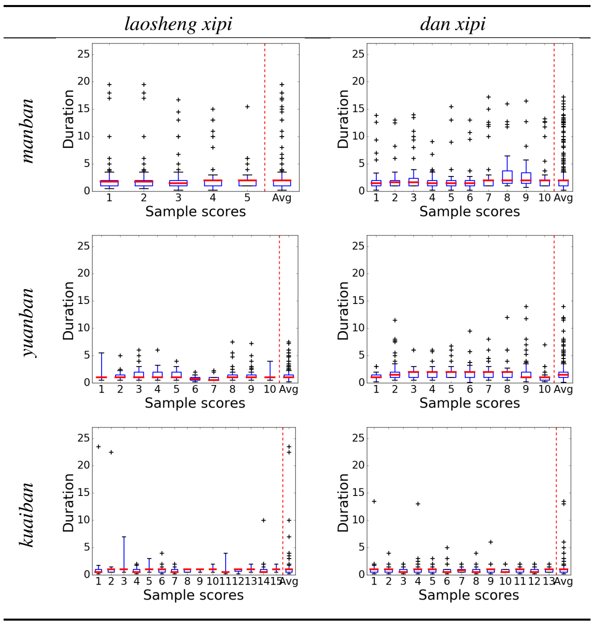


Table 7. Melodic density for the *banshi* groups *manban*, *yuanban*, and *kuaiban* and the role types *laosheng* and *dan* in the *shengqiang xipi*

literally "dragged tune", by which the melody of a syllable, usually at the end of a line or a line section, is extended by a long melisma. Table 7 shows how much more frequently this phenomenon occurs in *manban* than in *yuanban*, and how it is almost non-existing in *kuaiban*. However, when it appears in this *banshi*, it can be longer than in *yuanban*.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented the first collection of machine readable scores for the study of jingju singing in terms of its musical system elements. It includes 92 scores covering 897 melodic lines, and is accompanied by their metadata and curated annotations per score and melodic line. The collection is part of the corpus gathered in the CompMusic project for jingju music research. Its potential for jingju singing analysis has been tested by a series of statistical analyses, and some musicological findings have been discussed. The whole collection, together with the metadata, the annotations, and the developed code are available for research purposes.

In future work, it is expected to expand the collection by including other instances of the musical system elements that are not currently present, especially non metred *banshi*. At the same time, the collection's potential, as pointed out in Section 2.4, will be exploited for different research tasks. Among them, pattern discovery is a promising topic, since the accompanying structural annotations can be used to design heuristics to incorporate to state of the art approaches, as well as the analysis of the relationship between linguistic tones and melody. Most importantly, we hope that JMSC contributes to open up a new range of possibilities for MIR research on jingju music.

6. ACKNOWLEDGEMENTS

The authors would like to thank Mr. Shuo Zhang for automatically extracting the linguistic tones of the scores' lyrics, Prof. Shen Pengfei and Prof. Xiong Luxia from NACTA for helping to solve some questions about line segmentation, and Mr. Rong Gong for reviewing and correcting the annotations of linguistic tones. This research is funded by the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013), as part of the CompMusic project (ERC grant agreement 267583).

7. REFERENCES

[1] *Jingju qupu jicheng* 京剧曲谱集成 (Collection of jingju scores). Shanghai wenyi chubanshe, Shanghai, 1998. 10 Vols.

[2] *Zhongguo jingju liupai jumu jicheng* 中国京剧流派剧目集成 (Collection of plays of Chinese jingju schools). Xueyuan chubanshe, Beijing, 2006–2010. 21 Vols.

[3] Liu J. 刘吉典. *Jingju yinyue gailun* 京剧音乐分析 (Introduction to jingju music). Renmin yinyue chubanshe, Beijing, 1992.

[4] Zhang Z. 张正治. *Jingju chuantongxi pihuang changqiang jiegou fenxi* 京剧传统戏皮黄唱腔结构分析 (Structural analysis of *pihuang* singing in jingju traditional plays). Renmin yinyue chubanshe, Beijing, 1992.

[5] Cao B. 曹宝荣, editor. *Jingju changqiang banshi jiedu* 京剧唱腔板式解读 (Deciphering banshi in jingju singing). Renmin yinyue chubanshe, Beijing, 2010.

[6] Jiang J. 蒋菁. *Zhongguo xiqu yinyue* 中国戏曲音乐 (Music of Chinese traditional opera). Renmin yinyue chubanshe, Beijing, 2000.

[7] D. A. A. Black, M. Li, and M. Tian. Automatic identification of emotional cues in Chinese opera singing. In *13th ICMPC*, pages 250–255, Seoul, Korea, 2014.

[8] K. Chen. Characterization of pitch intonation of Beijing opera. Master's thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2013.

[9] Michael Scott Cuthbert and Christopher Ariza. `music21`: A toolkit for computer-aided musicology and symbolic music data. In *11th ISMIR*, pages 637–642, Utrecht, Netherlands, 2010.

[10] G. Dzhambazov, Y. Yang, R. Caro Repetto, and X. Serra. Automatic alignment of long syllables in a cappella Beijing opera. In *6th FMA*, pages 88–91, Dublin, Ireland, 2016.

[11] R. Gong, N. Obin, G. Dzhambazov, and X. Serra. Score-informed syllable segmentation for jingju a cappella singing voice with mel-frequency intensity profiles. In *7th FMA*, pages 107–113, Málaga, Spain, 2017.

[12] R. Gong, Y. Yang, and X. Serra. Pitch contour segmentation for computer-aided jingju singing training. In *13th SMC*, pages 172–178, Hamburg, Germany, 2016.

[13] R. Caro Repetto, R. Gong, N. Kroher, and X. Serra. Comparison of the singing style of two jingju schools. In *16th ISMIR*, pages 507–513, Málaga, Spain, 2015.

[14] R. Caro Repetto and X. Serra. Creating a corpus of jingju (Beijing opera) music and possibilities for melodic analysis. In *15th ISMIR*, pages 313–318, Taipei, Taiwan, 2014.

[15] X. Serra. Creating research corpora for the computational study of music: the case of the compmusic project. In *AES 53rd International Conference on Semantic Audio*, pages 1–9, London, UK, 2014.

[16] A. Srinivasamurthy, R. Caro Repetto, H. Sundar, and X. Serra. Transcription and recognition of syllable based percussion patterns: The case of Beijing opera. In *15th ISMIR*, pages 431–436, Taipei, Taiwan, 2014.

[17] J. Sundberg, L. Gu, Q. Huang, and P. Huang. Acoustical study of classical Peking opera. *Journal of Voice*, 26(2):137–143, 2012.

[18] M. Tian and M. B. Sandler. Towards music structural segmentation across genres: Features, structural hypotheses, and annotation principles. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):23–41, 2017.

[19] M. Tian, A. Srinivasamurthy, M. Sandler, and X. Serra. A study of instrument-wise onset detection in Beijing opera percussion ensembles. In *ICASSP 2014*, pages 2174–2178, Florence, Italy, 2014.

[20] E. Wichmann. *Listening to theatre: The aural dimension of Beijing opera*. University of Hawaii Press, Honolulu, 1991.

[21] L. Yang, M. Tian, and E. Chew. Vibrato characteristics and frequency histogram envelopes in Beijing opera singing. In *5th FMA*, pages 139–140, Paris, France, 2015.

[22] S. Zhang, R. Caro Repetto, and X. Serra. Study of the similarity between linguistic tones and melodic pitch contours in Beijing opera singing. In *15th ISMIR*, pages 343–348, Taipei, Taiwan, 2014.

[23] S. Zhang, R. Caro Repetto, and X. Serra. Predicting pairwise pitch contour relations based on linguistic tone information in Beijing opera singing. In *16th ISMIR*, pages 107–113, Málaga, Spain, 2015.

Oral Session 2

Multitrack

MONAURAL SCORE-INFORMED SOURCE SEPARATION FOR CLASSICAL MUSIC USING CONVOLUTIONAL NEURAL NETWORKS

Marius Miron, Jordi Janer, Emilia Gómez
Music Technology Group, Universitat Pompeu Fabra, Barcelona
`firstname.lastname@upf.edu`

ABSTRACT

Score information has been shown to improve music source separation when included into non-negative matrix factorization (NMF) frameworks. Recently, deep learning approaches have outperformed NMF methods in terms of separation quality and processing time, and there is scope to extend them with score information. In this paper, we propose a score-informed separation system for classical music that is based on deep learning. We propose a method to derive training features from audio files and the corresponding coarsely aligned scores for a set of classical music pieces. Additionally, we introduce a convolutional neural network architecture (CNN) with the goal of estimating time-frequency masks for source separation. Our system is trained with synthetic renditions derived from the original scores and can be used to separate real-life performances based on the same scores, provided a coarse audio-to-score alignment. The proposed system achieves better performance (SDR and SIR) and is less computationally intensive than a score-informed NMF system on a dataset comprising Bach chorales.

1. INTRODUCTION

As a special case of audio source separation, music source separation has gained significant attention during the past years. Recovering the sources corresponding to the instruments from an audio mixture allows for interesting applications such as music upmixing [9] or virtual-reality concerts [16], and it is useful in music information retrieval tasks [11,30].

In contrast to speech separation, music source separation poses different challenges due to the variety of sources which are correlated in time and frequency [7]. Because of the multitude of harmonic instruments, often related timbres, variations in dynamics, Western classical music is a challenging case [21]. On the other hand, results can improve if prior knowledge about the nature of sources [5, 29] and their timbre [2] informs the separation framework. Considerable improvements are obtained in the case of parametric models, such as NMF, which are restricted using coarsely aligned scores [4, 7, 10, 14].

Recently, neural network approaches have outperformed NMF in audio source separation challenges [18]. Deep learning systems estimate soft masks for specific instrument classes [3, 13, 15] or computing the instrument spectra directly [27]. In contrast to NMF methods, a deep learning framework is less computationally expensive [3] at the separation stage, as estimating the sources involves a single feed forward pass through the network rather than an iterative procedure. Thus, it can be used in a low latency scenario. Furthermore, recurrent [15] and convolutional [3, 13] networks have the advantage of modeling a larger time context.

Novel deep learning source separation systems propose specialized models which propose building an NMF logic into an autoenconder [26] or cluster components over large time spans [19]. Including score information into the deep learning separation frameworks can yield further improvements [8].

In this paper we introduce a monaural score-informed source separation framework for Western classical music using convolutional neural networks (CNN). We assume that for a given classical music piece the instruments are known and the score is available. Thus, for a set of given scores we generate renditions which are used to train a CNN. The trained model is used to separate real-life performances based on these scores [22].

A global alignment of the score with the audio of a performance can be obtained by a score following system [4]. Then, the resulting coarsely aligned score, with errors up to 0.2 seconds, is used to derive score-based soft masks for each of the sources. From these masks we generate score-filtered spectrograms as input features for the CNN.

Training neural networks for source separation requires isolated audio tracks which are difficult to obtain. Therefore, we use the data generation method in [22]. Accordingly, we synthesize renditions of original scores with variations in timbre, dynamics and local timing deviations.

The remainder of the paper is structured as follows. In Section 2 we state our contributions in relation with the previous work. In Section 3 we introduce the proposed method including the feature computation, the architecture of the network and the training procedure. In Section 4 we discuss the evaluation of the proposed method. We present our conclusions in Section 5.

2. RELATION TO PREVIOUS WORK

Score-informed constraints [7,10] are imposed to the NMF framework through restricting the activations of the note templates. In a similar manner, we use the score to generate sparse training features which are used as input to the CNN. Furthermore, since score following errors influence the quality of separation [4, 20], we compensate for local misalignments in a similar manner to [7, 10], by allowing a tolerance window around note onsets and offsets while computing our training features. For our experiments and the dataset we used the size of this window is 0.2 seconds.

Deep learning systems can become more robust to real-life cases by increasing the size and variability of the training dataset through data generation [22] or augmentation [24, 25]. In this sense, the difference in performance between two similar deep learning methods can be largely explained by the difference between training datasets rather than new features or methods [1]. We are motivated by recent advance in deep learning which go beyond the black-box model and try to integrate musically meaningful features [19,26]. Thus, we aim at improving source separation for classical music with a context-driven method which includes score information.

The CNN architecture in this paper is adapted from the convolutional autoencoder proposed in [3, 22]. In comparison to [3] our CNN architecture has different filter and layer sizes. Moreover, the original scores from which training data is generated are further used to derive score-informed features which are given as input to the CNN in a representation analogous to multi-channel images. To that extent, our approach contrasts with [8] which uses score restrictions inside the deep learning framework. Furthermore, to our best knowledge, deep learning audio processing methods do not use a multi-channel input as in image processing applications. Thus, we analyze whether the convolutional autoencoder introduced in [3,22] learns a better representation from a multi-channel input than from a single channel input, given that the feature maps are shared between all channels. In addition, we use bootstrapping with replacement to train such an architecture when working with big datasets.

3. METHOD

The diagram of the separation framework with the two stages, training and separation, can be seen in Figure 1. For the training stage, we start from the original scores from which we derive synthetic audio renditions with the method in [22]. The same scores are used to derive features for training the CNN in form of score-based soft masks, explained in Section 3.1.1, and score-filtered spectrograms, explained in Section 3.1.2. For the separation stage, our framework takes as input an audio mixture and the corresponding coarsely aligned score. Similar to the training stage, we compute the score-based soft masks and the score-filtered spectrograms which are feed-forwarded through the CNN model to obtain the magnitude spectrograms of the separated sources.

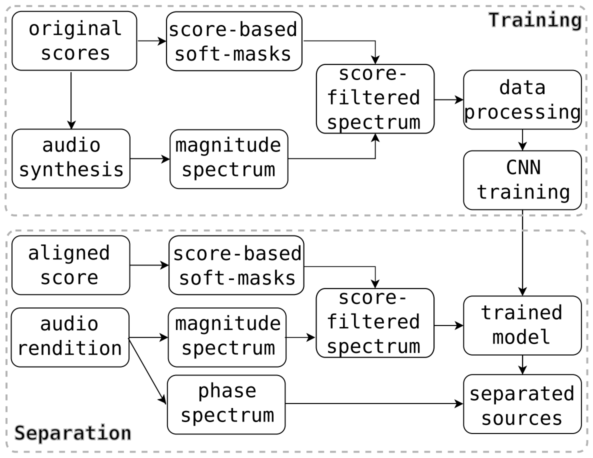


Figure 1. The overview of the separation system comprising the two stages: training and separation

3.1 Feature computation

The goal of computing score-based soft masks is to derive additional sparse score-filtered spectrograms which are used as an input to the CNN.

3.1.1 Score-based soft masks

A score gives the note onsets and offsets time and the MIDI note numbers. Assuming that the source is harmonic and we know the tuning frequency, f_q, the MIDI note associated with A4, m_{A4}, we can compute the fundamental frequency $f_0 = f_q \cdot 2^{\frac{1}{12}\cdot(m-m_{A4})}$, where m is the MIDI note number.

Score information yields the time-frequency zones where the notes are played. Correspondingly, for a given note n that plays between the time frames t_b and t_e we can define the time range as:

$$\boldsymbol{U}_n(t) = u(t - t_b - t_w) - u(t - t_e - t_w) \qquad (1)$$

where u is the unit step function, and t_w is tolerance window set around onset t_b and offset t_e which compensates for local misalignments in score-following, similarly to [4,7,10,14]. The tolerance window is applied at training and separation.

Furthermore, if we consider the fundamental frequency f_0 of the note n we define the frequency range as:

$$\boldsymbol{V}_n(f) = \sum_{h=1}^{H} u(f - hf_0/f_i) - u(f - hf_0 f_i) \qquad (2)$$

where $h = 1 : H$ are the harmonic partials, and $f_i = 2^{f_c/1200}$ is the allowed frequency interval below and above each harmonic partials, with f_c being the allowed interval in cents, and 1200 is the number of cents per octave.

For each source $j = 1 : J$ and all its notes $n = 1 : N_j$ we can compute score-based binary matrices $\boldsymbol{K_j}(t, f)$ as a sum of outer products:

$$\boldsymbol{K_j}(t, f) = \sum_{n=1}^{N_j} \boldsymbol{U}_n(t) \otimes \boldsymbol{V}_n(f) \qquad (3)$$

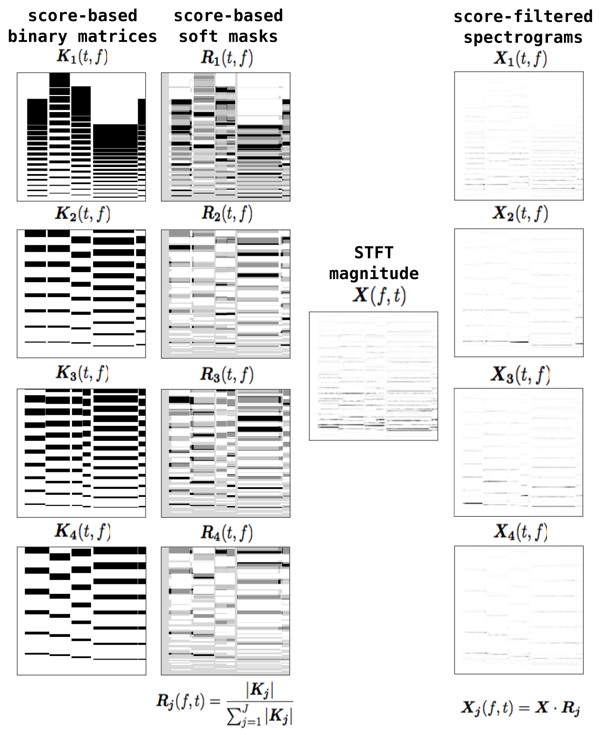


Figure 2. Feature computation for the first 4 seconds and frequencies between 0-6500Hz, for the piece *Ach Gottund Herr* of Bach10 dataset [4] comprising four instruments.

An example of $\boldsymbol{K_j}$ for a classical music piece comprising four harmonic sources is shown in the first column of Figure 2.

The score-based soft masks for each source, $j = 1 : J$, are given by the equation:

$$\boldsymbol{R_j}(f,t) = \frac{|\boldsymbol{K_j}|}{\sum_{j=1}^{J} |\boldsymbol{K_j}| + \epsilon} \tag{4}$$

where $\epsilon = 1^{-10}$ is a constant to handle division by zero. We illustrate a set of $\boldsymbol{R_j}$ matrices in the second column of Figure 2.

In this paper we consider solely combinations between harmonic sources, which are reflected in the initialization of $\boldsymbol{V}_n$ using a series of harmonic partials, as seen in Equation 2. However, the proposed solution can be easily extended to model non-harmonic sources by initializing the vector $\boldsymbol{V}_n(f) = 1$ along all the frequency range, resulting in a less sparse score-filtered spectrogram which is solely informed by onsets and offsets times through $\boldsymbol{U}_n(t)$.

3.1.2 Score-filtered spectrograms

We calculate the STFT magnitude spectrogram of the audio mixture as $\boldsymbol{X}(f,t)$. Then, we derive score-filtered spectrograms for each of the sources $j = 1 : J$, by computing the element-wise product between the spectrogram of the mixture, $\boldsymbol{X}$, and the score-based soft masks, $\boldsymbol{R_j}$:

$$\boldsymbol{X_j}(f,t) = \boldsymbol{X} \cdot \boldsymbol{R_j} \tag{5}$$

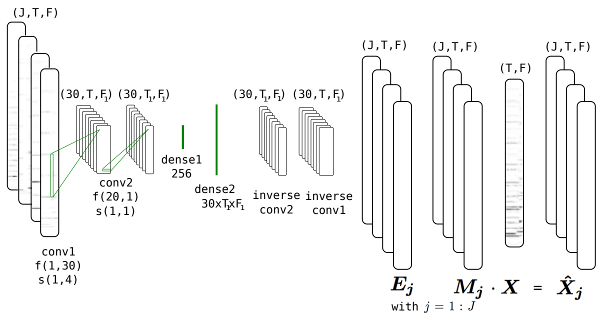


Figure 3. The CNN architecture used in the separation framework for $J = 4$ sources

3.2 Convolutional Neural Network architecture

The convolutional autoencoder architecture can be seen in Figure 3. It comprises a convolution stage with two convolution layers, two dense layers, and a deconvolution stage. The sources are reconstructed using the filters learned at the convolution stage. In addition, we have two deterministic layers to compute the spectrograms of the sources.

In contrast to the CNN architectures in [3, 13, 22], our CNN takes as input J score-filtered spectrograms for a time context T and a number of frequency bins F, rather than a single spectrogram of the mixture. The J score-filtered spectrograms share the same feature maps, in a similar way to image processing deep learning methods that use color RGB channels [1]. Our assumption is that this additional information can better guide the separation between the sources. Furthermore, as shown in Figure 2, the score-filtered spectrograms are sparser versions of the original spectrogram, offering a better representation for source separation [23].

The first layer *conv1* is a convolution layer with filter shape $(1, 30)$, hence the convolution only happens in frequency. For this layer we have a stride[1] of $(1, 4)$, which reduces dimensionality by keeping into account the sparsity of the input.

This layer outputs feature maps of size $(30, T, F_1)$, where $F_1 = (F - 30)/4 + 1$, where 30 is the length of the filter and 4 is the stride. The second layer *conv2* is a convolution layer with filter shape $(20, 1)$, which learns temporal patterns. The output of this layer has the size $(30, T_1, F_1)$, where $T_1 = (T - 20) + 1$ with 20 being the length of the filter. This layer has a stride of $(1, 1)$, since we are interested in maintaining a good temporal resolution at the reconstruction. Note that the convolution layers have a linear activation function.

We use a dense bottleneck layer as in [3] with 256 units and a rectifier linear unit activation function [1], denoted as *dense1*. The limited number of units and the activation function have been proven to better guide the parameter learning and prevent overfitting in the case of timbre-informed source separation [3].

To match dimensions necessary for the deconvolution $(30, T_1, F_1)$ for each of the J sources, we introduce a layer

[1] The stride controls how much a filter is shifted on the input.

dense2 comprising J dense layers of shape $30 \cdot T_1 \cdot F_1$. For each of these J layers we perform the inverse operations of *conv2* and *conv1* and we obtain a set of estimations $\boldsymbol{E_j}$ for each of the separated sources $j = 1 : J$.

Following [3, 15], we integrate the computation of the soft masks into the architecture of the network as an additional deterministic layer. Thus, the soft masks $\boldsymbol{M_j}$, for each source $j = 1 : J$, are computed from the output of the previous layer, $\boldsymbol{E_j}$, as:

$$\boldsymbol{M_j} = \frac{|\boldsymbol{E_j}|}{\sum_{j=1}^{J} |\boldsymbol{E_j}| + \epsilon} \tag{6}$$

where $\epsilon = 1^{-10}$ is a constant to handle division by zero. The magnitude spectrogram corresponding to the sources, $\hat{\boldsymbol{X}}_{\boldsymbol{j}}$, are given by the element-wise multiplication between input spectrogram and the soft-masks $\hat{\boldsymbol{X}}_{\boldsymbol{j}} = \boldsymbol{M_j} \cdot \boldsymbol{X}$. The soft masks $\boldsymbol{M_j}$ are not to be confounded with the score-based soft-masks $\boldsymbol{R_j}$ introduced in Section 3.1.1 and used to derive input features for the CNN.

3.3 Training procedure

The network is trained according to the mean-squared error between the magnitude spectrograms of the target sources, $\hat{\boldsymbol{X}}_{\boldsymbol{j}}$, and the magnitude spectrograms of the sources yielded by the network, $\boldsymbol{X_j}$, as: $Loss = \sum_{j=1}^{J} \|\hat{\boldsymbol{X}}_{\boldsymbol{j}} - \boldsymbol{X_j}\|^2$.

The parameters of the CNN are updated using mini-batch Stochastic Gradient Descent with the *AdaDelta* algorithm [31].

With the method in [22] we can generate a high number of renditions, covering a high number of possibilities, which makes the framework more robust to real-life data. However, training on big datasets is an expensive procedure and we experimented with a faster training method summarized in the Algorithm 1. In this case, we sample a limited number data points before each epoch rather than having a fixed dataset at the beginning of training. In statistics, this procedure is known as bootstrapping with replacement [17]. Note that, for this training procedure, the concept of *epoch* (a single pass through the entire training set) does not hold anymore.

Algorithm 1 *Bootstrapping with replacement*

```
1 repeat
2     randomly sample a number of data points from the dataset
3     for each training batch do
4         compute weights and bias gradients for the current
          batch
5         accumulate the gradients
6     end for
7     adjust weights and bias using accumulated gradients
8 until total number of stages is reached
```

3.4 Separated source estimation

We assume that the individual sources $y_j(t), j = 1 : J$, that compose the audio mixture $x(t)$ are linearly mixed, so that $x(t) = \sum_{j=1}^{J} y_j(t)$. Therefore, from the estimated magnitude spectrograms $\boldsymbol{X_j}$ and using the original phase of the audio mixture we can obtain the signals associated to the sources, $y_j(t)$, with an inverse overlap-add STFT [10].

The neural network yields estimations of shape (T, F) for each of the J sources. Considering an audio mixture of variable time shape, the estimation is done for overlapping segments of shape (T, F), with the algorithm described in [22].

4. EVALUATION

4.1 Datasets

For evaluation purposes we use ten Bach chorales from the Bach10 dataset [4], played by bassoon, clarinet, tenor saxophone, and violin. The mean duration of a piece is ≈ 30 seconds. In addition, each piece is accompanied by the score aligned with the audio, the original score, and an automatic alignment obtained with the algorithm in [4]. This dataset has been widely used in tasks as source separation, alignment, and transcription.

4.2 Generating training data

We generate training data with the method in [22] which uses sample-based synthesis with samples from the RWC instrument sound database [12]. The method synthesizes original scores at different tempos, dynamics, considering local timing deviations, and using different timbres to generate a wide variety of renditions of given pieces.

In this case, we have three different timbres and three level of dynamics. In addition, to account for local timing variations, we circular-shift the audio with $s = \{0, 0.1, 0.2\}$ seconds. An analogous transformation needs to be applied to the associated score by adding s seconds to the note onsets and offsets.

Considering the variations of the factors above ($3 \cdot 3 \cdot 3 = 27$) for the four instruments, we can generate a total number of $27^4 = 531441$ renditions for a single piece. Because it is not feasible to generate such a high number of audio files, we randomly choose 400 renditions to build our training dataset. Samples are uniformly distributed across the dataset. Since we are training a CNN model for all the 10 pieces in Bach10 dataset, we have a total number of 4000 renditions.

4.3 Evaluation setup

We used the evaluation framework and the metrics described in [28] and [6] : *Source to Distortion Ratio* (SDR), *Source to Interference Ratio* (SIR), and *Source to Artifacts Ratio* (SAR).

The STFT is computed using a Blackman-Harris window of length 4096 samples, which at a sampling rate of 44.1 KHz corresponds to 93 milliseconds (ms), and a hop size of 512 samples (11ms).

When computing the soft-masks from the score, as described in Section 3.1.1, we consider the tuning frequency,

$f_q = 440$Hz, the MIDI note associated with A4, $m_{A4} = 69$, and we allow $f_c = 40$ cents above and below each harmonic partials to account for vibrato. Additionally, because we want to train our score-informed system to account for errors in score following, we set the tolerance window to be $t_w = 0.2$ seconds around onsets and offsets.

The time context modeled by the CNN is $T = 30$ frames. Furthermore, a more robust system is achieved by taking consecutive T-sized frames with an overlap of 25 frames with the algorithm described in [22].

The number of epochs is variable for each training experiment. The size of a mini-batch is set to 32.

This paper follows the principles of research reproducibility [2]. The code used in this paper is made available online [3]. It is built on top of Lasagne, a framework for neural networks using Theano [4]. We ran the experiments on a Ubuntu 16.04 PC with GeForce GTX TITAN X GPU, Intel Core i7-5820K 3.3GHz 6-Core Processor, X99 gaming 5 x99 ATX DDR44 motherboard.

4.4 Experiments

In a first experiment, we compare the proposed framework with an NMF counterpart on the Bach10 dataset. We train our CNN framework on the synthetic dataset we described in Section 4.2 (10×400 renditions) and the corresponding scores. Because we want the model to learn to deal with errors in alignment we set a tolerance window around notes' onsets and offsets. Then, we test the resulting model on real-life performances in Bach10 dataset and the scores yielded by the score-following system in [4].

Because we want to isolate the influence of the score-following system, we test our system on the score perfectly aligned (PA) with the audio. For this case, denoted as CNN PA, the tolerance window is not needed, neither for training nor testing. Furthermore, to assess the influence of the proposed features, we train the CNN architecture without any score information, having as input the magnitude spectrogram of the mixture, similarly to the system in [22]. We denote this experiment as CNN T.

We compare our score-informed system to a state of the art NMF counterpart [20]. The note templates are trained on the RWC dataset and are kept fixed during the factorization. Score-information is introduced through the activation matrix by setting to zero the activations corresponding to notes which are not played. The activations which are set to zero will remain this way during factorization, allowing the energy to be distributed between the active templates.

For the NMF system we use as input the score aligned with [4] with a tolerance window of 0.2 seconds, and the perfectly aligned score, as two separate cases, denoted as NMF and NMF PA. Furthermore, for the NMF we kept the default parameters presented in the paper [20]: 50 iterations for the factorization, beta-divergence distortion $\beta = 1.3$, STFT window size $93ms$, and hop size $11ms$.

For this first experiment we do not test the bootstrapping with replacement procedure. To that extent, we train the CNN with all the 4000 renditions for a maximum number of 20 epochs and we stop training if the loss between two epochs drops below 0.2.

In a second experiment, we test the effectiveness of the training procedure based on bootstrapping with replacement, described in Algorithm 1 and compare it with the standard training procedure which maintains the same data points during training. Furthermore, since we want to determine the optimal value for the number of renditions used at each epoch or stage, we train the CNN successively with the two procedures using different numbers of renditions. For this experiment we train for a number of 50 epochs or stages.

4.5 Results

The SDR, SIR, and SAR for our system (CNN and CNN PA), the timbre informed version CNN T, and the NMF framework are presented in the Figure 4. Error bars are drawn for a confidence interval of 95%.

We observe that the proposed score-informed framework performs better than NMF when working with coarsely aligned scores: 6dB vs 5dB in SDR. Hence, with our framework we are able to compensate for local misalignment errors around 0.2 seconds. This results in less interference, since the CNN method has $2dB$ more in SIR than the NMF, and can be due to the fact that the CNN models temporal patterns in the *conv2* layer and to the non-linearities in the bottleneck *dense1* layer.

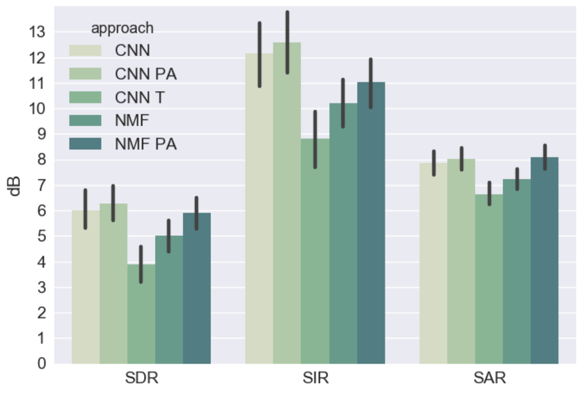


Figure 4. Results in terms of SDR, SIR, SAR for the proposed CNN framework and the NMF framework [20]

Having score-filtered spectrograms as input (CNN) improved $2dB$ in SDR in comparison to giving the magnitude spectrograms as input (CNN T), which proves the effectiveness of the features derived from score.

When the score is perfectly aligned with the audio, there is no significant difference in SDR between the CNN PA and NMF PA. However, the proposed method has $1dB$ higher SIR and similar SAR values to the NMF PA. Note that CNN PA is trained on the original scores and it is not

[2] http://soundsoftware.ac.uk/resources/
[3] https://github.com/MTG/DeepConvSep
[4] http://lasagne.readthedocs.io/en/latest/Lasagne and http://deeplearning.net/software/theano/Theano

targeted for special case. To that extent, as the CNN and CNN PA achieve similar results, we believe that having a perfect alignment does not improve results for this particular type of CNN architecture. This is in line with the results obtained in [22].

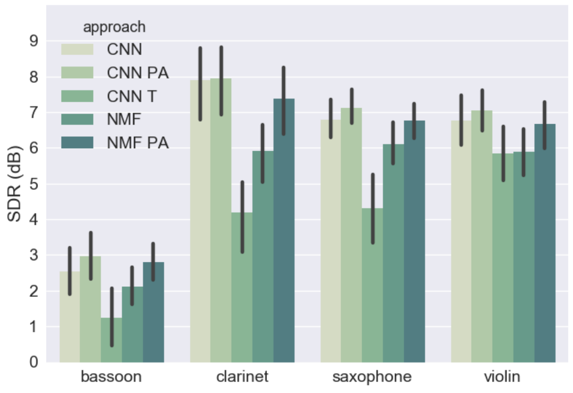


Figure 5. Results for each instrument in terms of SDR for the considered approaches: CNN and NMF [20]

We present the results in terms of SDR for each instrument in Figure 5. The CNN framework performs significantly better than the NMF for all the instruments, with the exception of bassoon. While experimenting with different STFT window sizes, we observed that the quality of the separation for bassoon improved considerably with the increase in the window size, while remaining the same for the other instruments. However, a larger window size means a higher feature dimensionality, hence more weights to be trained and a larger model.

We observe that the proposed framework effectively compensates for errors in alignment across all instruments, especially for clarinet.

The audio examples for the CNN framework and the computed metrics for CNN,CNN PA, CNN T, NMF, and NMF PA as .mat files can be accessed online [5] .

In the second experiment we are interested in testing the bootstrapping with replacement training procedure and the standard procedure. The results for various number of renditions can be seen in Figure 6.

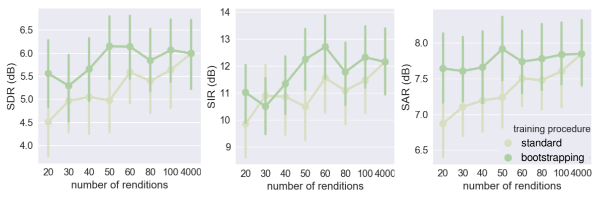


Figure 6. Results in terms of SDR,SIR,SAR when training the proposed CNN with stardard training method vs *bootstrapping with replacement* with various number of training samples

We observe that *bootstrapping with replacement* always improves over the standard training procedure, particularly for a small number of training renditions. However, a lower than 50 number of renditions, decreases the performance for both of the training methods. In some cases (50,60,100), using the proposed training procedure with fewer samples is slightly better than training with the whole dataset, as it prevents overfitting, in a similar way to early stopping [1]. The optimum number of renditions for our experimental scenario is 50 samples.

[5] `http://doi.org/10.5281/zenodo.821128`

5. CONCLUSION

We proposed a score-informed source separation framework targeted at Western classical music. Our framework is based on the assumption that classical music pieces are accompanied by scores and this information can be leveraged. Thus, we proposed a framework which is trained with generated renditions synthesized from the original scores. Provided an accurate automatic audio-to-score alignment can be obtained by a score-following system, our framework separates with low latency any real-life performances based on those scores, accompanied by a coarse alignment.

We presented a novel method to derive training features in the form of score-filtered spectrograms, which can easily be integrated with CNN architectures. In particular, these sorts of homogeneous features are well suited to learning convolutional filters which are shared between the input channels of the CNN.

The proposed system has better SDR and SIR than a state of the art score-informed NMF framework, particularly when working with coarsely aligned score, as it is the case of the output of score-following systems. Furthermore, we tested a faster training procedure, bootstrapping with replacement, which preserves the performance and in some cases prevents overfitting. As future work, we plan on extending this framework to multi-microphone orchestral music which is a more complex scenario due to increased number of instruments. Moreover, reiterating the method, by inputting the output of the network to another similar network, could improve results [27].

6. ACKNOWLEDGMENTS

The TITANX used for this research was donated by the NVIDIA Corporation. This work is partially supported by the Spanish Ministry of Economy and Competitiveness under CASAS project (TIN2015-70816-R) and by the Spanish Ministry of Economy and Competitiveness under the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). We thank Matthew E.P. Davies for his feedback.

7. REFERENCES

[1] Y Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[2] JJ Carabias-Orti, T Virtanen, P Vera-Candeas, N Ruiz-Reyes, and FJ Canadas-Quesada. Musical Instrument Sound Multi-Excitation Model for Non-Negative Spectrogram Factorization. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1144–1158, October 2011.

[3] P Chandna, M Miron, J Janer, and E Gómez. Monoaural audio source separation using deep convolutional neural networks. *International Conference on Latent Variable Analysis and Signal Separation*, 2017.

[4] Z. Duan and B. Pardo. Soundprism: An online system for score-informed source separation of music audio. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1205–1215, 2011.

[5] J.-L. Durrieu, A. Ozerov, C. Févotte, G. Richard, and B. David. Main instrument separation from stereophonic audio signals using a source/filter model. In *Signal Processing Conference, 2009 17th European*, pages 15–19. IEEE, 2009.

[6] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann. Subjective and Objective Quality Assessment of Audio Source Separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2046–2057, September 2011.

[7] S. Ewert and M. Müller. Using score-informed constraints for nmf-based source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 129–132. IEEE, 2012.

[8] S Ewert and MB Sandler. Structured dropout for weak label and multi-instance learning and its application to score-informed source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 2277–2281. IEEE, 2017.

[9] D Fitzgerald. Upmixing from mono-a source separation approach. In *Digital Signal Processing (DSP), 2011 17th International Conference on*, pages 1–7. IEEE, 2011.

[10] J. Fritsch and M.D. Plumbley. Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 888–891. IEEE, 2013.

[11] E. Gómez, F. Cañadas, J. Salamon, J. Bonada, P. Vera, and P. Cabañas. Predominant Fundamental Frequency Estimation vs Singing Voice Separation for the Automatic Transcription of Accompanied Flamenco Singing. *13th International Society for Music Information Retrieval Conference*, 2012.

[12] M. Goto. Development of the RWC music database. In *Proceedings of the 18th International Congress on Acoustics (ICA 2004)*, pages 553–556, 2004.

[13] E.M. Grais, M.U. Sen, and H. Erdogan. Deep neural networks for single channel source separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 3734–3738. IEEE, may 2014.

[14] R. Hennequin, B. David, and R. Badeau. Score informed audio source separation using a parametric model of non-negative spectrogram. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, number 1, pages 45–48. IEEE, 2011.

[15] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis. Deep Learning for Monaural Speech Separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 1562–1566, 2014.

[16] J. Janer, E. Gómez, A. Martorell, M. Miron, and B. de Wit. Immersive orchestras: audio processing for orchestral music VR content. In *Games and Virtual Worlds for Serious Applications (VS-Games), 2016 8th International Conference on*, pages 1–2. IEEE, 2016.

[17] R Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[18] A Liutkus, F-R Stöter, Z Rafii, D Kitamura, B Rivet, N Ito, N Ono, and J Fontecave. The 2016 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 323–332. Springer, 2017.

[19] Y. Luo, Z. Chen, J. R Hershey, J. Le Roux, and N. Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. *arXiv preprint arXiv:1611.06265*, 2016.

[20] M Miron, J.J. Carabias, and J Janer. Improving score-informed source separation for classical music through note refinement. *16th International Society for Music Information Retrieval Conference*, 2015.

[21] M Miron, JJ Carabias-Orti, JJ Bosch, E Gómez, and J Janer. Score-informed source separation for multichannel orchestral recordings. *Journal of Electrical and Computer Engineering*, 2016, 2016.

[22] M. Miron, J. Janer, and E. Gómez. Generating data to train convolutional neural networks for classical music source separation. In *Proceedings of the 14th Sound and Music Computing Conference*, pages 227–233, 2017.

[23] 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*, 98(6):995–1005, 2010.

[24] J. Salamon and J.P. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[25] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *16th International Society for Music Information Retrieval Conference*, pages 121–126, 2015.

[26] P. Smaragdis and S. Venkataramani. A neural network alternative to non-negative audio models. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 86–90. IEEE, 2017.

[27] S. Uhlich, F. Giron, and Y. Mitsufuji. Deep neural network based instrument extraction from music. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 2135–2139. IEEE, 2015.

[28] E. Vincent, R. Gribonval, and C. Fevotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech and Language Processing*, 14(4):1462–1469, jul 2006.

[29] T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1066–1074, 2007.

[30] JR Zapata and E Gomez. Using voice suppression algorithms to improve beat tracking in the presence of highly predominant vocals. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 51–55. IEEE, may 2013.

[31] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

DEEP SALIENCE REPRESENTATIONS FOR F_0 ESTIMATION IN POLYPHONIC MUSIC

Rachel M. Bittner[1*], **Brian McFee**[1,2], **Justin Salamon**[1], **Peter Li**[1], **Juan P. Bello**[1]
[1]Music and Audio Research Laboratory, New York University, USA
[2]Center for Data Science, New York University, USA
[*]Please direct correspondence to: rachel.bittner@nyu.edu

ABSTRACT

Estimating fundamental frequencies in polyphonic music remains a notoriously difficult task in Music Information Retrieval. While other tasks, such as beat tracking and chord recognition have seen improvement with the application of deep learning models, little work has been done to apply deep learning methods to fundamental frequency related tasks including multi-f_0 and melody tracking, primarily due to the scarce availability of labeled data. In this work, we describe a fully convolutional neural network for learning salience representations for estimating fundamental frequencies, trained using a large, semi-automatically generated f_0 dataset. We demonstrate the effectiveness of our model for learning salience representations for both multi-f_0 and melody tracking in polyphonic audio, and show that our models achieve state-of-the-art performance on several multi-f_0 and melody datasets. We conclude with directions for future research.

1. INTRODUCTION

Estimating fundamental frequencies in polyphonic music remains an unsolved problem in Music Information Retrieval (MIR). Specific cases of this problem include multi-f_0 tracking, melody extraction, bass tracking, and piano transcription among others. Percussion, overlapping harmonics, high degrees of polyphony, and masking make these tasks notoriously difficult. Furthermore, training and benchmarking is difficult due to the limited amount of human-labeled f_0 data available.

Historically, most algorithms for estimating fundamental frequencies in polyphonic music have been built on heuristics. In melody extraction, two algorithms that have retained the best performance are based on pitch contour tracking and characterization [8,27]. Algorithms for multi-f_0 tracking and transcription have been based on heuristics such as enforcing spectral smoothness and emphasizing harmonic content [17], comparing properties of co-occurring spectral peaks/non-peaks [11], and combining time and frequency-domain periodicities [29]. Other approaches to multi-f_0 tracking are data-driven and require labeled training data, e.g. methods based on supervised NMF [32], PLCA [3], and multi-label discriminative classification [23]. For melody extraction, machine learning has been used to predict the frequency bin of an STFT containing the melody [22], and to predict the likelihood an extracted frequency trajectory is part of the melody [4].

There are a handful of datasets with fully-annotated continuous-f_0 labels. The Bach10 dataset [11] contains ten 30-second recordings of a quartet performing Bach chorales. The Su dataset [30] contains piano roll annotations for 10 excerpts of real-world classical recordings, including examples of piano solos, piano quintets, and violin sonatas. For melody tracking, the MedleyDB dataset [5] contains melody annotations for 108 full length tracks that are varied in musical style.

More recently, deep learning approaches have been applied to melody and bass tracking in specific musical scenarios, including a BLSTM model for singing voice tracking [25] and fully connected networks for melody [2] and bass tracking [1] in jazz music. In multi-f_0 tracking, deep learning has also been applied to solo piano transcription [7,28], but nothing has been proposed that uses deep learning for multi-f_0 tracking in a more general musical context. In speech, deep learning has been applied to both pitch tracking [14] and multiple pitch tracking [18], however there is much more labeled data for spoken voice, and the space of pitch and spectrum variations is quite different than what is found in music.

The primary contribution of this work is a model for learning pitch salience representations using a fully convolutional neural network architecture, which is trained using a large, semi-automatically annotated dataset. Additionally, we present experiments that demonstrate the usefulness of the learned salience representations for both multi-f_0 and melody extraction, outperforming state-of-the-art approaches in both tasks. All code used in this paper, including trained models, is made publicly available. [1]

2. SALIENCE REPRESENTATIONS

Pitch salience representations are time-frequency representations that aim to measure the saliency (i.e. perceived am-

[1] github.com/rabitt/ismir2017-deepsalience

plitude/energy) of frequencies over time. They typically rely on the assumption that sounds humans perceive as having a pitch have some kind of harmonic structure. The ideal salience function is zero everywhere where there is no perceptible pitch, and a positive value that reflects the pitches' perceived loudness at the fundamental frequency. Salience representations are core components of a number of algorithms for melody [8, 12, 27] and multi-f_0 tracking [17,26]. Computations of salience representations usually perform two functions: (1) de-emphasize un-pitched or noise content (2) emphasize content that has harmonic structure.

The de-emphasis stage can be performed in a variety of ways, including harmonic-percussive source separation (HPSS), re-weighting frequency bands (e.g. using an equal loudness filter or a high pass filter), peak picking, or suppressing low amplitude or noise content [8, 12, 17, 26, 27]. In practice most salience functions also end up emphasizing harmonics and subharmonics because they are difficult to untangle from the fundamental, especially in complex polyphonies. The many parameters of these filtering and smoothing steps are typically set manually.

Harmonic content is most commonly emphasized via harmonic summation, which re-weights the input representation across frequency, where frequency bins in the salience representation are a weighted sum of harmonically related bins in the input representation [17, 27]. The weights in this summation vary from method to method, and are usually chosen heuristically based on assumptions about the data. In another variant, the input representation is modeled using non-negative least squares to a manually constructed set of ideal harmonic templates [19]. The Fan Chirp transform [9] uses harmonic information in the transform itself, thus directly performing the harmonic "weighting".

In melody extraction, the salience representation has been found to be a bottleneck in algorithmic performance [4], often because large portions of the melody are not emphasized. In particular, the salience representation used in Melodia [27] was found to emphasize vocal content well, but often miss instrumental content.

The combination of HPSS, equalization, and harmonic summation to emphasize pitched content and suppress the rest can be naturally extended in the context of deep learning architectures. For example, a simple version of HPSS performs median filtering with one kernel in time and frequency, and assigns bins to the harmonic or percussive component by a max filtering operation [13]. The harmonic and percussive decompositions can be cascaded to compute, for example, the harmonic component of the percussive signal as in [10, 25] to recover content that is not strongly activated by vertical or horizontal median filters such as singing voice. This cascade of median filtering can be naturally extended to a convolutional neural network setting, where instead of using only two manually set kernels, any number of kernels can be learned and their outputs combined in order to generalize to many types of musical sounds. Similarly, the parameters of harmonic summation can be implicitly learned by using an input representation that aligns harmonically related content—namely we introduce the *harmonic* CQT which we describe in Section 3.1. Furthermore, with a convolutional architecture, the parameters of the de-noising stage and the harmonic emphasis stage can be learned jointly.

3. METHOD

We frame our approach as a de-noising problem as depicted in Figure 1: given a time-frequency representation (e.g. a CQT), learn a series of convolutional filters that will produce a salience representation with the same shape in time and frequency. We constrain the target salience representation to have values between 0 and 1, where large values should occur in time-frequency bins where fundamental frequencies are present.

3.1 Input Representation

In order to better capture harmonic relationships, we use a *harmonic* constant-Q transform (HCQT) as our input representation. The HCQT is a 3-dimensional array indexed by harmonic, frequency, and time: $\mathcal{H}[h, t, f]$, measures the hth harmonic of frequency f at time t. The harmonic $h = 1$ refers to the fundamental, and we introduce the notation $\mathcal{H}[h]$ to denote harmonic h of the "base" CQT $\mathcal{H}[1]$. For any harmonic $h > 0$, $\mathcal{H}[h]$ is computed as a standard CQT where the minimum frequency is scaled by the harmonic: $h \cdot f_{\text{min}}$, and the same frequency resolution and number of octaves is shared across all harmonics. The resulting representation $\mathcal{H}$ is similar to a color image, where the h dimension is the *depth*.

In a standard CQT representation, the kth frequency bin measures frequency $f_k = f_{\text{min}} \cdot 2^{k/B}$ for B bins per octave. As a result, harmonics $h \cdot f_k$ can only be directly measured for $h = 2^n$ (for integer n), making it difficult to capture odd harmonics. The HCQT representation, however, conveniently aligns harmonics across the first dimension, so that the kth bin of $\mathcal{H}[h]$ has frequency $f_k = h \cdot f_{\text{min}} \cdot 2^{k/B}$, which is exactly the hth harmonic of the kth bin of $\mathcal{H}[1]$. By aligning harmonics in this way, the HCQT is amenable to modeling with two-dimensional convolutional neural networks, which can now efficiently exploit locality in time, frequency, and harmonic.

In this work, we compute HCQTs with $h \in \{0.5, 1, 2, 3, 4, 5\}$: one subharmonic below the fundamental (0.5), the fundamental (1), and up to 4 harmonics above the fundamental. Our hop size is $\approx$11 ms in time, and we compute 6 octaves in frequency at 60 bins per octave (20 cents per bin) with minimum frequency at $h = 1$ of $f_{\text{min}} = 32.7$ Hz (i.e. C1). We include a subharmonic in addition to harmonics to help disambiguate between the fundamental frequency and the first harmonic, whose patterns of upper harmonics are often similar – for the fundamental, the first subharmonic should have low energy, where for the first harmonic, a subharmonic below it will have energy. Our implementation is based on the CQT implementation in `librosa` [21].

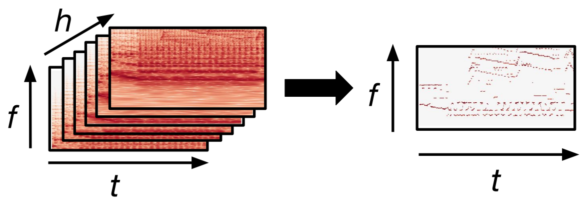


Figure 1. Input HCQT (left) and target salience function (right).

3.2 Output Representation

The target outputs we use to train the model are time-frequency representations with the same shape as $\mathcal{H}[1]$. Ground truth fundamental frequency values are quantized to the nearest time-frequency bin, and given magnitude $= 1$ in the target representation. The targets are Gaussian blurred in frequency such that the energy surrounding a ground truth frequency decays to zero within a quarter-tone, in order to soften the penalty for near-correct predictions during training. Additionally, since the data is human labeled it may not be accurate to 20 cents, so we do not necessarily want to label nearby frequencies as "wrong". Similar training label "blurring" techniques have been shown to help the performance of models for beat/downbeat tracking [6] and structural boundary detection [31].

3.3 Model

Our model uses a fully convolutional architecture, with 5 convolutional layers of varying dimensionality, as illustrated in Figure 2. The first two layers have 128 and 64 (5 x 5) filters respectively, which cover approximately 1 semitone in frequency and 50 ms in time. The following two layers each have 64 (3 x 3) filters, and the final layer has 8 (70 x 3) filters, covering 14 semitones in frequency to capture relationships between frequency content within an octave. At each layer, the convolutions are zero padded such that the input shape is equal to the output shape in the time-frequency dimension. The input to each layer is batch normalized [15], and the outputs are passed through rectified linear units. The final layer uses logistic activation, mapping each bin's output to the range $[0, 1]$. The predicted saliency map can be interpreted as a likelihood score of each time-frequency bin belonging to an f_0 contour. Note that we do not include pooling layers, since we do *not* want to be invariant to small shifts in time frequency.

The model is trained to minimize cross entropy:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \qquad (1)$$

where both y and $\hat{y}$ are continuous values between 0 and 1. We fit our model using the Adam [16] optimizer.

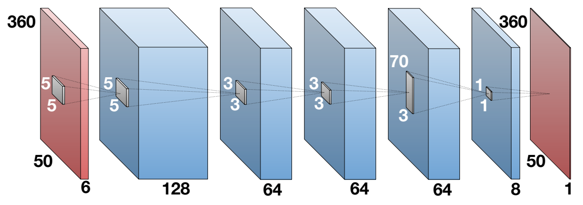


Figure 2. CNN architecture. The input to each layer is batch-normalized. The output of each layer is passed through a rectified linear unit activation function except the last layer which is passed through a sigmoid.

4. MULTIPLE-F_0 TRACKING EXPERIMENTS

We first explore the usefulness of our model when trained to produce a multi-f_0 salience representation.

4.1 Data Generation

Because there is no large human-labeled dataset to use for training, we generate a dataset from a combination of human and machine generated f_0 annotations by leveraging multitrack data. Our total dataset contains 240 tracks from a combination of the 108 MedleyDB multitrack dataset [5] and a set of 132 pop music multitracks. The pop multitrack set consists of western popular music from the 1980s through today, and were obtained from a variety of sources and are not available for redistribution—because of this we only use these examples during training. The tracks are split into train, validate, and test groups using an artist-conditional randomized split (i.e. tracks belonging to the same artist must all belong to the same group). The test set is constrained to contain only tracks from MedleyDB, and contains 28 full-length tracks. The training and validation sets contain 184 and 28 full-length tracks respectively, totaling to about 10 hours of training data and 2 hours of validation data.

Each multitrack in the dataset contains mixes and isolated stems, and a subset of these stems contain human-labeled f_0 annotations. To have a mix where all pitched content is annotated, we re-create partial mixes by combining any stems with human annotations, all stems with monophonic instruments (e.g. electric bass), and all percussive stems, effectively creating mixes that are similar to the originals, but with all "unknown" pitch content removed. The stems are linearly mixed with weights estimated from the original mixes using a least squares fit. The human-labeled f_0 annotations are directly added to the ground truth labels. Annotations for monophonic instrument stems without human labels are created by running `pYIN` [20] and using the output as a proxy for ground truth.

4.2 Results

To generate multi-f_0 output, we need to explicitly select a set of fundamental frequency values for each time frame from our salience representation. A natural way to do this would be to threshold the representation at 0.5, however since the model is trained to reproduce Gaussian-blurred frequencies, the values surrounding a high energy bin are usually above 0.5 as well, creating multiple estimates very close to one another. Instead, we perform peak picking on the learned representation and select a minimum amplitude threshold by choosing the threshold that maximizes the multi-f_0 accuracy on the validation set.

We evaluate the model on three datasets: the Bach10 and Su datasets, and the test split of the MedleyDB data described in Section 4.1, and compare to well-performing baseline multi-f_0 algorithms by Benetos [3] and Duan [11].

Figure 3 shows the results for each algorithm on the three datasets. We see that our CNN model under-performs on Bach10 compared to Benetos' and Duan's models by about 10 percentage points, but outperforms both algorithms on the Su and MedleyDB datasets. We attribute the difference in performance across these datasets to the way each model was trained. Both Benetos' and Duan's methods were in some sense developed with the Bach10 dataset in mind simply because it has been one of the few available test sets when the algorithms were developed. On the other hand, our model was trained on data most similar to the MedleyDB test set, so it is unsurprising that it performs better on this set. The Bach10 dataset is homogeneous (as can be seen by the small variance in performance across all methods), and while our model performs obtains higher scores on the Bach10 dataset than the other two used for evaluation, this dataset only measures how well an algorithm performs on simple 4-part harmony classical recordings. Indeed, we found that on the MedleyDB test set, both Benetos' and Duan's models perform best (50% and 48% accuracy respectively) on the example that is most similar to the Bach10 data (a string quartet), and our approach performs similarly on that track to the overall performance on the Bach10 set with 59% accuracy.

To get a better sense of the track level performance, Figure 4 displays the difference between the CNN accuracy and the best accuracy of Benetos and Duan's model per track. In addition to having a better score on average for MedleyDB (from Figure 3), we see that the CNN model outperforms the other two models on every track on MedleyDB by quite a large margin. We see a similar result for the Su dataset, though on one track (Beethoven's Moonlight sonata) we have a lower score than Benetos. A qualitative analysis of this track showed that our algorithm retrieves the melody and the bass line, but fails to emphasize several notes that are part of the harmony line. Unsurprisingly, on the Bach10 dataset, the other two algorithms outperform our approach for every track.

To further explain this negative result, we explore how our model will perform in an oracle scenario by constraining the maximum polyphony to 4 (the maximum for the Bach10 dataset) and look at the accuracy when we vary the detection threshold. Figure 5 shows the CNN's average accuracy on the Bach10 dataset as a function of the detection thresholds. The solid dotted line shows the threshold automatically estimated from the validation set. For the Bach10 dataset, the optimal threshold is much lower (0.05 vs. 0.3), and the best performance (63% accuracy) gets closer to that of the other two datasets (68% for Duan and 76% for Benetos). Even in this ideal scenario, the difference in performance is due to recall – similarly to the Su example, our algorithm is good at retrieving the melody and bass lines in the Bach10 dataset, but often misses notes that occur in between. This is likely a result of the characteristics of the artificial mixes in our training set: the majority of automatically annotated (monophonic) stems are either bass or vocals, and there are few examples with simultaneous harmonically related pitch content.

Overall, our model has good precision, even on the Bach10 dataset (where the scores are hurt by recall), which suggests that the learned salience function does a good job of de-emphasizing non-pitched content. However, the low recall on the Bach10 and Su datasets suggests that there is still room for the model to improve on emphasizing harmonic content. Compared to the other two algorithms, the CNN makes fewer octave mistakes (3% of mistakes on MedleyDB compared with 5% and 7% of mistakes for Benetos and Duan respectively), reflected in the difference between the accuracy and chroma accuracy.

While the algorithm improves on the state of the art on two datasets, the overall performance still has a lot of room to improve, with the highest score on the Su dataset reaching only 41% accuracy on average. To explore this further, in Figure 6 we plot the outputs on excerpts of tracks from each of the three datasets. In each of the excerpts, the outputs look reasonably accurate. The top row shows an excerpt from Bach10, and while our model sometimes misses portions of notes, the salient content (e.g. melody and bass) is emphasized. Overall, we observe that the CNN model is good at identifying bass and melody patterns even when higher polyphonies are present, while the other two models try to identify chords, even when only melody and bass are present.

4.3 Model Analysis

The output of the CNN for an unseen track from the Su dataset is shown in Figure 7. $\mathcal{H}[1]$ is plotted in the left plot, and we can see that it contains a complex polyphonic mixture with many overlapping harmonics. Qualitatively, we see that the CNN was able to de-noise the input representation and successfully emphasize harmonic content.

To better understand what the model learned, we plot the 8 feature maps from the penultimate layer in Figure 8. The red-colored activations have positive weights and the blue-colored have negative weights in the output filter. Activations (a) and (b) seem to emphasize harmonic content, including some upper harmonics. Interestingly, activation (e) deemphasizes the octave mistake from activation (a), as does activation (d). Similarly, activations (f) and (g) act as a "cut out" for activations (a) and (b), deemphasizing the broadband noise component. Activation (h) appears to deemphasize low-frequency noise.

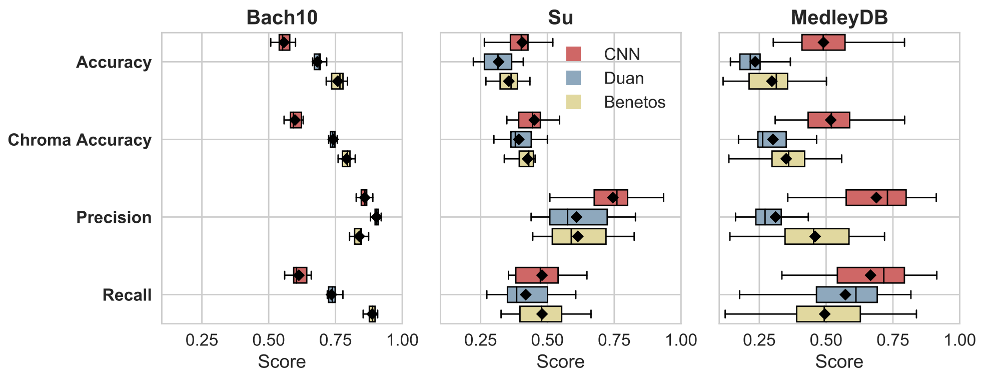


Figure 3. A subset of the standard multiple-f0 metrics on the Bach10, Su, and MedleyDB test sets for the proposed CNN-based method, Duan [11], and Benetos [3].

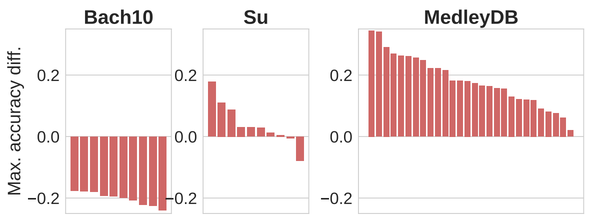


Figure 4. The per-track difference in accuracy between the CNN model and the maximum score achieved by Duan or Benetos' algorithm on each dataset. Each bar corresponds to CNN - $\max(\text{Duan, Benetos})$ on a single track.

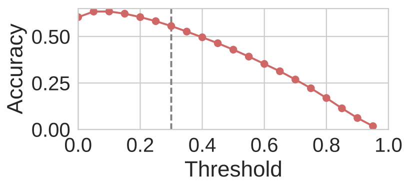


Figure 5. CNN accuracy on the Bach10 dataset as a function of the detection threshold, and when constraining the maximum polyphony to 4. The vertical dotted line shows the value of the threshold chosen on the validation set.

5. MELODY ESTIMATION EXPERIMENTS

To further explore the usefulness of the proposed model for melody extraction, we train a CNN with identical an architecture on melody data.

5.1 Data Generation

Instead of training on HCQTs computed from partial mixes and semi-automatic targets (as described in Section 4.1), we use HCQTs from the original full mixes from MedleyDB, as well as targets generated from the human-labeled melody annotations. The ground truth salience functions contain only melody labels, using the "Melody 2" definition from MedleyDB (i.e. one melody pitch per unit time coming from multiple instrumental sources). We estimate the melody line from the learned salience repre-

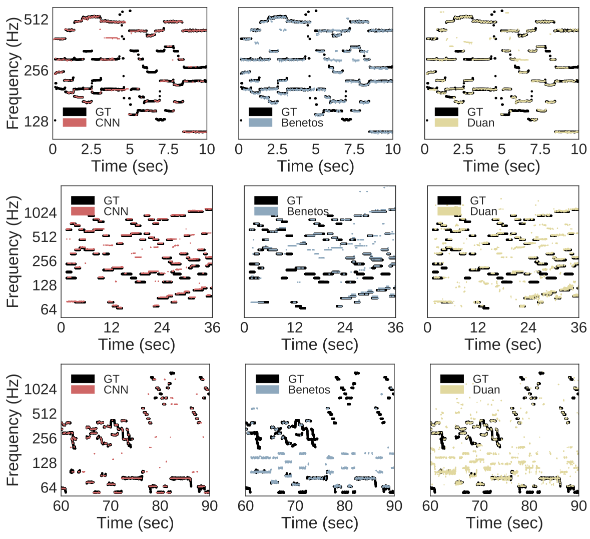


Figure 6. Multi-f0 output for each of the 3 algorithms for an example track from the Bach10 dataset (top), the Su dataset (middle), and the MedleyDB test set (bottom)

sentation by choosing the frequency with the maximum salience at every time frame. The voicing decision is determined by a fixed threshold chosen on the validation set. In this work we did not explore more sophisticated decoding methods.

5.2 Results

We compare the output of our CNN-based melody tracking system with two strong, salience-based baseline algorithms: "Salamon" [27] and "Bosch" [8]. The former is a heuristic algorithm that long held the state of the art in melody extraction. The latter recently reached state-of-the-art performance by combining a source-filter based salience function and heuristic rules for contour selection—this model is the current best performing baseline. Figure 9 shows the results of the three methods on the MedleyDB test split described in Section 4.1.

On average, the CNN-based melody extraction outperforms both Bosch and Salamon in terms of Overall (+ 5 and

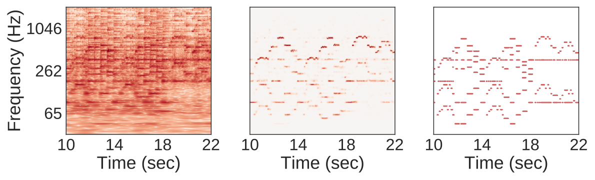


Figure 7. (left) Input $\mathcal{H}[1]$, (middle) predicted output, (right) ground truth annotation for an unseen track in the Su dataset.

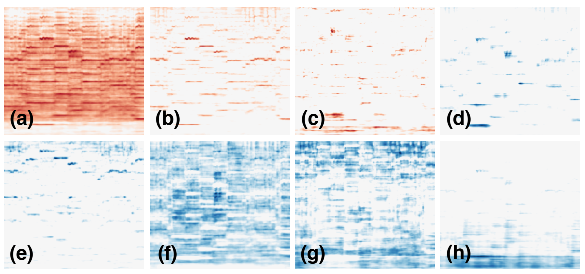


Figure 8. Activations from the final convolutional layer with octave height filters for the example given in Figure 7. Activations (a)–(c) have positive coefficients in the output layer, while the others have negative coefficients.

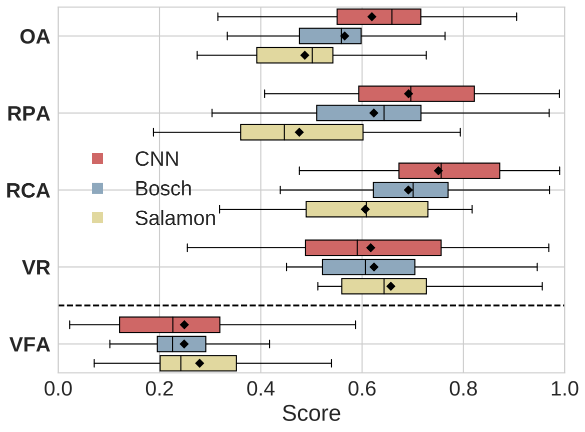


Figure 9. Melody metrics – Overall Accuracy (OA), Raw Pitch Accuracy (RPA), Raw Chroma Accuracy (RCA), Voicing Recall (VR) and Voicing False Alarm (VFA) – on the MedleyDB test set for the proposed CNN-based method, Salamon [27], and Bosch [8].

13 percentage points), Raw Pitch (+15 and 22 percentage points), and Raw Chroma Accuracy (+6 and 14 percentage points). The CNN approach is also considerably more varied in performance than the other two algorithms, with a wide range in performance across tracks.

Because we choose the frequency with maximum amplitude in our approach, the Raw Pitch Accuracy measures effectiveness of the salience representation: in an ideal salience representation for melody, the melody should have the highest amplitude in the salience function over time. In our learned salience function, $\approx$ 62% of the time the melody has the largest amplitude. A qualitative analysis

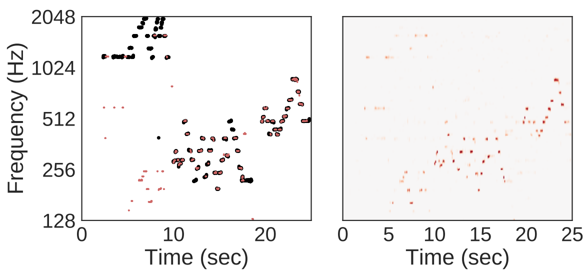


Figure 10. CNN output on a track beginning with a piano melody (0 - 10 seconds) and continuing with a clarinet melody (10 - 25 seconds). (left) CNN model melody output in red against the ground truth in back. (right) CNN melody salience output.

of the mistakes made by the CNN method revealed that the vast majority incorrect melody estimates occurred for melodies played by under-represented melody instrument classes in the training set, such as piano and guitar. For example, Figure 10 shows the output of the CNN model for an excerpt beginning with a piano melody and continuing with a clarinet melody. Clarinet is well represented in our training set and the model is able to retrieve most of the clarinet melody, while virtually none of the piano melody is retrieved. Looking at the salience output (Figure 10 right), there is very little energy in the early region where the piano melody is active. This could be a result of the model not being exposed to enough examples of the piano timbre to activate in those regions. Alternatively, in melody salience scenario, the model is trained to suppress "accompaniment" and emphasize melody. Piano is often playing accompaniment in the training set, and the model may not have enough information to untangle when a piano timbre should be emphasized as part of the melody and when it should be suppressed as accompaniment. We note that while in this qualitative example the errors could be attributed to the pitch height, we observed that this was not a consistent factor in other examples.

6. CONCLUSIONS

In this paper we presented a model for learning a salience representation for multi-f_0 tracking and melody extraction using a fully convolutional neural network. We demonstrated that simple decoding of both of these salience representations yields state-of-the art results for multi-f_0 tracking and melody extraction. Given a sufficient amount of training data, this architecture would also be useful for related tasks including bass, piano, and guitar transcription.

In order to further improve the performance of our system, data augmentation can be used to both diversify our training set and to balance the class distribution (e.g. include more piano and guitar). The training set could further be augmented by training on a large set of weakly-labeled data such as the Lakh-midi dataset [24]. In addition to augmentation, there is a wide space of model architectures that can be explored to add more temporal information, such as recurrent neural networks.

7. REFERENCES

[1] Jakob Abeßer, Stefan Balke, Klaus Frieler, Martin Pfleiderer, and Meinard Müller. Deep learning for jazz walking bass transcription. In *AES International Conference on Semantic Audio*, 2017.

[2] Stefan Balke, Christian Dittmar, Jakob Abeßer, and Meinard Müller. Data-driven solo voice enhancement for jazz music retrieval. In *ICASSP*, Mar. 2017.

[3] Emmanouil Benetos and Tillman Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *ISMIR*, pages 701–707, 2015.

[4] Rachel M Bittner, Justin Salamon, Slim Essid, and Juan P Bello. Melody extraction by contour classification. In *ISMIR*, October 2015.

[5] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan P. Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *ISMIR*, October 2014.

[6] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. of the 17th Int. Society for Music Information Retrieval Conf.(ISMIR)*, 2016.

[7] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP), 2012 ieee international conference on*, pages 121–124. IEEE, 2012.

[8] Juan José Bosch, Rachel M Bittner, Justin Salamon, and Emilia Gómez. A comparison of melody extraction methods based on source-filter modeling. In *ISMIR*, pages 571–577, New York, August 2016.

[9] Pablo Cancela, Ernesto López, and Martín Rocamora. Fan chirp transform for music representation. In *DAFx*, 2010.

[10] Jonathan Driedger and Meinard Müller. Extracting singing voice from music recordings by cascading audio decomposition techniques. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 126–130. IEEE, 2015.

[11] Zhiyao Duan, Bryan Pardo, and Changshui Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE TASLP*, 18(8):2121–2133, 2010.

[12] Jean-Louis Durrieu, Bertran David, and Gael Richard. A musically motivated mid-level representation for pitch estimation and musical audio source separation. *IEEE J. on Selected Topics on Signal Processing*, 5(6):1180–1191, Oct. 2011.

[13] Derry Fitzgerald. Harmonic/percussive separation using median filtering. 2010.

[14] Kun Han and DeLiang Wang. Neural network based pitch tracking in very noisy speech. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):2158–2168, 2014.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[16] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] Anssi Klapuri. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE TASLP*, 11(6):804–816, Nov. 2003.

[18] Yuzhou Liu and DeLiang Wang. Speaker-dependent multipitch tracking using deep neural networks. *The Journal of the Acoustical Society of America*, 141(2):710–721, 2017.

[19] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, pages 135–140, 2010.

[20] Matthias Mauch and Simon Dixon. PYIN: a Fundamental Frequency Estimator Using Probabilistic Threshold Distributions. In *ICASSP*, pages 659–663. IEEE, 2014.

[21] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, and et al. librosa 0.5.0, Feb 2017.

[22] Graham E. Poliner and Dan PW Ellis. A classification approach to melody transcription. In *ISMIR*, pages 161–166, London, Sep. 2005.

[23] Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1):154–154, 2007.

[24] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, COLUMBIA UNIVERSITY, 2016.

[25] François Rigaud and Mathieu Radenen. Singing voice melody transcription using deep neural networks. In *ISMIR*, pages 737–743, 2016.

[26] Matti Ryynänen and Anssi Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music J.*, 32(3):72–86, 2008.

[27] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE TASLP*, 20(6):1759–1770, Aug. 2012.

[28] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.

[29] Li Su and Yi-Hsuan Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(10):1600–1612, 2015.

[30] Li Su and Yi-Hsuan Yang. Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *International Symposium on Computer Music Multidisciplinary Research*, pages 309–321. Springer, 2015.

[31] Karen Ullrich, Jan Schlüter, and Thomas Grill. Boundary detection in music structure analysis using convolutional neural networks. In *ISMIR*, pages 417–422, 2014.

[32] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):528–537, 2010.

AN ANALYSIS/SYNTHESIS FRAMEWORK FOR AUTOMATIC F0 ANNOTATION OF MULTITRACK DATASETS

Justin Salamon[1*], Rachel M. Bittner[1], Jordi Bonada[2], Juan J. Bosch[2], Emilia Gómez[2] and Juan Pablo Bello[1]
[1]Music and Audio Research Laboratory, New York University, USA
[2]Music Technology Group, Universitat Pompeu Fabra, Spain
[*]Please direct correspondence to: `justin.salamon@nyu.edu`

ABSTRACT

Generating continuous f_0 annotations for tasks such as melody extraction and multiple f_0 estimation typically involves running a monophonic pitch tracker on each track of a multitrack recording and manually correcting any estimation errors. This process is labor intensive and time consuming, and consequently existing annotated datasets are very limited in size. In this paper we propose a framework for automatically generating continuous f_0 annotations without requiring manual refinement: the estimate of a pitch tracker is used to drive an analysis/synthesis pipeline which produces a synthesized version of the track. Any estimation errors are now reflected in the synthesized audio, meaning the tracker's output represents an accurate annotation. Analysis is performed using a wide-band harmonic sinusoidal modeling algorithm which estimates the frequency, amplitude and phase of every harmonic, meaning the synthesized track closely resembles the original in terms of timbre and dynamics. Finally the synthesized track is automatically mixed back into the multitrack. The framework can be used to annotate multitrack datasets for training learning-based algorithms. Furthermore, we show that algorithms evaluated on the automatically generated/annotated mixes produce results that are statistically indistinguishable from those they produce on the original, manually annotated, mixes. We release a software library implementing the proposed framework, along with new datasets for melody, bass and multiple f_0 estimation.

1. INTRODUCTION

Research on Music Information Retrieval (MIR) tasks such as melody extraction and multiple f_0 estimation requires audio datasets annotated with precise, continuous, sometimes multiple, f_0 values at time-scales on the order of milliseconds. Generating such annotations manually is very time consuming and labor intensive, and thus insufficient to sustain current research efforts. This is aggravated by the lack of educational or other intrinsic motivations for performing f_0 annotations, limiting the applicability of gamification and other crowdsourcing strategies to this problem. Alternative solutions for f_0 annotation include the use of instruments outfitted with sensors that are able to simultaneously generate audio and annotations [18], or of MIDI-controlled instruments to support annotation by playing [39]. Such approaches are limited either in the type of sources that they can use, e.g. piano, or in the annotations they can generate, e.g. notes instead of continuous f_0. Other approaches rely on audio to MIDI alignment [19], but are limited both by the robustness of the alignment and, to a lesser extent, the availability of good quality MIDI data. Perhaps the most common methodology for annotating f_0 is to use automatic f_0 estimation methods on monophonic stems of existing multitracks [7, 16, 29]. However, the limited accuracy of the estimation has the potential to create discrepancies between the audio and the annotation [16], and correcting such discrepancies is in itself very laborious. For example, manual corrections for MedleyDB (108 songs, most 3–5 minutes long) required approximately 50 hours of effort across annotators [7,29]. As a result, existing datasets for f_0 estimation in polyphonic music (whether for melody, bass, or multiple f_0) are extremely small: most such datasets are on the order of tens of recordings with a total duration of less than an hour. Even MedleyDB is but a fraction of the size of datasets used in other MIR tasks [4], speech recognition [12] or image recognition [14]. This is particularly problematic for developing data-driven solutions to f_0 estimation, which require large amounts of annotated audio data.

To tackle this problem, the MIR community, and the machine learning (ML) community in general, have proposed solutions based on *data augmentation* and *data synthesis*. Augmentation involves the transformation of existing data, and has been shown to improve the generalizability of ML models across domains [25, 31]. However, if the initial dataset is very small there is a limit to the benefits of augmentation, and thus researchers have also explored data synthesis approaches, e.g. for chord recognition [27], monophonic pitch tracking [30] or environmental sound analysis [26]. The earliest dataset for melody extraction, ADC2004 [10], contains some synthesized vo-

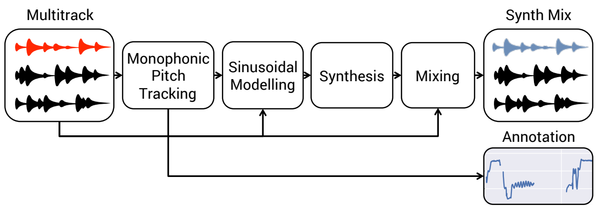


Figure 1. Block diagram of the proposed framework.

cal tracks and is still in use for melody extraction evaluation in MIREX [15] today. Synthesized data is not only useful for model training, it can also be used for model evaluation [26]. As the authors of that study note, while evaluation on synthesized data might not always represent model performance on real-world data, it allows for a detailed and controlled comparative evaluation using significantly larger amounts of data, which can provide invaluable insight into the comparative performance of different models under different, controlled, audio conditions.

Building on these ideas, in this paper we present a method for continuous f_0 annotation that is fully automatic. The key concept is the use of multitrack recordings in combination with an analysis/synthesis framework: starting with a multitrack recording, we select a monophonic instrument track that we are interested in annotating, and run a monophonic pitch tracker to obtain its f_0 curve. Since the f_0 estimate is likely to contain (albeit a small amount of) errors, it would be methodologically unsound to treat it as a reference annotation for either training or evaluation. Instead, we use it as the input to a wide-band harmonic modelling algorithm that estimates not just the frequency of the f_0, but the frequency, amplitude and phase of every harmonic in the signal. We use this information to re-synthesize the monophonic recording, resulting in an audio signal that perfectly matches the f_0 curve produced by the pitch tracker. Thanks to the wide-band harmonic modelling, the synthesized track is very similar to the original recording in pitch, timbre and dynamics [1]. Finally, we mix the synthesized track back with the rest of the instruments in the multitrack recording, resulting in a polyphonic music mixture for which we have an accurate, fully automatic annotation of the synthesized track. A block diagram of the proposed framework is displayed in Figure 1. The methodology can be used to automatically generate annotations for working on melody extraction, bass line extraction and multiple f_0 estimation, and essentially any model designed to extract f_0 content from polyphonic music mixtures.

The proposed framework can be readily used to generate training data. The question remains whether using the synthesized mixes as evaluation data produces a representative measure of model performance. To answer this question, after describing the framework we present a series of experiments designed to explore whether the synthesized mixes result in performance scores that are representative of the scores algorithms obtain on the original mixes. As a final contribution of this work, we release a software library implementing the proposed framework [2], as well as new datasets for melody, bass, and multiple f_0 estimation [3].

[1] For examples of synthesized tracks (solo and mixed with the multitrack) see: `http://synthdatasets.weebly.com/examples`

2. METHOD

2.1 Pitch Track Analysis/Synthesis

2.1.1 Pitch Tracking

We use a monophonic pitch tracker to get an initial f_0 estimate of the stem we would like to annotate. We tested SAC [21] and YIN [13] and compared both to the manually corrected f_0 annotations provided in MedleyDB [7]. Based on this comparison we decided to use SAC for our experiments, see Section 3.2 for further details. The output of SAC is automatically cleaned by filling short gaps (<50 ms), removing short voiced segments (<50 ms), and smoothing the voiced segments. Note that we do not use pYIN [30], a state-of-the-art pitch tracking algorithm, since the manually corrected annotations in MedleyDB are based on the output of this algorithm and so using it for this stage could bias our experimental results. Still, it is important to note that the methodology is independent of the specific pitch tracker used, and the software library we release supports multiple monophonic pitch trackers, including pYIN.

2.1.2 Sinusoidal Modelling

We use the wide-band harmonic sinusoidal modelling algorithm [8] for estimating the harmonic parameters (frequency, amplitude and phase) at every signal period. The algorithm first segments the signal into periods corresponding to the fundamental frequency. Then each period is analyzed with a certain windowing configuration that has the property that the Fourier transform of the window has the zeros located at multiples of the f_0. This property reduces the interference between harmonics, and allows the estimation of harmonic parameters using a temporal resolution close to one period of the signal. For details see [8].

2.1.3 Synthesis

The synthesis is performed with a bank of oscillators. The harmonics' parameters previously estimated are linearly interpolated at the synthesis sampling rate. Frequencies are set to exact multiples of the f_0. Phases are arbitrarily initialized at each voiced segment with a non-flat shape to avoid producing signals that are too peaky:

$$\Phi_h = \pi + \frac{\pi}{2}\sin\left(\frac{h}{20\pi} + \pi\right) \tag{1}$$

where h corresponds to the harmonic index, and Φ is the harmonic phase. Phases are incremented at each sample using the interpolated frequency value. At voiced segment boundaries harmonic amplitudes are faded out to zero within one signal period. Unvoiced segments are muted.

[2] `https://github.com/marl/massage`
[3] `http://synthdatasets.weebly.com/`

2.2 Remixing

The final step is to recreate a mix of the song that is as close as possible to the original. Even when using the original stems as source material, a simple unweighted sum of the stems will not necessarily be a good approximation: the stems may not be the same volume as they occur in the mix, and the final mix may have mastering effects such as compression or equalization. To estimate the mixing weights, we model the (time-domain) mix $y[n]$ as a weighted linear combination[4] of the original stems $x_1, x_2, \ldots, x_M$:

$$y[n] \approx \sum_{i=1}^{M} a_i x_i[n] \tag{2}$$

where $x_i[n]$ is the audio signal at sample n for stem i and M is the total number of stems. Let N be the total number of samples in each audio signal. We then estimate the mixing weights a_i by minimizing a non-negative least squares objective $||X\mathbf{a} - Y||_2$ over $\mathbf{a}$ for $a_i > 0$, where X is the $N \times M$ matrix of the absolute values of the stem audio signals $|x[n]|$, $\mathbf{a}$ is the $M \times 1$ vector of mixing weights a_i, and Y the $N \times 1$ is the absolute value of the mixture audio signal $|y[n]|$. We use the computed weights $\mathbf{a}$ to create a (linear) remix $\tilde{y}[n]$, substituting the melody track(s) (or bass track or multiple instrument tracks) $\tilde{x}_1, \ldots, \tilde{x}_I$ with the synthesized stems:

$$\tilde{y}[n] = \sum_{i=1}^{I} a_i \tilde{x}_i[n] + \sum_{i=I+1}^{M} a_i x_i[n] \tag{3}$$

3. EXPERIMENTS

As noted above, the proposed framework can be readily used for generating training data. However, and perhaps precisely due to the problem of data scarcity, current state-of-the-art algorithms for melody extraction (e.g., [9, 17, 35]) and multiple f_0 estimation (e.g., [3, 16, 24]) are either fully or partially based on heuristic DSP pipelines, meaning it is not possible to demonstrate an improvement due to additional training data, as these systems do not have a learning stage (or the learning happens towards the end of the pipeline and the main source of errors is the heuristic front-end [6]). We are actively working on f_0 estimation algorithms based on deep models that operate on a low-level representation of the signal [5], and plan to evaluate their performance when trained on synthesized data as part of our future work.

Instead, we explore the representativeness of the synthesized mixes for the purpose of model evaluation. To this end, we run a series of evaluation experiments, once using the original mixes and annotations and a second time using the synthesized mixes and automatically generated annotations. The experiments involve evaluating several melody extraction and multiple f_0 estimation algorithms. Ideally, we would like the scores obtained by each algorithm to remain unchanged between the original and synthesized mixes, as this would indicate that the synthesized (automatically annotated) mixes can be used to obtain realistic estimates of model performance, opening the door to the generation of significantly larger datasets not only for model training, but also for model evaluation.

3.1 Data

We use the MedleyDB dataset [7] to evaluate the proposed methodology for melody f_0 annotation. Of the 108 tracks containing melodies, we need to filter out tracks that are not completely monophonic such as those containing recording bleed from other instruments and melody tracks played by polyphonic instruments such as the piano and guitar. After filtering we end up with 65 songs, for which we generate new mixes and melody f_0 annotations following the methodology described in Section 2. The remixing is performed using the `medleydb` python module[5]. We call the resulting dataset MDB-melody-synth.

For multiple f_0 estimation we use the Bach10 dataset [16]. The dataset contains ten pieces of four-part (soprano, alto, tenor, bass) J.S. Bach chorales performed by the violin, clarinet, saxophone and bassoon, respectively. The synthesized dataset including new mixes and automatically generated multiple f_0 annotations, Bach10-mf0-synth, was created following the methodology described in Section 2, the only difference being that since the original mixes are just unweighted sums of the stems, the synthesized mixes are also unweighted.

Finally, we use the proposed methodology to create a synthesized version of MedleyDB with multiple f_0 annotations, MDB-mf0-synth, and another version in which only the bass track is synthesized (for bass line extraction), MDB-bass-synth. For MDB-mf0-synth, we need to filter out stems that are not monophonic. For instance, if the original mix contains drums, bass, piano, guitar, trumpet and singing voice, the new mix will contain drums, bass, trumpet and voice. We must also discard tracks that are left with only percussive instruments after removing all non-monophonic stems. After filtering we are left with 85 songs, for which we generate new mixes and multiple f_0 annotations as per Section 2. Most of the mixes in the resulting dataset have a polyphony between 1 and 4, but there are also songs with higher polyphonies, up to 16. Overall, the mixes in the new dataset include 25 different instruments (not counting percussive instruments) which are combined to produce 29 unique instrumentations (not counting percussive instruments). For MDB-bass-synth we can use all tracks that contain a bass line with no recording bleed, resulting in a dataset of 71 songs. To the best of our knowledge this is the largest publicly available dataset with continuous bass f_0 annotations. Note that due to space constraints we do not use this dataset in the experiments reported in this paper. All four new datasets, MDB-melody-synth, MDB-mf0-synth, MDB-bass-synth and Bach10-mf0-synth are made freely available online (cf. footnote 3).

[4] Recreating mastering effects is left for future work.

[5] `https://github.com/marl/medleydb`

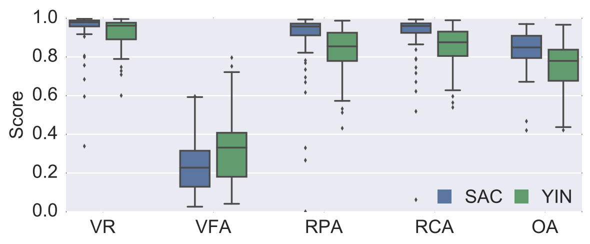


Figure 2. f_0 tracking scores for SAC and YIN evaluated against the MedleyDB manually corrected f_0 annotations.

3.2 Monophonic Pitch Tracking

We start by evaluating the pitch tracking accuracy of the SAC and YIN algorithms on the 65 monophonic melody stems from MedleyDB, presented in Figure 2. We use `mir_eval` [33] to compute the standard five evaluation metrics used in MIREX: Voicing Recall (VR), Voicing False Alarm (VFA), Raw Pitch Accuracy (RPA), Raw Chroma Accuracy (RCA) and Overall Accuracy (OA). For details about the metrics see [36]. We see that SAC produces a more accurate f_0 estimate compared to YIN for these data, with a mean raw pitch accuracy of 0.9. The overall accuracy is slightly lower due to voicing false positives, but these frames will turn into voiced frames in the synthesized mixes thus accurately matching the annotation. This is the key advantage of the proposed approach: pitch tracking errors do not cause a mismatch between the audio and the annotation and require no manual correction. Since 90% of the f_0 values in MDB-melody-synth match those in MedleyDB, we can also safely say the synthesized dataset is representative of the original in terms of continuous pitch values. Finally, since SAC makes practically no octave errors (the difference between the RPA and RCA is below 0.02), there is little to no risk of a perceptual mismatch between the estimated f_0 and the synthesized audio.

3.3 Melody Extraction

To evaluate the representativeness of MDB-melody-synth compared to MedleyDB, we evaluate the performance of three melody extraction algorithms: Melodia [35], the source-separation-based algorithm by Durrieu [17], and the recently proposed algorithm by Bosch [9] which uses a salience function based on Durrieu's model in combination with the contour characterization employed in Melodia for voicing detection and melody selection.

In Figure 3(a) we plot the results obtained by the Melodia algorithm, where for each metric we plot the result for the original mixes and the MDB-melody-synth mixes side-by-side. We see that while the results are not identical, the distribution of scores for each metric remains stable. A two-sided Kolmogorov-Smirnov test confirms that for all 5 metrics the differences in the score distributions between the original and synthesized datasets are not statistically significant (p-values of 0.39, 0.05, 0.68, 0.28 and 0.82 for VR, VFA, RPA, RCA and OA respectively). We repeat the same experiment for the algorithms by Durrieu and Bosch, displayed in Figure 3 subplots (b) and (c) respectively. As before, the score distributions for all metrics remain stable and the difference between them is not statistically significant. The only exception is the OA score for Durrieu's algorithm: this is an artefact of the algorithm's tendency to report most frames as voiced, which leads to a small increase in OA given that MDB-melody-synth contains slightly more voiced frames compared to MedleyDB. Still, reporting most frames as voiced also heavily penalizes the algorithm (on both datasets), and despite the increase in OA the algorithm remains consistently ranked below Melodia and Bosch's algorithm in terms of OA. Indeed, the relative *ranking* of all three algorithms in terms of pitch and overall accuracy remains unchanged between MedleyDB and MDB-melody-synth, as shown in Figure 4.

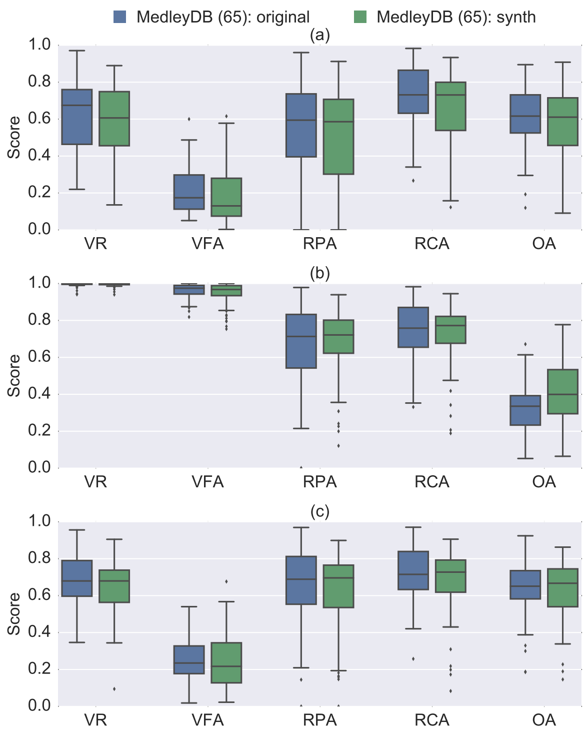


Figure 3. Melody extraction evaluation scores for 65 songs: (blue) original MedleyDB mixes and (green) MDB-melody-synth mixes. (a) Melodia, (b) Durrieu, (c) Bosch.

3.4 Multiple f_0 Estimation

As noted earlier, we use the Bach10 dataset [16] to evaluate the representativeness of the synthesized mixes resulting from our proposed methodology for multiple f_0 estimation. For this task 14 different metrics are computed in MIREX. It suffices to know that the first six measure "goodness" and go from 0 (worst) to 1 (best): Precision, Recall, Accuracy, and a chroma version (ignoring octave errors) for each, which we indicate with a "C_" prefix in our plots. The latter eight measure four different types of errors and their chroma counterparts, where 0 is the best score and greater values mean more errors. The reader is referred to [2, 32] for a detailed description of each metric. As before, all metrics are computed with `mir_eval`.

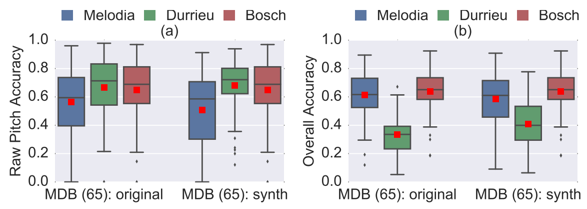


Figure 4. Evaluation scores for the three melody extraction algorithms on 65 MedleyDB and MDB-melody-synth mixes: (a) Raw Pitch Accuracy and (b) Overall Accuracy.

We use two multiple f_0 estimation algorithms for our evaluation: those by Benetos [3] and Duan [16]. The results are presented in Figure 5. For Benetos's method there is no statistically significant difference between Bach10 and Bach10-mf0-synth for any of the 14 metrics, and for Duan's there is no statistically significant difference for 10 of the 14 including the most important metrics such as Recall, Precision, Accuracy, and E_tot. The relative ranking of the two algorithms remains unchanged for all 14 metrics, as shown in Figure 6 subplots (a), (b), and (c) for Precision, Recall, and Accuracy respectively.

Since MedleyDB does not include multiple f_0 annotations, we cannot compare the performance of Benetos's and Duan's algorithms on MDB-mf0-synth to the original dataset as we did for MDB-melody-synth and Bach10-mf0-synth. In essence, MDB-mf0-synth is a completely new dataset for evaluating multiple f_0 estimation algorithms. The results obtained by Benetos's and Duan's algorithms for this new dataset are presented in Figure 7. We see that the performance of both algorithms drops considerably compared to the results they obtain on Bach10 (note the change in y-axis range), indicating that this new dataset is more challenging. The difference in performance between the two algorithms is smaller, and both seem to make an increased number of octave errors compared to Bach10, as indicated by the greater difference between the metrics and their chroma counterparts. The false alarm rate (E_fa) for both algorithms is also greater, which could be due to the greater proportion of tracks in MDB-mf0-synth with low polyphonies compared to Bach10, or due to the presence of percussive sources which are completely absent from the latter. Another interesting result is the significantly higher *variance* of all the metrics on MDB-mf0-synth compared to Bach10, which is likely due to the considerably greater variety in MDB-mf0-synth in terms of musical genre, instrumentation and polyphony. As an example of the performance analysis that can be done using MDB-mf0-synth, in Figure 8 we present the accuracy scores for the two algorithms broken down by polyphony. While it is beyond the scope of this paper, similar breakdowns could be performed by genre, instrumentation, vocal/instrumental, the presence/absence of percussion, etc.

4. DISCUSSION

We have proposed a methodology for the automatic f_0 annotation of polyphonic music by means of multitrack

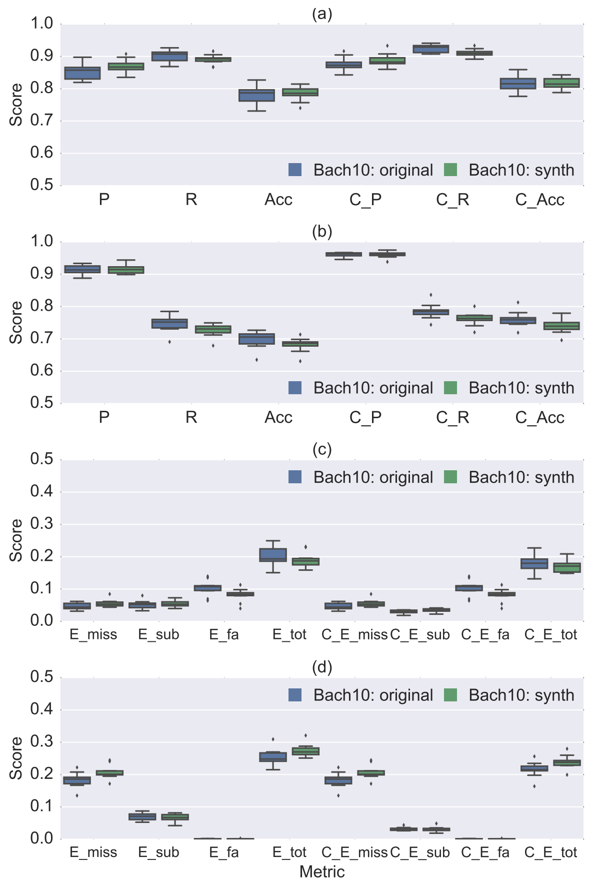


Figure 5. Multiple f_0 estimation scores on the Bach10 dataset, original mixes (blue) and synthesized mixes (green): (a) Benetos (b) Duan (c) Benetos errors (d) Duan errors. The chroma versions of each metric are indicated by a "C_" prefix.

datasets and an analysis/synthesis framework. We applied this methodology to create automatic f_0 annotations for melody extraction, bass line extraction and multiple f_0 estimation using the MedleyDB and Bach10 datasets. As noted in the introduction, these datasets can be used to train learning based f_0 estimation algorithms, as well as conduct controlled evaluation experiments. Furthermore, by means of a comparative evaluation we have shown that algorithms evaluated against the synthesized mixes and automatically generated f_0 annotations produce results that are, in almost all cases, equivalent (up to statistical significance) to those they produce for the original mixes. This suggests that in addition to providing insight from large-scale evaluation and facilitating multiple controlled evaluation breakdowns, the results are in fact quite representative (in terms of absolute scores) of the results we would have obtained by manually annotating the original mixes.

Since the proposed methodology is scalable and fully automatic, it can be readily applied to other existing multitrack datasets [1, 20, 22, 28, 37, 41], most of which were originally intended for source separation or automatic mixing evaluation. It can also be applied to datasets that provide separate melody and accompaniment tracks [11, 23].

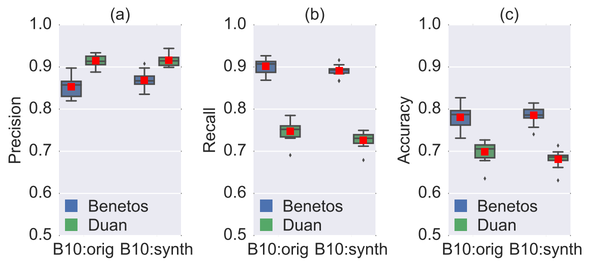


Figure 6. Multiple f_0 estimation scores for Duan's and Benetos's algorithms on Bach10 (B10:orig) and Bach10-mf0-synth: (a) Precision, (b) Recall and (c) Accuracy.

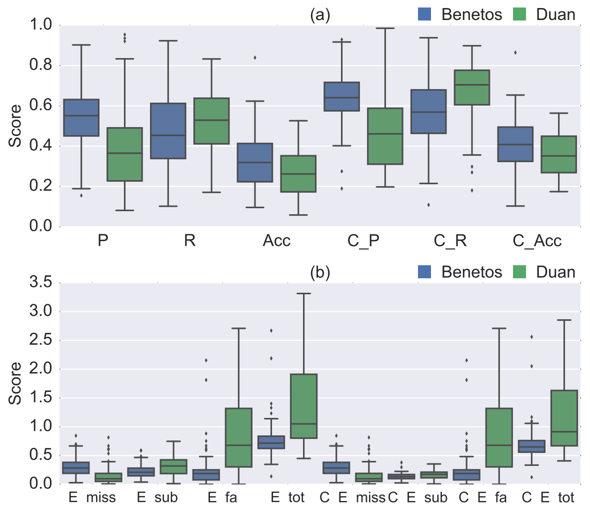


Figure 7. Evaluation scores for the multiple f_0 estimation algorithms by Benetos and Duan on the new MDB-mf0-synth dataset: (a) score metrics, (b) error metrics.

An important limitation of our methodology is that it can only be applied to monophonic stems, meaning it cannot be used to annotate polyphonic instruments such as the piano and the guitar. To address this, we are currently working on expanding the proposed framework by incorporating polyphonic transcription algorithms that can be applied in place of the monophonic pitch tracker for executing the first stage of the proposed framework on polyphonic stems. It can also be argued that since our approach requires generating new mixes (with a subset of the tracks replaced by synthesized versions), the resulting audio data do not reflect real-world data as reliably as the original mixes. While this is true, the results of our experiments suggest that the scores obtained using the synthesized datasets are in fact to a great extent representative of those one would obtain on the original mixes. Furthermore, since existing datasets for f_0 estimation in polyphonic music are so small, it is unlikely for the results obtained on these datasets to generalize to significantly larger audio collections, regardless of how they were annotated. We believe that the benefits of training and evaluating f_0 estimation algorithms on large-scale datasets with significantly greater variety in terms of audio content, enabled by our proposed framework, outweigh its limitations and have

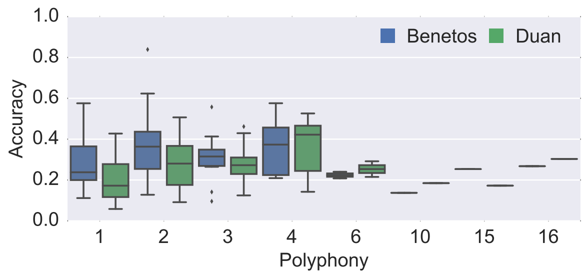


Figure 8. Accuracy scores for the algorithms by Benetos and Duan on MDB-mf0-synth, by polyphony.

the potential to lead to new insights and novel models for f_0 estimation in polyphonic music.

As research on analysis/synthesis algorithms and automatic mixing [34, 37, 38] advances, we can expect our framework to produce mixes that are increasingly authentic and true to the original mixes. The synthesis used in this study is purely harmonic, which affects the quality of the synthesis and could potentially affect the perception of note onsets (e.g., vocals with fricatives). We are currently expanding the framework to support harmonic+noise synthesis, and updated versions of the released datasets will be made available on the companion website. Still, it is important to highlight that the key contribution of this work is the proposed methodology itself, and our experimental results showing the representativeness of the mixes and annotations it produces. The value of this framework is precisely in the fact that we can use analysis and synthesis algorithms which, despite not being perfect, produce data of sufficient quality to be of value for MIR research. It means we can generate datasets whose size is only constrained by our (ever growing) access to multitrack recordings.

In a recent study [39], Su and Yang define four criteria for assessing the "goodness" of a dataset and its annotations for evaluating automatic music transcription (AMT) algorithms, which we summarize here: (1) Generality: the form, genre and instrumentation of the music excerpts should be representative of the music universe to which we expect the algorithm to generalize [6]; (2) Efficiency: the annotation process should be fast and scalable; (3) Cost: the cost of building the dataset, in terms of money and human resources, should be minimized. (4) Quality: the annotations should be accurate enough to facilitate correct evaluation of AMT algorithms. The methodology proposed in this paper satisfies these criteria to a great extent: since the generation of annotations only depends on the availability of multitrack data, it is relatively independent of (1) and can be applied to most musical genres. With regards to criteria (2), (3), and (4): since our methodology generates annotations completely automatically, one could argue that it is as efficient as any annotation technique could possibly be. For the same reason, it is also very cost efficient, since creating annotations is essentially free. Finally, the quality of the annotations is guaranteed since the synthesized tracks match the annotations perfectly.

[6] For a detailed discussion of these considerations see [40].

5. REFERENCES

[1] J. Abeßer, O. Lartillot, C. Dittmar, T. Eerola, and G. Schuller. Modeling musical attributes to characterize ensemble recordings using rhythmic audio features. In *IEEE ICASSP*, pages 189–192, May. 2011.

[2] M. Bay, A. Ehmann, and J. S. Downie. Evaluation of multiple-F0 estimation and tracking systems. In *10th Int. Soc. for Music Info. Retrieval Conf.*, pages 315–320, Kobe, Japan, Oct. 2009.

[3] E. Benetos, S. Cherla, and T. Weyde. An efficient shift-invariant model for polyphonic music transcription. In *6th Int. Workshop on Machine Learning and Music*, pages 1–4, Prague, Czech Republic, Sep. 2013.

[4] 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.*, pages 591–596, Miami, USA, Oct. 2011.

[5] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello. Deep salience representations for f_0 estimation in polyphonic music. In *18th Int. Soc. for Music Info. Retrieval Conf.*, Suzhou, China, Oct. 2017.

[6] R. M. Bittner, J. Salamon, S. Essid, and J. P. Bello. Melody extraction by contour classification. In *16th Int. Soc. for Music Info. Retrieval Conf.*, Malaga, Spain, Oct. 2015.

[7] 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 Int. Soc. for Music Info. Retrieval Conf.*, pages 155–160, Taipei, Taiwan, Oct. 2014.

[8] J. Bonada. Wide-band harmonic sinusoidal modeling. In *11th Int. Conf. on Digital Audio Effects (DAFx-08)*, pages 265–272, Espoo, Finland, Sep. 2008.

[9] J. J. Bosch, R. M. Bittner, J. Salamon, and E. Gómez. A comparison of melody extraction methods based on source-filter modelling. In *17th Int. Soc. for Music Info. Retrieval Conf.*, New York City, USA, Aug. 2016.

[10] P. Cano, E. Gómez, F. Gouyon, P. Herrera, M. Koppenberger, B. Ong, X. Serra, S. Streich, and N. Wack. ISMIR 2004 audio description contest. Technical report, Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain, Apr. 2006.

[11] T.-S. Chan, T.-C. Yeh, Z.-C. Fan, H.-W. Chen, L. Su, Y.-H. Yang, and R. Jang. Vocal activity informed singing voice separation with the iKala dataset. In *IEEE ICASSP*, pages 718–722, 2015.

[12] S. F. Chen, B. Kingsbury, Lidia Mangu, D. Povey, G. Saon, H. Soltau, and G. Zweig. Advances in speech transcription at IBM under the DARPA EARS program. *IEEE Trans. on Audio, Speech, and Language Processing*, 14(5):1596–1608, Sep. 2006.

[13] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am.*, 111(4):1917–1930, Apr. 2002.

[14] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F.-F. Li. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, Miami, FL, USA, Jun. 2009.

[15] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, Jul. 2008.

[16] Z. Duan, B. Pardo, and C. Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(8):2121–2133, 2010.

[17] Jean-Louis Durrieu, Gaël Richard, Bertrand David, and Cédric Févotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE TASLP*, 18(3):564–575, March 2010.

[18] V. Emiya, R. Badeau, and B. David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE TASLP*, 18(6):1643–1654, 2010.

[19] S. Ewert, M. Müller, and P. Grosche. High resolution audio synchronization using chroma onset features. In *ICASSP*, pages 1869–1872, Taipei, Taiwan, Apr. 2009.

[20] J. Fritsch. High quality musical audio source separation. Master's thesis, UPMC / IRCAM / Telecom ParisTech, 2012.

[21] E. Gómez and J. Bonada. Towards computer-assisted flamenco transcription: An experimental comparison of automatic transcription algorithms from a cappella singing. *Computer Music journal*, 37(2):73–90, 2013.

[22] S. Hargreaves, A. Klapuri, and M. Sandler. Structural segmentation of multitrack audio. *IEEE TASLP*, 20(10):2637–2647, 2012.

[23] C.-L. Hsu and J.-S.R. Jang. On the improvement of singing voice separation for monaural recordings using the MIR-1K dataset. *IEEE Trans. on Audio, Speech, and Language Processing*, 18(2):310–319, Feb. 2010.

[24] A. Klapuri. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Trans. on Speech and Audio Processing*, 11(6):804–816, Nov. 2003.

[25] A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[26] G. Lafay, M. Lagrange, M. Rossignol, E. Benetos, and A. Röbel. A morphological model for simulating acoustic scenes and its application to sound event detection. *IEEE/ACM Trans. on Audio, Speech, and Lang. Proc.*, 24(10):1854–1864, Oct. 2016.

[27] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent HMMs trained on synthesized audio. *IEEE Trans. on Audio, Speech, and Language Processing*, 16(2):291–301, Feb. 2008.

[28] A. Liutkus, R. Badeau, and G. Richard. Gaussian processes for underdetermined source separation. *IEEE Trans. on Signal Processing*, 59(7):3155–3167, 2011.

[29] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. P. Bello, and S. Dixon. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. In *TENOR*, Paris, France, 2015.

[30] M. Mauch and S. Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663, Florence, Italy, May 2014.

[31] B. McFee, E.J. Humphrey, and J.P. Bello. A software framework for musical data augmentation. In *16th Int. Soc. for Music Info. Retrieval Conf.*, pages 248–254, Malaga, Spain, Oct. 2015.

[32] G. E. Poliner and D. P. W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1):154–154, 2007.

[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 *15th ISMIR*, pages 367–372, Taipei, Taiwan, 2014.

[34] J.D. Reiss. Intelligent systems for mixing multichannel audio. In *17th Int. Conf. on Digital Signal Processing*, pages 1–6, Corfu, Greece, Jul. 2011.

[35] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, Aug. 2012.

[36] J. Salamon, E. Gómez, D. P. W. Ellis, and G. Richard. Melody extraction from polyphonic music signals: Approaches, applications and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, Mar. 2014.

[37] J. Scott and Y. E. Kim. Instrument identification informed multi-track mixing. In *14th Int. Soc. for Music Info. Retrieval Conf.*, pages 305–310, Nov. 2013.

[38] J. Scott, M. Prockup, E. M. Schmidt, and Y. E. Kim. Automatic multi-track mixing using linear dynamical systems. In *8th SMC Conf.*, Jul. 2011.

[39] L. Su and Y.-H. Yang. Escaping from the abyss of manual annotation: New methodology of building polyphonic datasets for automatic music transcription. In *CMMR*, pages 221–233, Plymouth, UK, Jun. 2015.

[40] J. Urbano, M. Schedl, and X. Serra. Evaluation in music information retrieval. *J. of Intelligent Info. Systems*, 41:345–369, 2013.

[41] E. Vincent, S. Araki, F. Theis, G. Nolte, P. Bofill, H. Sawada, A. Ozerov, V. Gowreesunker, D. Lutter, and N. Q. K. Duong. The signal separation evaluation campaign (2007–2010): Achievements and remaining challenges. *Signal Processing*, 92(8):1928–1936, Aug. 2012.

MAKE YOUR OWN ACCOMPANIMENT: ADAPTING FULL-MIX RECORDINGS TO MATCH SOLO-ONLY USER RECORDINGS

TJ Tsai[1] **Steven K. Tjoa**[2] **Meinard Müller**[3]
[1]Harvey Mudd College, Claremont, CA
[2]Galvanize, Inc., San Francisco, CA
[3]International Audio Laboratories Erlangen, Erlangen, Germany
ttsai@hmc.edu, steve@stevetjoa.com, meinard.mueller@audiolabs-erlangen.de

ABSTRACT

We explore the task of generating an accompaniment track for a musician playing the solo part of a known piece. Unlike previous work in real-time accompaniment, we focus on generating the accompaniment track in an off-line fashion by adapting a full-mix recording (e.g. a professional CD recording or Youtube video) to match the user's tempo preferences. The input to the system is a set of recorded passages of a solo part played by the user (e.g. solo part in a violin concerto). These recordings are contiguous segments of music where the soloist part is active. Based on this input, the system identifies the corresponding passages within a full-mix recording of the same piece (i.e. contains both solo and accompaniment parts), and these passages are temporally warped to run synchronously to the solo-only recordings. The warped passages can serve as accompaniment tracks for the user to play along with at a tempo that matches his or her ability or desired interpretation. As the main technical contribution, we introduce a segmental dynamic time warping algorithm that simultaneously solves both the passage identification and alignment problems. We demonstrate the effectiveness of the proposed system on a pilot data set for classical violin.

1. INTRODUCTION

Ima Amateur loves her recording of Itzhak Perlman performing the Tchaikovsky violin concerto with the London Symphony Orchestra. She has been learning how to play the first movement herself, and she would love to play along with the recording. Unfortunately, there are parts of the recording that are simply too fast for her to play along with. She finds an app that can slow down the parts of the Perlman recording that are difficult. All she has to do is upload several solo recordings of herself performing sections of the concerto, along with the original full-mix recording that she would like to play along with. The app analyzes her playing and generates a modified version of the Perlman recording that runs in sync with her solo recordings.

This paper explores the technical feasibility of such an application. In technical terms, the problem is this: given a full-mix recording and an ordered set of solo-only recordings that each contain a contiguous segment of music where the soloist is active, design a system that can time-scale modify the full-mix recording to run synchronously with the solo recordings. [1]

There are three main technical challenges underlying this scenario. The first challenge is to identify the passages in the full-mix recording that correspond to the solo-only recordings. The second challenge is to temporally align the corresponding passages in the full-mix and solo recordings. The third challenge is to time-scale modify the full-mix recording to follow the calculated alignment without changing the pitch of the original recording. This paper focuses primarily on the first two challenges, and it assesses the technical feasibility of solving these problems on a pilot data set. The main technical contribution of this work is to propose a segmental dynamic time warping (DTW) algorithm that simultaneously solves the passage identification and temporal alignment problems. We will simply adopt an out-the-box approach to solve the third challenge.

The idea of generating accompaniment for amateur musicians has been explored in two different directions. On one end of the spectrum, companies have explored fixed accompaniment tracks. Some examples include the popular Aebersold Play-A-Long recordings for jazz improvisation and Music Minus One for classical music. The benefit of fixed accompaniment tracks is their simplicity – all you need is a device that can play audio. The drawback of fixed accompaniment tracks is their lack of adaptivity – they do not respond or adapt to the user's playing in any way. On the other end of the spectrum, academics have explored real-time accompaniment (e.g. see work by Raphael [23] [24] and Cont [3]). These are complex systems that can track a musician's (or group's) playing and generate accompaniment in real-time. The benefit of real-time accompaniment is the adaptivity of the system. The drawbacks of real-time accompaniment systems are that they are not easy to use for the general population (e.g. require software packages on a laptop) and may not be very

[1] Without changing the pitch, of course!

expressive (e.g. sound like MIDI). Also, for the purposes of academic study, another drawback is the difficulty of evaluating such a system in an objective way. Because the user and the accompaniment system influence each other in real-time, it is difficult to decouple the effect of one from the other. When there are errors, for example, it is difficult to say whether the error is because the accompaniment system failed, the user failed to respond appropriately, or some combination of both.

This work explores the realm in between these two extremes. Like fixed accompaniment tracks, the proposed system has the benefit of simplicity – the user does not need any specialized software or hardware, but simply receives an audio track that can be played on any audio device. Like real-time accompaniment, the proposed system has the benefit of (partial) adaptivity – the system tailors the accompaniment track to the user's playing in an off-line manner. This middle realm has several additional benefits. Because the user and the accompaniment are no longer coupled in real-time, we can measure how well the accompaniment system "follows" the user's playing with objective metrics. Another benefit is that the off-line nature of this system makes it suitable for a client-server model, which is ideal for the envisioned app. Lastly, by approaching this problem through adapting an existing recording, we can also potentially get the benefit of a very musical and expressive accompaniment track (assuming we don't introduce too many artifacts from time-scale modification).

The two challenges we will focus on – passage identification and temporal alignment – are closely related to previous work in audio matching and music synchronization. The passage identification problem has strong similarities to audio matching, where the goal is to identify a given passage in other performances of the same (usually classical) work. Previous work has introduced robust features for this task [20] and efficient ways to handle global tempo variations such as using multiple versions of a query that have been tempo-adjusted [19]. Subsequent work has explored the use of indexing techniques to scale the system to large data sets [14] [2]. The temporal alignment problem has strong similarities to music synchronization, where the goal is to temporally align two performances of the same piece. The bread-and-butter approach is to apply DTW with suitably designed features [12] [4] [10]. One problem with this approach is that the memory and computation requirements increase quadratically as the feature sequences increase in length. Many variants have been proposed to mitigate this issue, including limiting the search space to a band [25] or parallelogram [13] around the cost matrix diagonal, doing the time-warping in an online fashion [5] [15], or adopting a multiscale approach that estimates the alignment at different granularities [26] [21] [9]. Other variants tackle issues like handling repeats [11], identifying partial alignments between recordings [17] [18], dealing with memory constraints [22], and taking advantage of multiple recordings [27] [1].

Though similar, the proposed scenario differs from most previous work in three important ways. First, we are matching solo-only recordings to full-mix recordings (i.e. solo and accompaniment). Most work in audio matching and music synchronization assumes that the recordings of interest are different performances of the same piece, and therefore have the same audio sources. One could think of the current scenario as audio matching with very high levels of additive noise (i.e. the accompaniment). Second, the task is off-line but there are still stringent runtime constraints. In music synchronization, the best approach is the one with the highest alignment precision, and we are willing to accept significant runtimes since the task is off-line. In the current scenario, however, the runtime is a very important factor because the application is user-facing. A user will not be willing to wait 30 seconds for the accompaniment track to be generated. For this reason, in this paper we will not consider any approaches to these two challenges that require more than 5-6 seconds of runtime. Third, the current scenario deals with consumer-produced recordings. Much previous work focuses on album tracks from professional CDs and professional musicians. In contrast to this, amateur musicians will play wrong notes, count incorrectly, rush, and play out of tune. These issues will be important factors affecting system performance.

This paper is structured around our main goal: to assess the technical feasibility of solving the passage identification and temporal alignment problems in a robust and efficient manner. Section 2 describes our system, including an explanation of the proposed segmental DTW algorithm. Section 3 discusses the experimental setup. Section 4 presents empirical results of our experiments on the pilot data set. Section 5 investigates several questions of interest to gain more intuition into system performance. Section 6 concludes the work.

2. SYSTEM DESCRIPTION

We describe the proposed system in three parts: the segmental DTW algorithm, the features, and the time-scale modification.

2.1 Segmental DTW Algorithm

There are four main steps in the segmental DTW algorithm, each explained below.

Step 1: Frame-level cost matrices. The first step is to compute a subsequence DTW cumulative cost matrix for each solo segment. Subsequence DTW is a variant of the regular DTW algorithm in which one of the recordings (the query) is assumed to only match a *section* of the other recording (the reference), rather than matching the entire recording from beginning to end. This can be accomplished by allowing the query to *begin* matching anywhere in the reference without penalty, and allowing the query to *end* matching anywhere in the reference without penalty. We allow the following $(query, reference)$ steps in the dynamic programming stage: $(1,1)$, $(1,2)$, and $(2,1)$. These steps have weights of 1, 1, and 2, respec-

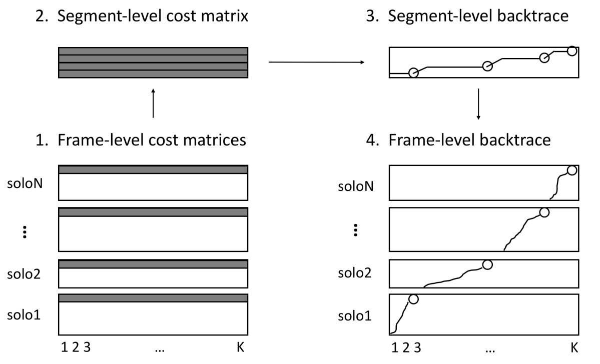

2. Segment-level cost matrix
3. Segment-level backtrace
1. Frame-level cost matrices
4. Frame-level backtrace
soloN
⋮
solo2
solo1
1 2 3 ... K

Figure 1. A graphical overview of the segmental DTW algorithm for aligning an ordered set of solo recordings against a full-mix recording. Rows correspond to solo recording frames and columns correspond to full-mix recording frames. Time increases from bottom to top and left to right. In this example, $N = 4$.

tively.[2] This set of steps assumes that the instantaneous tempo in the query and reference will differ at most by a factor of 2. For more details about subsequence DTW, see chapter 7 in [16]. In the case of our proposed algorithm, we compute the subsequence DTW cumulative cost matrix but refrain from backtracing until step 4. Rather than backtracing from the local optimum in each cumulative cost matrix, we will instead backtrace from the element on the globally optimum path. This globally optimum path will be determined in steps 2 and 3.

Step 2: Segment-level cost matrix. The second step is to compute a cumulative cost matrix of global path scores across all solo segments. This can be done in two sub-steps. The first sub-step is to create a matrix that contains the last row of each subsequence cumulative cost matrix from step 1.[3] This matrix will have N rows and K columns, where N is the number of solo segments and K is the number of frames in the reference (i.e. full-mix) recording. Note that this matrix is analogous to a pairwise cost matrix, where instead of pairwise frame-level costs we have segment-level subsequence path costs. The second sub-step is to compute a (segment-level) cumulative cost matrix on this (segment-level) pairwise cost matrix by doing dynamic programming. This dynamic programming step differs from regular DTW dynamic programming in one important way. Unlike most scenarios where the set of possible transitions is fixed regardless of position in the cost matrix, here the possible transition steps changes from row to row. Specifically, for an element in row n, the two possible transitions are $(0, 1)$ and $(1, \frac{L_{n+1}}{2})$, where L_{n+1} is the length (in frames) of the $(n+1)^{\text{th}}$ solo segment. The weights on these two transitions are 0 and 1, respectively. In words, we are looking for the N elements in the segment-level pairwise cost matrix (one per row) that have the minimum total path score under two constraints: (1) they are consistent with the given ordering (i.e. segment n comes before segment $n+1$), and (2) elements in adjacent rows must be separated by a minimum distance, which is determined by the length of the solo segment and the maximum tempo difference in the subsequence DTW step (in this case, a factor of 2).

Step 3: Segment-level backtrace. The third step is to backtrace through the segment-level cumulative cost matrix. We start at the last element of the matrix (i.e. the upper right hand corner) and backtrace until we reach the first element of the matrix (i.e. the lower left hand corner). Note that the $(0, 1)$ steps with 0 weight allow for skipping portions of the full-mix recording without penalty. The $(1, \frac{L_{n+1}}{2})$ transitions in the backtraced path indicate the element in each row that contributes to the globally optimal path.

Step 4: Frame-level backtrace. The final step is to backtrace through each subsequence DTW cumulative cost matrix from step 1, where we begin the backtracing at the elements selected in step 3. These elements have been selected to optimize a global path score across all solo segments, rather than a local path score across a single solo segment. After performing this frame-level backtrace step, we have achieved our desired goal: identifying both segment-level and frame-level alignments for each solo segment.

Figure 1 shows a graphical summary of the segmental DTW algorithm. In this figure, rows correspond to different solo segment frames and columns correspond to different full-mix frames. Time increases from bottom to top and from left to right. The four rectangles in the lower left are the frame-level cumulative cost matrices for each solo recording. The segment-level cost matrix (top left) is constructed by aggregating the last row from each frame-level cumulative cost matrix (highlighted in dark gray). We then backtrace at the segment level, and use the predicted segment ending points to backtrace at the frame level. The final predicted alignments are shown in the lower right. Note that the proposed system only indicates how the full-mix recording should be warped during the segments of the piece when the soloist is playing. One could interpolate the tempo for the other segments.

2.2 Features

The segmental DTW algorithm is compatible with any frame-based feature and cost metric. For the experiments in this paper, we computed L2-normalized chroma features every 22 ms and used a cosine distance metric. This combination was selected for two practical reasons. First, we wanted to demonstrate the segmental DTW algorithm with a standard feature, so as not to conflate the performance benefits of both a new matching algorithm and a novel (or less widely used) feature. Second, this combination allows the subsequence DTW cost matrices to be computed very efficiently with simple matrix multiplication. Given the constraints on runtime of this consumer-facing application, efficiency is an important consideration. We selected the feature rate to ensure that the average time required to

[2] Note that the $(2, 1)$ step should be weighted double to prevent degenerate matchings to very short sections.

[3] Here, we assume that rows correspond to different query frames, and columns correspond to different reference frames.

Composition	full	solo	avgLen	segs
Seitz concerto no2, mv3	5	5	187 s	5
Bach double concerto, mv1	5	5	250 s	5
Vivaldi concerto in a, mv1	5	5	229 s	5
Veracini sonata in d, mv4	3	4	223 s	4

Table 1. Summary of the pilot data set. Each row indicates the number of full-mix and solo recordings, the average length, and the number of segments in the composition.

align a single query (i.e. multiple solo recordings against a full-mix recording) was under 6 seconds. This threshold could be set arbitrarily depending on how long we are willing to make the user wait. In the discussion section, we will compare our main results with a system that uses more state-of-the-art features, which were developed in an off-line context where runtime is not a significant consideration. These latter features can provide a lower bound on error rate when we ignore runtime constraints.

2.3 Time-Scale Modification

The goal of the time-scale modification (TSM) step is to stretch or compress the duration of a given audio signal while preserving properties like pitch and timbre. Typically, TSM approaches stretch or compress an audio signal in a linear fashion by a constant stretching factor. In our scenario, we need to stretch the full-mix recording according to the solo-mix alignment, which leads to a non-linear time-stretch function. To deal with non-linear stretches, we apply the strategy described in [8], where the positions of the TSM analysis frames are specified according to the time-stretch function instead of a constant analysis hopsize.

To attenuate artifacts and to improve the quality of the time-scale modified signal, we use a recent TSM approach [7] that involves harmonic-percussive separation and combines the advantages of a phase-vocoder TSM approach (preserving the perceptual quality of harmonic signal components) and a time-domain TSM approach (preserving transient-like percussive signal components). An overview of different TSM procedures can be found in [6, 8].

3. EXPERIMENTAL SETUP

The experimental setup will be described in three parts: the data collection, the data preparation, and the evaluation metric.

3.1 Data Collection

Our data collection process was dictated by practicality. In order to evaluate the proposed system, we need two different types of audio data: full-mix recordings and solo recordings. Clearly, the full-mix recordings are in abundant supply and can be selected from any professional CD recording or Youtube video. The solo recordings, however, are much more difficult to find, as musicians typically do not record performances that are missing the accompaniment part. Our solution to this problem was to focus data collection efforts on a small subset of pieces from the highly popular Suzuki violin method. The Suzuki method prescribes a specific sequence of violin works in order to develop a violinist's mastery of the instrument. Because of the popularity of the Suzuki method, we were able to find Youtube videos of violinists performing the solo parts (in isolation) from several works. Some of these recordings are violin teachers demonstrating how to perform a piece. Some recordings are young adults wishing to document their progress on the violin. Other recordings are doting parents trying to show off their talented children.

Table 1 shows a summary of the audio recordings. The data set contains four violin pieces or movements selected from Suzuki books five and six. For each piece, we collected multiple full-mix recordings and solo recordings from Youtube. By focusing on annotating multiple recordings of the same piece, we can make the most of the limited amount of (annotated) data by considering different combinations of full-mix and solo recordings. At the same time, we wanted several pieces of music from different composers and periods, so as to avoid a composer-specific bias. The recordings range in length from 161 to 325 seconds, and they range in quality from cell phone videos to professionally recorded performances. All audio tracks were converted to mono wav format with 22050 Hz sampling rate. In total, there is approximately 2 hours and 20 minutes of annotated audio data.

3.2 Data Preparation

Once the audio data was collected, there were two additional steps needed to prepare the data for use in our experiments.

The first preparation step was to generate beat-level annotations. The annotations were done in SonicVisualizer [4] by three different individuals with extensive training in classical piano. We kept only those beats that had two or more independent annotations, and we use the mean annotated time as the ground truth.

The second data preparation step was to divide the solo recordings into segments. Recall that the input to the system is a set of contiguous segments of music where the soloist is active. Each segment is specified by a pair of unique identifiers (e.g. start at measure 5 beat 1 and end at measure 37 beat 4), and the segments are non-overlapping. For each composition, we manually selected segments by identifying natural breakpoints where a violinist would likely end a segment, such as section boundaries or the start/end of a long rest.

We can summarize the prepared data set as follows. Each query in the benchmark is a pairing of a full-mix recording and a solo recording (i.e. the 4-5 segments from a solo recording). There are thus a total of 87 queries in the benchmark. This is clearly not a large data set. It is meant to serve as a pilot data set to assess the feasibility of the proposed system.

[4] http://www.sonicvisualiser.org/

tolerance	global	subseq	segmental
1s	40.2%	8.4%	2.2%
2s	20.2%	6.1%	0.0%
5s	14.9%	6.1%	0.0%
10s	9.3%	6.1%	0.0%

Table 2. Boundary prediction error rates for global, subsequence, and segmental DTW algorithms. Each entry indicates the percentage of predicted boundary points that are incorrect at a specified allowable error tolerance.

3.3 Evaluation Metric

In this paper, we will focus only on the aspects of the system that can be evaluated objectively: the segment boundaries and frame-level alignments. To evaluate segment boundary predictions, we compare the predicted and ground truth boundary points for each solo segment, and then determine what fraction of predicted boundary points are correct (or incorrect) for a given allowable error tolerance. To evaluate frame-level alignments, we compare predicted and ground truth timestamps in the full-mix recording that correspond to the annotated beat locations in the solo segments. [5] We then determine what fraction of alignments are correct (or incorrect) for a given allowable error tolerance. By considering a range of different error tolerances, we can determine an error tolerance curve. Note that the error tolerances for the segment boundary metric are much larger than the error tolerances for frame alignment, since the former is measuring retrieval at the segment level.

4. RESULTS

To assess the effectiveness of the proposed segmental DTW algorithm, we compared its performance against two other baseline systems. The first baseline system is to simply concatenate all of the solo audio segments and perform a single global DTW against the full-mix recording. For this baseline system, we use transition steps $(0, 1)$, $(1, 0)$, and $(1, 1)$ in order to handle the discontinuities between solo segments. All steps are given equal weight. The second baseline system is to perform subsequence DTW on each solo segment independently, where the best locally optimal path in each cost matrix is taken as the predicted segment-level and frame-level alignment. In order to make the comparison between systems fair, all three systems use the same chroma features. Any differences in performance should thus reflect the effectiveness of the matching algorithm.

Table 2 compares the performance of the three systems on passage identification. The rows in the table show the percentage of predicted boundary points that are incorrect at four different error tolerances. The three rightmost columns compare the performance of the global DTW baseline ('global'), the subsequence DTW baseline ('subseq'), and the proposed segmental DTW algorithm ('segmental').

[5] Since the annotated beat locations generally fall between frames, we use simple linear interpolation between the nearest predicted alignments.

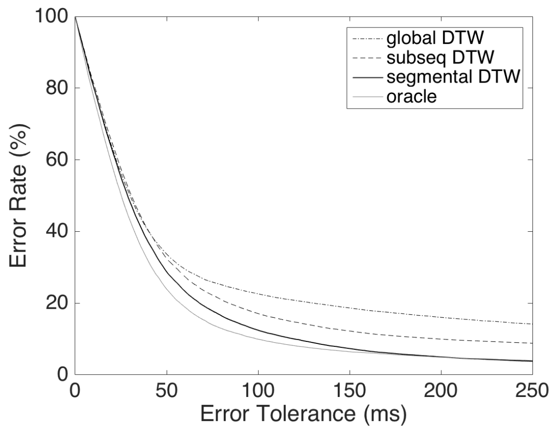


Figure 2. Error tolerance curves for the global, subsequence, and segmental DTW algorithms. Each point on a curve indicates the percentage of predicted beat alignments that are incorrect for a given error tolerance. An additional curve is shown for an oracle system, which provides a lower bound on performance.

There are three things to notice about Table 2. First, the error rates clearly decrease from left to right. Thus, the relative performance of the three algorithms is clear: global DTW performs worst, subsequence DTW performs better, and segmental DTW performs best. Second, subsequence DTW reaches an asymptotic error rate of 6.1%. These errors are passages that the subsequence DTW algorithm is matching incorrectly because it fails to take into account the temporal ordering of the solo segments. For example, it incorrectly matches the main theme to the recapitulation or matches repeated segments to the wrong repetition. Better features are unlikely to fix these errors. Third, the segmental DTW algorithm has perfect performance for error tolerances of 2 seconds and above. This suggests that the 2.2% of errors at a 1 second error tolerance are an indication of poor alignments but correctly identified passages. We will investigate these errors in the discussion section.

Figure 2 compares the performance of the three systems on temporal alignments. The figure shows the error tolerance curves for error tolerances ranging from 0 to 250 ms. Each point on a curve indicates the percentage of predicted beat timestamps that are incorrect at a given error tolerance. There is also a curve for an oracle system, which will be explained in section 5.2.

There are three things to notice about Figure 2. First, the curves are identical for error tolerances < 25 ms. This indicates that when an algorithm is "locked onto" a signal, the limit to its precision is the same for all three algorithms. This is what we expect, since all three algorithms are based on the same fundamental dynamic programming approach and use the same features. This is a realm where the segmental DTW algorithm does not help, but where better features are needed to improve performance. Second, the curves begin to diverge significantly for error tolerances > 50 ms. This is a realm where the segmental DTW algorithm provides significant benefit to system performance. For example, at 100 ms error tolerance, the segmental DTW algorithm improves the error rate from 22.6% and 17.1% to 12.4%. Third, the curves do not intersect. In other words, the segmental DTW algorithm provides unilateral benefit across all error tolerances.

5. DISCUSSION

In this section, we investigate three questions of interest that will give deeper insight into system performance.

5.1 Investigation of Boundary Errors

The first question of interest is: "What is causing the segment boundary errors?" We saw from Table 2 that 2.2% of predicted segment boundaries are incorrect at an error tolerance of 1 second. We investigated all of these errors to determine the root cause of the problem.

There are three main observations we can make from our investigations of segment boundary errors. First, most segment boundary errors are a result of a mistake on the part of the musician. In one instance, the violin player messes up and stops playing for 3-4 beats at the end of a phrase. In another instance, the group is very out of sync on the last note. These two specific mistakes caused more than 50% of the segment boundary errors, since a single mistake will cause errors on all of the queries that contain the recording. Second, the maximum tempo ratio of 2x imposed by the DTW step sizes causes errors when the instantaneous tempo difference is extreme. For example, one recording has a very pronounced rubato at the end of the piece, which causes problems when the recording is paired with a performance that has very little rubato at the end. Third, all of the segment boundary errors were predictions of the end of a segment. The DTW algorithm (and its variants) do well in smoothing out errors in the beginning and middle of segments, but it often fails at the end of a segment because there is no signal "on the other side" to smooth out the prediction.

5.2 Lower Bound on Error Rate

The second question of interest is: "What is the lower bound on error rate?" In other words, what is the best error rate that we could hope to achieve given a current state-of-the-art alignment system? In order to answer this question, we ran an experiment with two major changes. The first change is that we assume this system is an oracle and knows the ground truth segment boundaries for each solo segment. The second change is that we use an alignment system [22] that was designed to maximize alignment precision in an off-line context. Note that this oracle system requires more than 45 sec on average to align each query (i.e. align multiple solo recordings against a full-mix recording), so it would not be suitable given the runtime constraints of our user-facing application. (In contrast, our proposed system required an average of 5.20 sec.) Thus, we can interpret the performance of the oracle system as a lower bound on error rate when runtime constraints are ignored.

The performance of this oracle system is shown in Figure 2 (overlaid on the same figure from the results section). There are two things to point out about this lower bound curve. First, the proposed system approximately achieves the lower bound for error tolerances > 175 ms. Second, the lower bound shows the most room for improvement in the 50 to 100 ms error tolerance range. For a 75 ms error tolerance, the proposed system and oracle system achieve error rates of 17.8% and 14.0%, respectively.

5.3 Listening to the Accompaniment Track

The third question of interest is: "How does the time-scale modified accompaniment track actually sound?" One useful way we can get a sense of how well the accompaniment is "following" the solo recordings is to create a stereo track in which one channel contains the unchanged solo recording and the other channel contains the time-stretched accompaniment track. By listening to both tracks simultaneously, we can gain an intuitive sense of how well the system is doing. We have posted several samples of these stereo recordings for interested readers. [6]

There are three qualitative observations we can make regarding these informal listening tests. First, the system performs much more erratically when the solo part is not dominant. This was particularly a problem for the Bach double violin concerto since there are two equally important violin parts. When the 2^{nd} violin part is dominant, the accompaniment track has significantly more time-warping artifacts. Second, the system handles rapid notes very well and prolonged notes very poorly. When the solo part is holding a single long note, the accompaniment track would sometimes have very severe temporal distortion artifacts. Third, the time-stretched accompaniment track often has a "jerky" tempo, especially when the solo part has a prolonged note. The accompaniment track is clearly tracking the solo recordings, but it often has short, sudden bursts of tempo speedups and slowdowns. One way to address this issue would be to do some type of temporal smoothing of the predicted alignment.

6. CONCLUSION

We have described a system that time-scale modifies an existing full-mix recording to run synchronously to an ordered set of solo-only user recordings of the same piece. We propose a segmental DTW algorithm that simultaneously solves the passage identification and temporal alignment problems, and we demonstrate the benefit of this algorithm over two other baseline systems on a pilot data set of classical violin music. Areas of future work include expanding the pilot data set, exploring features that are both computationally efficient and well-suited to the asymmetric nature of the scenario, and investigating pre-processing steps for solo detection and separation.

[6] `http://pages.hmc.edu/ttsai`

7. ACKNOWLEDGMENTS

Thanks to Zhepei Wang and Thitaree Tanprasert for helping with the data annotation. The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

8. REFERENCES

[1] Andreas Arzt and Gerhard Widmer. Real-time music tracking using multiple performances as a reference. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 357–363, Málaga, Spain, 2015.

[2] Michael A. Casey, Christophe Rhodes, and Malcolm Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015–1028, 2008.

[3] Arshia Cont, José Echeveste, and Jean-Louis Giavitto. The Cyber-Physical System Approach for Automatic Music Accompaniment in Antescofo. In *Acoustical Society Of America*, Providence, Rhode Island, United States, May 2014.

[4] Roger B. Dannenberg and Ning Hu. Polyphonic audio matching for score following and intelligent audio editors. In *Proc. of the International Computer Music Conference (ICMC)*, pages 27–34, San Francisco, USA, 2003.

[5] Simon Dixon. Live tracking of musical performances using on-line time warping. In *Proc. of the 8th International Conference on Digital Audio Effects*, pages 92–97. Citeseer, 2005.

[6] Mark Dolson and Jean Laroche. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing*, 7(3):323–332, 1999.

[7] Jonathan Driedger and Meinard Müller. Improving time-scale modification of music signals using harmonic-percussive separation. *IEEE Signal Processing Letters*, 21(1):105–109, 2014.

[8] Jonathan Driedger and Meinard Müller. A review on time-scale modification of music signals. *Applied Sciences*, 6(2):57–82, February 2016.

[9] Sebastian Ewert and Meinard Müller. Refinement strategies for music synchronization. In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval (CMMR)*, volume 5493 of *Lecture Notes in Computer Science*, pages 147–165, Copenhagen, Denmark, May 2008.

[10] Sebastian Ewert, Meinard Müller, and Peter Grosche. High resolution audio synchronization using chroma onset features. In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1869–1872, Taipei, Taiwan, April 2009.

[11] Christian Fremerey, Meinard Müller, and Michael Clausen. Handling repeats and jumps in score-performance synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 243–248, Utrecht, The Netherlands, 2010.

[12] Ning Hu, Roger B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, NY, USA, 2003.

[13] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

[14] Frank Kurth and Meinard Müller. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395, February 2008.

[15] Robert Macrae and Simon Dixon. Accurate real-time windowed time warping. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 423–428, 2010.

[16] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[17] Meinard Müller and Daniel Appelt. Path-constrained partial music synchronization. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 65–68, Las Vegas, Nevada, USA, April 2008.

[18] Meinard Müller and Sebastian Ewert. Joint structure analysis with applications to music annotation and synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 389–394, Philadelphia, Pennsylvania, USA, September 2008.

[19] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 288–295, London, UK, 2005.

[20] Meinard Müller, Frank Kurth, and Michael Clausen. Chroma-based statistical audio features for audio matching. In *Proc. of the Workshop on Applications of Signal Processing (WASPAA)*, pages 275–278, New Paltz, New York, USA, October 2005.

[21] Meinard Müller, Henning Mattes, and Frank Kurth. An efficient multiscale approach to audio synchronization. In *Proc. of the International Conference on Music Information Retrieval (ISMIR)*, pages 192–197, Victoria, Canada, October 2006.

[22] Thomas Prätzlich, Jonathan Driedger, and Meinard Müller. Memory-restricted multiscale dynamic time warping. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 569–573, Shanghai, China, 2016.

[23] Christopher Raphael. Music plus one and machine learning. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 21–28, 2010.

[24] Christopher Raphael and Yupeng Gu. Orchestral accompaniment for a reproducing piano. In *Proc. of the International Computer Music Conference (ICMC)*, 2009.

[25] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[26] Stan Salvador and Philip 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.

[27] Siying Wang, Sebastian Ewert, and Simon Dixon. Robust and efficient joint alignment of multiple musical performances. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2132–2145, 2016.

Oral Session 3

Cultural

QUANTIFYING MUSIC TRENDS AND FACTS USING EDITORIAL METADATA FROM THE DISCOGS DATABASE

Dmitry Bogdanov, Xavier Serra
Music Technology Group, Universitat Pompeu Fabra
dmitry.bogdanov@upf.edu, xavier.serra@upf.edu

ABSTRACT

While a vast amount of editorial metadata is being actively gathered and used by music collectors and enthusiasts, it is often neglected by music information retrieval and musicology researchers. In this paper we propose to explore Discogs, one of the largest databases of such data available in the public domain. Our main goal is to show how large-scale analysis of its editorial metadata can raise questions and serve as a tool for musicological research on a number of example studies. The metadata that we use describes music releases, such as albums or EPs. It includes information about artists, tracks and their durations, genre and style, format (such as vinyl, CD, or digital files), year and country of each release. Using this data we study correlations between different genre and style labels, assess their specificity and analyze typical track durations. We estimate trends in prevalence of different genres, styles, and formats across different time periods. In our analysis of styles we use electronic music as an example. Our contribution also includes the tools we developed for our analysis and the generated datasets that can be re-used by MIR researchers and musicologists.

1. INTRODUCTION

In this paper we propose to explore the editorial metadata available in the Discogs [1] database and show how its analysis can be used as a potential tool to support musicological studies and research in music information retrieval (MIR). Discogs is one of the largest online databases of editorial metadata [2] used by music collectors and enthusiasts. It hosts all metadata under Public Domain license and provides complete monthly data dumps [3] which makes it very easy to access and re-use the data.

[1] https://discogs.com
[2] Discogs mission statement is "to build the biggest and most comprehensive music database and marketplace".
[3] https://data.discogs.com/

Discogs metadata contains information about music releases (such as albums or EPs) including artists name, track list including track durations, genre and style, format (e.g., vinyl or CD), year and country of release. It also contains information about artist roles and relations as well as recording companies and labels. The quality of the data in Discogs is considered to be high among music collectors because of its strict guidelines, moderation system and a large community of involved enthusiasts. The database contains contributions by more than 347,000 people. It contains 8.4 million releases by 5 million artists covering a wide range of genres and styles (although the database was initially focused on Electronic music).

Remarkably, it is the largest open database containing explicit crowd-sourced genre annotations. Discogs implements a two-level genre hierarchy including top-level broad *genres* and more specific sub-genres called *styles*. [4] This taxonomy is determined by the community moderators and there are guidelines for annotation. Genre/style labels are non-exclusive, meaning that a release can be annotated by multiple genres and styles. Releases almost always contain both genre- and style-level annotations as both are required by the submission system of the database.

To our surprise, little is known about Discogs metadata among the MIR community and there is a lack of musicological studies using this data. We identified two MIR studies, using this data in the past. One study used it for music recommendation by building and comparing artist tag-cloud profiles including genres, styles, record labels, years, and countries associated with each artist [3]. Another study analyzes graphs of artist collaborations in order to identify music clusters that can then be associated with genres [5].

There has been done some MIR research in the context of mining genre annotations using other music databases, including AllMusic [14], Wikipedia [16] and Last.fm [15], however, all these studies are limited in their scope and their use-cases of the data. Furthermore, the employed data sources are either ill-structured (Wikipedia, Last.fm), miss explicit genre information (Last.fm), or contain proprietary data (AllMusic) which cannot be reused on the large scale. In order to address these issues we are currently creating a new genre metadata corpus for MIR tasks, including Discogs metadata [10].

[4] https://reference.discogslabs.com/wiki/Style-Guide

In this paper we propose new ways of how editorial metadata can be used in MIR, specifically in the context of studies directed to assist musicological research. To this end, we present a number of example studies using the data available in Discogs in which we identify and quantify some music trends and facts, some of which were previously documented by musicologists and music journalists. In these examples we consider overall trends across broad genres together with a more specific analysis for the case of electronic music which commonly lacks attention of the research community.[5] Our analysis is exploratory and is not aimed at concrete musicological conclusions. Instead we present how data can be used to identify interesting facts and raise questions for further research. Importantly, our study is focused exclusively on music recorded and published on physical or digital media in form of collectable artifacts. We share the tools we developed for the analysis, preprocessed datasets and the complete results for future use by the MIR community and musicologists.[5]

2. DISCOGS DATASET

We downloaded the data dump of music releases dated April 2017. Each release contains the information about a specific album, single, EP, or compilation.[6] For our exploratory study we are interested in artist name, tracklist including track durations, genre and style, format, year and country of each release.

As we wanted to perform per-year analysis of the data, we decided to discard all releases from the ongoing year 2017 to avoid any bias due to incompleteness of data.[7] We extracted the desired data from the dump. The resulting dataset includes 7,954,870 releases by 1,290,943 artists with the total of 67,895,500 released tracks. All releases are annotated by 11 genres and 442 styles.[8] Approximately half of the releases (52%) are annotated with track durations, the 11% of releases are annotated as compilations, and around 1% are marked as mixed.

We estimated the overall genre coverage in terms of percentage of releases, and tracks and artists associated with those releases, out of their total number in the dataset (Figure 1). The rationale behind counting track numbers is that many releases in the databases are vinyl EPs and singles with a smaller number of tracks than albums or compilations. The number of artists provides an alternative useful estimate, as artists may vary in their "release productivity". Our inspection revealed the predominance of the Rock, Electronic, and Pop genre categories in the database,

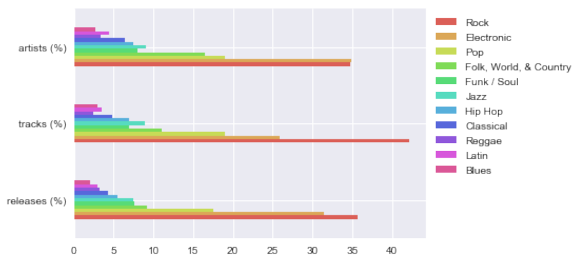


Figure 1: Genre coverage (%).

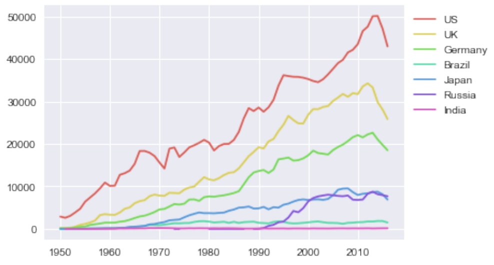


Figure 2: Number of releases per country by year.

with the largest styles being Pop Rock and House. Still, even the least represented genres (such as Blues) have almost 200,000 releases. Around 90% and 74% of styles have more than 1000 and 10,000 releases, respectively.

Inspecting country distribution for releases, we can see an overall predominance of music from western countries in the dataset. Top countries included US, UK, Germany, France, Italy, Japan and Netherlands. Figure 2 presents an example of total number of releases published in various countries by year. Our observations lead us to an open question of whether the disbalance in the distribution represents the actual evolution of the recording industry in each country, or that the Discogs database has insufficient coverage of music from some countries/cultures. While both reasons are plausible, we suppose that our data is still valid for research focused on western countries and music.

3. EXAMPLE STUDIES

3.1 Average Track Duration

In our first example study we analyzed the distribution of the duration of tracks[9] across different genres and styles and its evolution in time. To this end, we discarded all mixed compilations that contain only fragments of tracks instead of full recordings. As there are music releases annotated by multiple genres, we computed tracks duration statistics twice: first, including all releases annotated by a particular genre, and second, excluding releases that were also annotated by other genres.

Figure 3 presents box plots with the obtained results. In both cases, Classical, Electronic, and Jazz music has the largest median durations and the largest variability, ac-

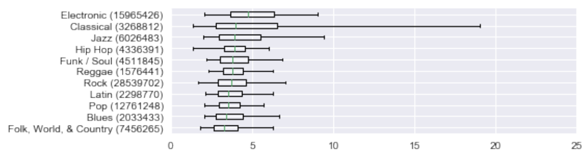
[5] We refer the reader to the additional materials online including code and the detailed analysis results: `https://github.com/dbogdanov/ismir2017-discogs`

[6] See an example of a release page: `https://www.discogs.com/LFO-Frequencies/release/3649`

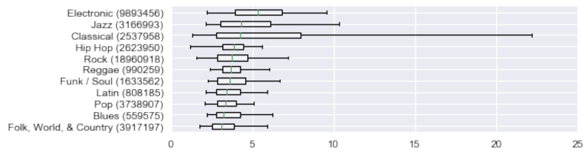
[7] We can also suspect that the database is still missing some releases for 2016 as there was possibly not enough time to gather contributions from the Discogs community. Still, we decided to keep those releases.

[8] For simplicity, we ignored *Brass & Military*, *Children's*, *Non-Music*, and *Stage & Screen* genre categories present in the Discogs taxonomy due to being less represented and/or not being strictly related to music. See the complete genre taxonomy at `https://github.com/dbogdanov/ismir2017-discogs/tree/master/taxonomy`.

[9] Note that we are considering duration of tracks on a recording medium, not of the original music pieces. The former can be seen as a proxy for the latter, at least for some of the music styles.

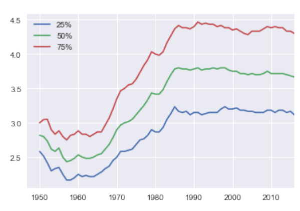


(a) all releases

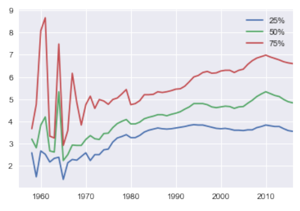


(b) releases annotated by a single genre

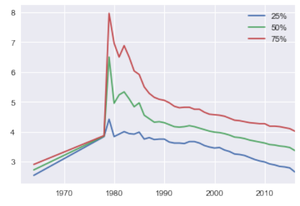
Figure 3: Boxplot of track durations (mins.) for each genre. Number of computed tracks is given in brackets. Whiskers are set at 5% and 95%.



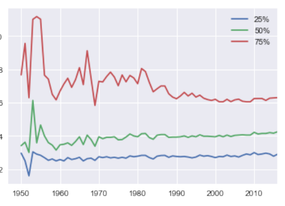
(a) Pop



(b) Electronic



(c) Hip Hop



(d) Classical

Figure 4: Evolution of duration of Rock, Electronic, Hip Hop, and Classical tracks (mins.) by year.

cording to the interquartile range. We observed similar results when including and excluding releases with multiple genre annotations. Still there are some variations in median values, quartiles and whiskers positions, most notably for Classical, Electronic and Jazz. This suggests that music annotated by these genres on the crossover with other genres tends to be shorter. The duration of some music tracks by these genres reached over 8 minutes in contrast to other genres that were far shorter. [10]

Similarly, we analyzed all styles present in the Discogs taxonomy. [11] Overall comparison suggests that the median duration for the majority of styles is below 10 minutes, the shortest starting at 2 to 2.5 minutes (e.g., Grindcore, Crust, Surf, Doo Wop, Beat and Rockabilly). We can observe that some styles can be associated with higher variability in the duration of tracks, while others have durations condensed around a common value. For example, in the case of Electronic music, we can identify a cluster of styles with a large variability (including Harsh Noise Wall, Drone, Noise, Musique-Concrete, Berlin-School, and Dark Ambient) with the duration of tracks surpassing 15 minutes. All these styles share a unconventional, or experimental, approach to sound and music composition. In contrast, the Electronic styles with the lowest variability (such as Eurodance, Jumpstyle, or Grime, among many others) are commonly related to dance/club music.

As a next step, we checked whether there is a global trend in change of average track duration across years. We computed per-genre distributions of track durations across years, some of them shown in Figure 4. [12] We observe existence of a time period with a clear tendency of increase in almost all genre categories: Blues (early 60s to early 70s), Folk, World & Country (mid-60s to 00s), Funk/Soul (early 60s to late 70s), Jazz (late 60s to late 70s), Latin (mid-60s to mid-00s), Pop (mid-60s to mid-80s), Rock (dramatic increase in mid-60s to early 70s) and Electronic (early 70s to 2010 with a consequent decrease). It appears that all these genres have reached a plateau in median track duration after a period of stable increase. We can also see an increment in variability of durations with time.

The increase in duration may be associated with the change of record formats, which we propose to assess in future studies. Interestingly no such tendency was found for Classical music. In contrast, while median duration remains constant, we can observe a decrease in variation with the longer half of the tracks getting shorter. Furthermore, Hip Hop tracks are steadily decreasing in duration since the origin of the genre in the late 70s. All these findings can raise questions for further musicological research.

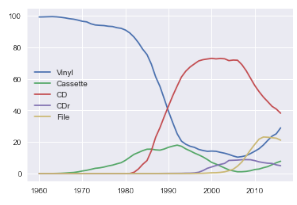
3.2 Release Formats

There is a large number of release formats registered in the Discogs database. [13] In our next example study we quantify the evolution of the most common formats of the past half-century: vinyl, cassettes, CD/CDr, and files. [14] We compare the amount of music released on each of these formats from the 1960s to nowadays. To this end, we count the overall number of releases and tracks recorded on a particular medium. We can then compute the percentage of music released on each format each year. Track and release percentages can vary significantly due to the capacity of each medium: typically CD releases contain a larger number of tracks then vinyl. Figures 5a, 5b and 5c present track statistics for all music in the dataset, and for Blues and Electronic genres in particular. [15]

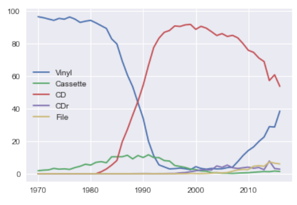
Overall, our analysis corroborates existing RIAA sales reports [12]. We can evidence the commonly known rapid growth of CD format from the mid-80s to the late 90s, followed by a plateau period till the mid-00s. The following decline of CD can be clearly associated with the growth of digital file formats. Remarkably, vinyl, following its de-

[10] Of course, this only suggests a general trend excluding outliers.
[11] See results for all styles grouped per genre in additional materials.
[12] In the case of Electronic and Classical, large value jumps in early years may be associated to small amount of tracks annotated by duration.

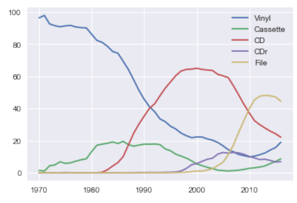
[13] `https://www.discogs.com/help/formatslist`
[14] Note that we cannot be confident in estimations for the file format as digital releases may be under-represented in the Discogs database.
[15] See additional materials for complete results.



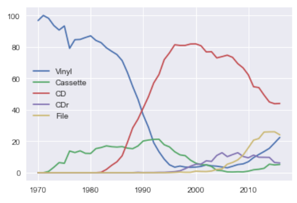
(a) All music



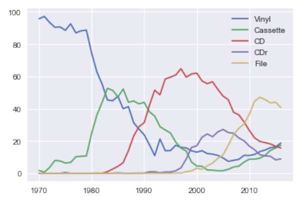
(b) Blues



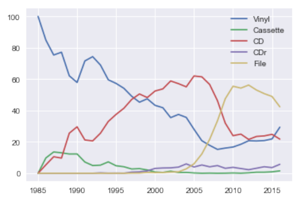
(c) Electronic



(d) Synth-pop (Electronic)



(e) Experimental (Electronic)



(f) Deep House (Electronic)

Figure 5: Percentage of released tracks per format by year (%).

cline since the mid-80s, is now growing since the early 10s, which corroborates recent observations of the new "vinyl boom" [2].

Cassette releases appear to be always below vinyl or CD releases through the history of the format, descending to its supposed death in the mid-00s. However, since then, we can observe the second growth of cassettes, which confirms observations of the growing "cassette culture" by some music journalists and musicologists [8,7,6]. Interestingly, there is a considerable amount of music released on a CDr format, which appeared in the late 90s and achieved its maximum at the time of the death of cassettes. We can suppose that the observed CDr and cassette trends are linked to the DIY culture of independent music distribution [18].

Analyzing particular genres, we identified Blues, Rock, Reggae, and Funk/Soul to have the highest percentage of tracks released on vinyl in the recent years, surpassing 30%. Remarkably, these genres can be nowadays considered somewhat "old-school", and therefore of a potentially higher interest among vinyl music collectors. We hypothesize that many of new vinyl releases are reissues.

Finally, we ran per-style analysis of data, on the example of Electronic music, in order to identify peculiarities of music distribution within certain styles. Figures 5d, 5e and 5f demonstrate differences in formats on the example of Synth-pop, Ambient and Deep House. From our results we can evidence a transition to the predominance of CDs for all considered styles with the turning points [16] starting since 1988. The styles that moved to CD first were Ambient, Synth-pop and Experimental, with their turning points in transition between 1988 and 1991. We speculate that early transition to CD was at least partially motivated by the demand of home consumers, meanwhile other styles, supported by DJs, had a technical demand for vinyl. Such styles had their transition point to CD later between 1993 (e.g. House, Techno) to 1996-97 (Drum n Bass, Deep House). Remarkably, we identified the existence of styles with the absolute predominance of digital formats (CD, CDr and file). In the case of Glitch, this fact may be linked to musical characteristics of the style, which make releases on vinyl/cassettes aesthetically or technically unfeasible.

In 2016 the digital file format is leading in almost all styles. The turning point towards its predominance appeared between 2007 and 2011. Still, we can observe a trend in growth of vinyl in the recent 6 years. Moreover we can also see the growth of cassettes since 2005, with the most significant example being Experimental electronic music (reaching almost 20% of tracks being released on cassettes in 2016). Interestingly, almost 30% of released Ambient tracks in the early-to-late 80s, and similarly over 40% of Experimental from the early 80s to the early 90s, were released on cassettes (which again supports the existence of the DIY cassette scene in experimental music of the 80s) [17].

3.3 Genre and Style Trends

In this section we present another use-case example: an analysis of genre and style trends across time periods. We consider overall trends in genres and exemplify style analysis on Electronic music. Again, we quantified the amount of music in terms of number of releases, tracks and artists associated with those releases. Their absolute values (the amount of music in year *N* by genre *G*) and proportions (the percentage of music from year *N* by genre *G*) allow us to suggest possible trends. Figure 6 presents results of the analysis. [17] Below we summarize our observations on the tracks level.

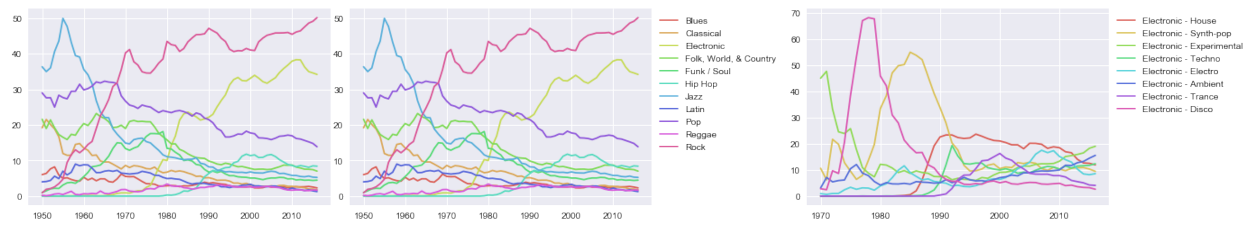
Rock appears to be the major genre since the late 60s covering more then 40% of released tracks since the late 70s and reaching 50% nowadays. It is currently followed by Electronic music that reached its peak at 38% in 2011. Pop music is in a steady decline since its short dominance in the mid-60s, falling below Electronic since the early 90s. We can also see the decline of Jazz after its huge 50% peak in the mid-50s, being a predominant genre at that time, the rise and fall of Funk/Soul, and the growth of Hip Hop being the 4th leading genre in the 2000s. A valley in the Rock plot in the mid-70s corresponds to the peak of Funk/Soul (including "the disco boom" [9]). [18]

We then repeated the analysis for styles of Electronic music. Again, we used absolute and relative release/track/artist counts. The relative values represent the

[16] The year of an equal number of tracks released on both formats.

[17] Note that the percentage values do not sum up to 100 because releases can be annotated by multiple genres.

[18] This is especially well seen on the release and artist-level plots, with the number of Funk/Soul artists being temporarily higher than for Rock.



(a) Tracks (all genres) (b) Releases (all genres) (c) Tracks (Electronic)

Figure 6: Percentage of released music per genre and style by year (%).

percentage of music within a style out of the total amount of Electronic music released each year. Out of 110 Electronic styles, we identified the most important ones in terms of their overall presence: we computed their relative share in all Electronic music across years from 1970 to 2016. We then summed these values for each style and identified the styles with the highest values. Figure 6c presents track-level results for those styles. [19]

According to our data, we can see how electronic Disco peaked in the late 70s followed by the Synth-pop peak in the mid-80s, both being the predominant styles of their time. The decline of Synth-pop in the late 80s/early 90s met the peak of House (1990) and the first peak of Techno (1992). Later, growing styles included Trance (peaked in 2000), and Electro (having its second peak in the late 00s, the first one in 1984). We can also guess the period of birth of each genre using our data (e.g., house in the mid-80s, techno in the late 80s, and trance in the early 90s). All of these observations seem to be well-aligned with the existing literature on the history of electronic music [11].

Interestingly, after the year 2010 we observe lower percentage values for all styles. This suggests the diversification of electronic music: more styles are taking a share in the amount of music released each year. Finally, it is worth noting that similar analyses can be run on per-country basis. This can be useful for identifying potential regional trends or analyzing the following of a particular style in various countries. [20]

3.4 Genre and Style Co-occurrences

In this section, we consider another use-case for editorial metadata with genre/style annotations and study co-occurrences between different genre and style labels. We also attempt to assess the specificity of labels: while top-level genre categories are very broad, styles may vary a lot in their specificity and coverage.

Given that releases can be annotated by multiple genre/style labels within the Discogs taxonomy, we computed a genre (and style) co-occurrences matrix in order to identify possible relations. For each pair of genres (or styles) X and Y we counted the number times both appear on the same release across all releases in the dataset. The

[19] See additional materials for full results.

[20] For example, we could observe that Hardcore, Breakbeat and Drum n Bass styles, well-represented in UK in the early-to-mid 90s, were never prevailing in Germany.

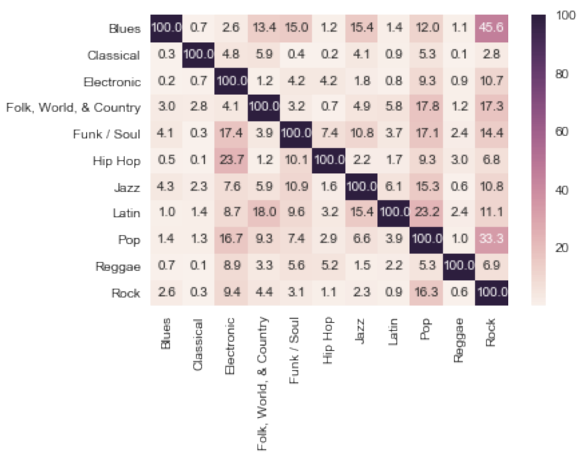


Figure 7: Genre co-occurrences (%).

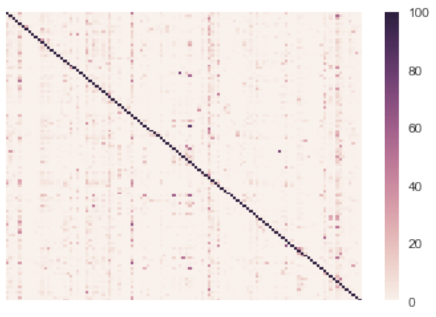


Figure 8: Electronic styles co-occurrences (%).

resulting matrix is asymmetric and its values represent the percentage of music by genre (style) X (on the x-axis) also being annotated by genre (style) Y (on the y-axis).

Figure 7 presents the resulting genre co-occurrences matrix computed using all our data. We can conclude from it that Classical, Electronic, and Reggae seem to be the genres that are well isolated from others, that is, the music under these genre labels is self-contained and all co-occurrences with other genres are relatively small (below 11%). On the contrary, the most interconnected genres are Blues (46% of it is also Rock), Pop (33% is Rock) and Hip-Hop (24% is Electronic). We can also observe how Pop and Rock, and Electronic appear to be commonly co-occurring genres for many genres, probably, due to being the most popular ones.

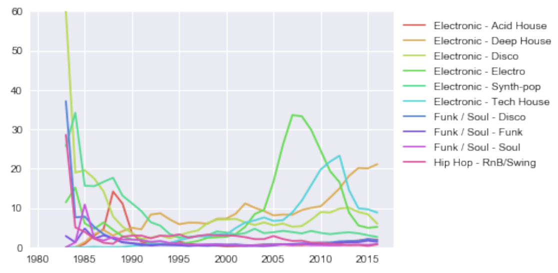


Figure 9: Percentage of House releases also annotated by other styles by year (%).

We proceeded with style co-occurrences in a similar manner. The resulting matrix is huge and here we analyze a portion, again, for the case of Electronic music. [21] To give an idea, Figure 8 provides an overview of this matrix containing 110 Electronic styles. As we can see, there are a number of traceable vertical lines corresponding to particular styles that often co-occur with many other styles. Those include Techno, House, Experimental, Synth-pop and Ambient — all being among the most frequent styles within Electronic music in our dataset. Less predominant, but still traceable vertical lines include Electro and Downtempo. We can suppose that these style labels are wide in coverage or generic enough to embrace some other styles. In contrast, we can also see styles that typically do not co-occur with others (e.g., Speed Garage or New Beat), which might indicate their high specificity or "nicheness".

Summing all row values for a column we can get a "genericness-vs-nicheness" score. According to it, some examples of niche styles are Beatdown, Neo Trance, Skweee, New Disco, and Italo House, while the most generic styles include Downtempo, Synth-pop, Electro, Ambient, Techno, Experimental, and House. On the other side, co-occurrence values above 50% might indicate a subgenre-to-genre relation and give us a degree of potential "sub-styleness", or "derivativeness". For example, 86% of Gabber is also annotated as Hardcore, 75% of Hardbeat as New Beat, and 68% of Power Electronics as Noise. After identifying all such examples in the matrix, we were able to corroborate this hypothesis.

Interestingly, it is also possible to compute co-occurrences for particular time epochs. Figure 9 illustrates the idea of evolution of style co-occurrences in time on the example of House. We can see how Disco, Synth-pop, Funk and Soul potentially had an influence on the style at the time of its origin (indeed these styles are often cited as such [11]), followed by a peak of Acid-House and then, later, Electro and Tech-House.

The co-occurrence matrices demonstrate the intrinsic variability in genre annotations and we believe that such data can be very useful for the MIR community in the context of evaluating music genre classifiers and for other tasks. Indeed, some studies on audio-based genre classification (such as [13, 1, 4]) reveal similar patterns in misclassifications, and they can be supported by our data.

[21] See additional materials for full results.

4. DISCUSSION

Naturally, the presented data analysis is limited by the coverage of the Discogs database, with a possible bias towards Western music and collectable music items, and other sociocultural factors. Digital releases are possibly underrepresented since the new online distribution models allow artists to instantly share their work and the concept of "release" might be changing. We are far more confident in the data for the former time period of predominance of physical releases. Assessment of coverage of editorial music databases is an open question for future research.

Our analysis is essentially grounded on the statistics of music production, not consumption. No analysis of music trends in terms of popularity among listeners is addressed. Instead, we deliberately focused on another aspect: what music artists tend to produce, including in the long tail. A release of 100 copies is treated equally to a release of thousands in our analysis.

We are also aware of the problem of release-level genre/style annotations: labels do not necessarily apply to all tracks. Still, we suppose a certain congruency between tracks on a release. Interestingly, our data reveals that even releases with a single track have multiple labels (1.2 genres and styles on average). This suggests that a genre annotation problem is inherently multi-label. Finally, in our analysis we are limited to the Discogs' genre taxonomy. Their broad genre categories might not be appropriate for some research tasks, but we can be much more confident in style annotations, at least for some genres.

Many music releases actually correspond to the same conceptual items (e.g., album CD version, CD version in another country, vinyl version, reissue). Discogs provides information about their groupings, and it should be considered depending on a task at hand. For simplicity, in our example studies we treat such releases as if they were independent. Such releases are often released in various formats, countries, years, and can have different track lengths, bringing additional information to our analysis.

5. CONCLUSIONS

In this paper we propose to take a closer look on the editorial metadata in the Discogs database. We believe that analysis of this data can be a valuable tool for researchers. It can help to identify and analyze various musical phenomena and raise different musicological questions. Importantly, Discogs is one of the largest sources for such data in the public domain which allows to address potential research questions on a very large scale. We demonstrated the use of this data in a number of example studies in which we attempted to quantify a number of music trends and facts, some previously documented by musicologists and music journalists. Our examples are far from being complete and of course there are more potential questions to be raised and addressed using this data. We share the analysis tools we developed, our preprocessed datasets and the complete results for our example studies for further re-use by MIR and musicology researchers.

6. ACKNOWLEDGEMENTS

This research has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 688382 (AudioCommons). We also thank Alastair Porter for proofreading.

7. REFERENCES

[1] B. K. Baniya and J. Lee. Importance of audio feature reduction in automatic music genre classification. *Multimedia Tools and Applications*, 75(6):3013–3026, 2016.

[2] D. Bartmanski and I. Woodward. The vinyl: The analogue medium in the age of digital reproduction. *Journal of Consumer Culture*, 15(1):3–27, 2015.

[3] D. Bogdanov and P. Herrera. Taking Advantage of Editorial Metadata to Recommend Music. In *International Symposium on Computer Music Modeling and Retrieval (CMMR'12)*, 2012.

[4] D. Bogdanov, A. Porter, P. Herrera, and X. Serra. Cross-collection evaluation for music classification tasks. In *International Society for Music Information Retrieval Conference (ISMIR'16)*, 2016.

[5] J. Burke, R. Rygaard, and Z. Yellin-Flaherty. Clam!: Inferring genres in the Discogs collaboration network. 2014.

[6] J. Demers. Cassette tape revival as creative anachronism. *Twentieth-Century Music*, 14(1):109–117, 2017.

[7] C. Eley. Technostalgia and the resurgence of cassette culture. *The Politics of Post-9/11 Music: Sound, Trauma, and the Music Industry in the Time of Terror*, pages 43–55, 2011.

[8] M. Hogan. This is not a mixtape. `http://pitchfork.com/features/article/7764-this-is-not-a-mixtape/`, 2010. Accessed on 13.07.2017.

[9] T. Lawrence. *Love saves the day: A history of American dance music culture, 1970–1979*. Duke University Press, 2004.

[10] A. Porter, D. Bogdanov, and X. Serra. Mining metadata from the web for AcousticBrainz. In *International workshop on Digital Libraries for Musicology (DLfM'16)*, pages 53–56. ACM, 2016.

[11] S. Reynolds. *Energy flash: A journey through rave music and dance culture*. Soft Skull Press, 2012.

[12] RIAA. U.S. sales database. `https://www.riaa.com/u-s-sales-database`, 2017. Accessed on 13.07.2017.

[13] N. Scaringella, G. Zoia, and D. Mlynek. Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine*, 23(2):133–141, 2006.

[14] A. Schindler, R. Mayer, and A. Rauber. Facilitating comprehensive benchmarking experiments on the million song dataset. In *International Society for Music Information Retrieval Conference (ISMIR'12)*, 2012.

[15] H. Schreiber. Improving genre annotations for the Million Song Dataset. In *International Society for Music Information Retrieval Conference (ISMIR'15)*, 2015.

[16] H. Schreiber. Genre ontology learning: Comparing curated with crowd-sourced ontologies. In *International Society for Music Information Retrieval Conference (ISMIR'16)*, 2016.

[17] J. Scott. The Noise-Arch archive. `https://archive.org/details/noise-arch`, 2015. Accessed on 13.07.2017.

[18] R. Strachan. Do-It-Yourself: Industry, ideology, aesthetics and micro independent record labels in the UK. *Unpublished masters thesis, University of Liverpool, Liverpool, England*, 2003.

THE MUSIC LISTENING HISTORIES DATASET

Gabriel Vigliensoni and Ichiro Fujinaga
Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
McGill University, Montréal, Québec, Canada
`[gabriel.vigliensonimartin,ichiro.fujinaga]@mcgill.ca`

ABSTRACT

We introduce the Music Listening Histories Dataset (MLHD), a large-scale collection of music listening events assembled from more than 27 billion time-stamped logs extracted from Last.fm. The logs are organized in the form of listening histories per user, and have been conveniently preprocessed and cleaned. Attractive features of the MLHD are the self-declared metadata provided by users at the moment of registration whose identities have been anonymized, MusicBrainz identifiers for the music entities in each of the logs that allows for an easy linkage to other existing resources, and a set of user profiling features designed to describe aspects of their music listening behavior and activity. We describe the process of assembling the dataset, its content, its demographic characteristics, and discuss about the possible uses of this collection, which, currently, is the largest research dataset of this kind in the field.

1. INTRODUCTION

The modeling of users for multimedia information retrieval systems has been a research topic since the first International Symposium on Music Information Retrieval (ISMIR) in 2000. In that meeting, it was observed that to create modern, more efficient, and personalized music information retrieval systems, the modeling of users would be necessary because many features of multimedia content delivery are perceptual and user-dependent [6].

Sixteen years after the first ISMIR meeting, the landscape of music consumption has changed enormously. The rise and fall of peer-to-peer networking led to the reinvention of the music industry: the paradigmatic music product was no longer a full album in a physical format, but individual music files available in online digital music stores. Thanks to the miniaturization of portable media players and also to almost ubiquitous Internet access, a change of paradigm in music consumption has happened again, and people seem to not want to pay for individual tracks. Instead, they are willing to pay for services that allow them to access, search, and discover music items—artists, albums, or tracks—within large repositories [23].

On-demand digital music streaming services are currently the fastest growing sector of the global music industry [11]. In fact, in 2015 the digital revenues that these systems generated overtook the income from physical music goods for the first time in music industry history [12]. As a result, the on-demand music streaming landscape these days seems to be a lucrative battlefield, and one on which many companies want to compete. However, since the majority of the listeners' accounts in music streaming services use the "free" or "freemium" business model—advertisement-supported basic streaming services—a large share of the income of music and media streaming companies comes from targeting ads more precisely at listeners [18]. It seems that the streaming model is like modern advertising.

In this model, people are no longer passive observers but direct participants in the battlefield that is the digital media and music streaming landscape. In fact, the traded goods in this business are individual profiles and psycographic traits (e.g., interests, lifestyle, personality, values) which are extracted from correlating people's listening habits with their sociographic characteristics [17]. As a result, listeners are the source of information, but they also are the final target for all the advertising these companies are making money from.

As music information researchers, our community has to be able to observe, investigate, and to gather insights from the listening behavior of people in order to develop better, personalized music retrieval systems. Yet, since most media streaming companies know that the data they collect from their customers is very valuable, they usually do not share their datasets. A honorable mention goes to Netflix, company that challenged the recommendation research community in 2006 with a large dataset of ratings of users on movies. Insights and techniques developed for that competition are still being used widely today.

2. PREVIOUSLY AVAILABLE DATASETS

A number of datasets for music listening research have been collected and released by research groups. These datasets provide information relating the interaction of a large number of listeners and music items.

Celma assembled the Last.fm Dataset-360K, a dataset of playcounts with listeners' demographic data for 360K

Dataset name	Type	Source	Size	Demographics	Linkage	Other
Last.fm Dataset 360-K [5]	Playcounts	Last.fm	18M logs, 360K users	Yes	Yes	Only includes most frequently listened artists
Last.fm Dataset 1-K [5]	Listening histories	Last.fm	19M logs, 1K users	Yes	Yes	Full music listening histories
Yahoo! Music Dataset [7]	Ratings	Yahoo! Music Radio	262M logs, 1M users	No	No	Hierarchical structure of music items
HetRec2011-last.fm.2k [4]	Playcount	Last.fm	2K users	No	No	Bidirectional users' relations and artist tags
Echo Nest Taste Profile subset [13]	Playcounts	Undisclosed	48M logs, 1.2M users	No	Yes	Linked to Million Song Dataset
EMI Million Interview Dataset [8]	Interviews	Individual interviews	1M users	Yes	Unknown	Partial information available
MusicMicro 11.11-09.12 [19]	Listening histories	Twitter	600K logs, 137K users	Geolocalized logs	No	Precise geolocation data of each log
Million Musical Tweets Dataset [9]	Listening histories	Twitter	1M logs, 215K users	Geolocalized logs	Yes	Many users have only a few listening events
#nowplaying Music Dataset [24]	Listening histories	Twitter	50M logs, 4.2M users	No	Yes	Many users have only a few listening events
LFM-1B [20]	Listening histories	Last.fm	1B logs, 120K users	Yes	No	Comes with a set of features describing music consumption behavior. The music listening histories are shifted according to the time zone of listeners, and so they are not directly comparable.
MLHD	Listening histories	Last.fm	27B logs, 583K users	Yes	Yes	Comes with MBIDs, estimation of listeners' time zone, and users' activity features.

Table 1. Comparison of freely available datasets of music listening events.

listeners, and the Last.fm Dataset-1K, a set of full listening histories with time-stamped logs [5]. Though richer, the latter dataset included logs for only 1K listeners. Following the Netflix prize, Dror, Koenigstein, and Koren released the Yahoo! Music Dataset, a collection of 1M people's aggregated ratings on music items [7]. Later on, Cantador, Brusilovsky, and Kuflik presented the HetRec2011-Last.fm-2K, another dataset with song playcounts for the 50 most listened artists of 2K listeners [4]. McFee et al. introduced The Echo Nest Taste Profile subset, a dataset of song playcounts of 1M listeners collected from undisclosed services [13]. Neither of these two datasets, however, provided timestamps of the music logs or demographic information about the listeners. The EMI Group Limited promised a dataset of 1M interviews about people's music appreciation, behavior, and attitudes [8], but only partial information was made available. None of the aforementioned datasets simultaneously provided individual music listening logs as well as demographic data for a large amount of listeners.

More recently, music listening logs have been collected from the social networking service Twitter. Schedl released MusicMicro 11.11-09.12, a dataset of about 600K music-related *tweets* with temporal and spatial data [19]. Hauger et al. released the Million Musical Tweets Dataset [9], a collection of 1M music-related geolocalized microblog posts with partial linkages to other services. Zangerle et al. introduced the #nowplaying Music Dataset, a collection of 50M music-related posts linked to MusicBrainz [24]. In these collections, however, there were a large number of listeners with only one or two logs, and so, in many cases, the datasets provided a few listening events for many users instead of listening histories.

Finally, Schedl introduced LFM-1B, a very large dataset of more than 1B logs collected from Last.fm user interactions [20]. Each log includes artist, album, and track names, the timestamp of the log, as well as each user's Last.fm identifier. The dataset also comes with users' demographic information as well as a set of features that describe music consumption behavior per user. However, the dataset does not provide common identifiers with other music databases, and so the only way to link the music items is by string matching.

In Table 1 we provide a summary of available databases of music listening logs. We can see that among all the datasets reviewed, the only one that provides full music listening histories, listeners' self-declared demographic data, as well as identifiers easily linkable to other databases of music information is the Last.fm Dataset-1K. However, the size of the dataset is very small to perform a large-scale analysis with global reach. In order to ameliorate this situation, we decided to collect our own dataset considering all the aforementioned characteristics.

3. THE MUSIC LISTENING HISTORIES DATASET

In this section we will describe the creation of the Music Listening Histories Dataset (MLHD), a large dataset of full music listening histories. We will review the concept of music listening history and will present the criteria for the data collection and cleaning of the data. We will also provide insights about the demographic characteristics of users in the dataset and will explain the need of providing a value for normalizing the time zone of the logs.

3.1 Music Listening Histories and Last.fm

Listening histories are a timeline of listening events. Analyzing them in a linear fashion is interesting because we can observe when people consume music, and what music they enjoy or do not enjoy over time. However, since people seem to follow periodic listening cycles [10], the aggregation of these listening histories by collapsing them into different periods of time can provide extra layers of information that can be used to infer people's listening patterns and preferences.

Last.fm is an online digital music service available since 2002. It was originally conceived as a web-based radio station. Immediately after its launch, the company incorporated the tracking of music listening logs as a core part of its service. However, Last.fm stands out from most music streaming services that collect user data because it not only gathers listening logs (known as *scrobbles*) from the interaction of its users withing the system's ecosystem, but also from the interaction between users and a wide range of third-party music and media players by means of the *scrobbler* service.

Last.fm offers free access to the listening data they collect from listeners, as well as music metadata, biographies, pictures, charts, tags, ranking data by country, and other information by means of a well-documented API. At the moment of registration, every user must accept the Last.fm Terms of Use and the Last.fm Privacy Policy. [1] These terms establish that their listening habit data will be available to third parties via their API for commercial and/or non-commercial purposes. The users are also asked to provide basic demographic information such as their date of birth, country, and gender.

All aforementioned characteristics, added to the fact that the Last.fm API Terms of Service establish that Last.fm offers a "limited terminable licence to copy and use the Last.fm Data" that is free of charge "for non-commercial purposes" [2] persuaded us to choose Last.fm as the data source to assemble the MLHD.

3.2 Data Collection

In order to retrieve full music listening histories and to obtain even data across aggregated periods of time, we followed previous research [1] and searched only for listeners with a minimum of two years of activity since they started to submit music logs to Last.fm. Also, in order to prevent collecting data from casual users that registered for a service, tried it, but never used it again, we collected data only from listeners which had an arbitrary average of, at least, ten scrobbles per day. The two constraints forced all listeners in our dataset to have a minimum of 7,300 (i.e., 365 $\times$ 2 $\times$ 10) music logs submitted to the Last.fm database.

Differently to all other datasets with Last.fm data, we collected listening data by using an undocumented (but deprecated) method that allowed us to not need actual usernames for calling the Last.fm Web services [21]. Instead, we simply passed Last.fm users' internal identifiers as arguments of the API requests. Since these IDs are sequential, this approach permitted us to sample users randomly across the entire database instead of sampling users based on their *friends* or on an artist's *top fans*, which are methods probably more biased. We aimed to collect full listening histories, and so we fetched people's listening logs by using the Last.fm's API method `user.getRecentTracks()`, and paginated iteratively throughout the chosen listeners' full music listening histories.

3.3 Data Cleaning, Sanitization, and Organization

Within each music listening history, we organized each of the logs in quadruples with the form of `<timestamp, artist-MBID, release-MBID, track-MBID>`, where timestamp is a global coordinated universal time (UTC) stamp, and MBID stands for MusicBrainz identifier. MBIDs are 36-character universally unique identifiers (UUID) that are permanently assigned to entities within the MusicBrainz database to ensure a reliable and unambiguous form of identification. Since Last.fm exposes MBIDs as public identifiers of music entities in their database, we collected them directly for each artist, release, and track. These three entities are hereafter denominated "music entities." Finally, all data per user was stored within a single file, with the logs sorted sequentially by their timestamp.

After close inspection of the data, we realized that there were two issues in some of the listening histories: (i) there were duplicated music logs (i.e., same timestamp and MBIDs); and (ii) some logs were too close in time (i.e., less than 30 seconds apart, which is the minimum that Last.fm requires to consider a played track as a valid log). We hypothesized that these issues were artifacts produced by the interaction of the Last.fm servers and some scrobblers. As a result, we decided to perform a cleaning process before storing the data, and so we filtered out all logs with the same MBID and timestamp, and we also filtered out all scrobbles that were less than 30 seconds apart in time. All in all, the average percentage of duplicated logs removed for each user was eight percent, and one percent for those logs that were too close.

It is worth mentioning that sometimes the metadata provided by the scrobbler is not enough to produce a full match for artist, release, or track. In cases like this, the music listening log returned by the Last.fm API will have only partial information. As a result, not all logs in the MLHD have a full set of MBIDs.

In Figure 1 we show the percentage of all combinations of MBIDs across all music logs in the dataset. It can be seen that about 58 percent of all music logs in the MLHD

[1] Privacy Policy available at `http://www.last.fm/legal/privacy`

[2] Terms of Service available at `http://www.last.fm/api/tos`

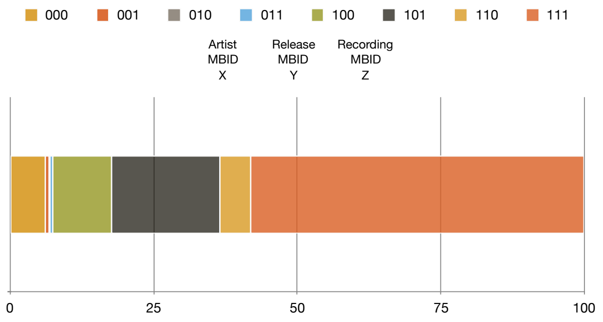


Figure 1. Percentage of music logs with combination of MBIDs. 0 stands for no presence of the corresponding MBID in the scrobble and 1 for its existence.

have full data (i.e., MBIDs for the three music entities), and 93 percent of the logs have, at least, the artist MBID.

3.4 Data Exploration

We computed from the music listening histories' UTC timestamps a series of features that aggregated the number of scrobbles of each listening history into several time spans. These low-dimensional representations of a user activity may facilitate the creation of plots and their visual inspection in order to gain insights or detect anomalies from single listener or groups of them. These per-user features are: *hourly activity*, *hourly activity by week hour*, *weekly activity*, *monthly activity*, *yearly activity*, *weekday activity*, *Saturday activity*, and *Sunday activity*.

3.5 Demographics

The MLHD currently consists of more than 27 billion music logs taken from the listening histories of 583K people that have linked their digital music players to Last.fm. In this massive repository, we counted more than 555K different artists, 900K albums, and seven million tracks. Table 2 summarizes the number of logs, unique listeners, and music entities in the dataset.

Dataset	Logs	Listeners	Artists	Albums	Tracks
	27MM	583K	555K	900K	7M

Table 2. Music listening histories dataset summary.

The distribution of the average number of daily submitted music logs per listener is shown in Figure 2. Axes in the plot are in log scale. The curve exhibits a close to power law characteristic. As expected, due to the constraints we set for collecting listeners' listening histories, the minimum average daily number of music logs per user was ten. Listeners with an average of eleven logs were the largest group, with about 30K listeners. The median number of submitted logs per user was 35K. The median age of the listening histories was 4.5 years.

Now we will describe the nature of the users in the dataset according to their self-declared age, gender, and country. This information is asked to the users at the moment of registration.

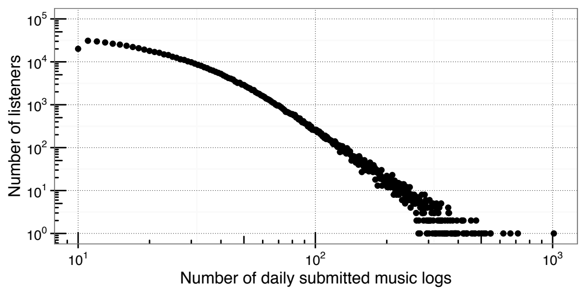


Figure 2. Distribution of the average number of daily scrobbles per listener.

3.5.1 Age

In terms of age, 71 percent of the listeners in the dataset declared their date of birth, which is much higher than similar datasets [5, 20]. Among them, 98 percent of the users had a self-declared age within 15 and 54 years old. In spite of the small magnitude of the probably deceiving information found in the two percent out of this age range, we decided to filter them out from the dataset. The mean age of listeners in the dataset is 25.4 years old, the median is 24, and the mode is 22. Since these are values similar to the ones found in similar datasets, this skew in the distribution indicates a bias in our dataset—and probably in Last.fm users—towards youth and young adults. We show the age distribution of listeners of the MLHD in Figure 3.

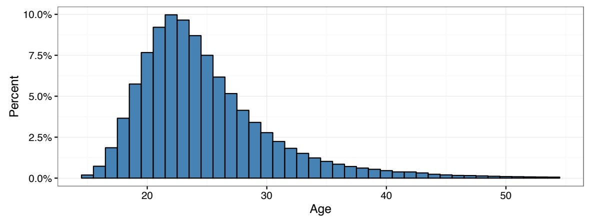


Figure 3. Age distribution of MLHD listeners within the [15, 54] years old range.

3.5.2 Gender

In terms of gender, about 82 percent of the people in the dataset declared a gender at the moment of their registration or afterwards. In Figure 4 we show the self-declared gender distribution among these users.

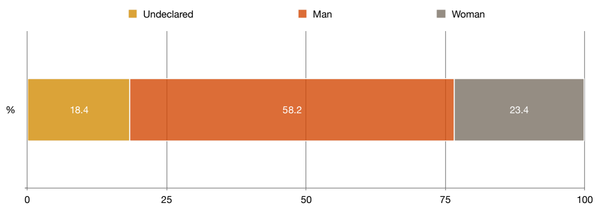


Figure 4. Percentage of listeners' self-declared gender.

It can be seen that there is a bias towards male listeners in the MLHD. Since this bias is also observed in similar dataset, this can be an indication that Last.fm has more male than female users.

We compared the age within each self-declared gender with balanced groups. The total number of listeners without self-declared gender was slightly more than 100K, and so we sampled 100K listeners from each group. The mean of the Not declared ($\mu = 25.67$) and Male ($\mu = 25.60$) groups did not differ greatly ($p = .400$), perhaps indicating that the first group may have a large proportion of male users. On the other hand, users self-declared as Female ($\mu = 22.99$) had a different lower mean age than the Male group ($p < .001$). In other words, users in our dataset self-declared as Female are younger than the ones declared as Male.

3.5.3 Country

In terms of location, 82 percent of users in the MLHD self-reported a country. These users belong to 239 different countries or territories as defined in the ISO 3166-1 International Standard for country codes. Among these territories, 19 countries had at least one percent of the total amount of listeners in the dataset. These "top countries" combined accounted for more than 85 percent of the total number of listeners in the dataset.

In order to determine how countries were relatively represented in the MLHD, we divided the percentage of users per country by the actual country population. [3] This metric gave us a better description about how different countries' populations were represented in our dataset. In Figure 5 we show a map that presents the relative number of listeners per country normalized by the corresponding number of inhabitants in each country.

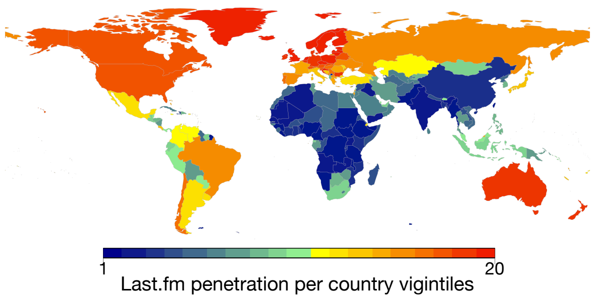


Figure 5. Relative number of listeners per country, normalized by the number of inhabitants in each country.

The color palette of the plot was based on vigintiles (20 quantiles) of the data, with red indicating the highest vigintile, and blue the lowest one. If our dataset has similar distinctive qualities in comparison with the overall Last.fm data, this map can be interpreted as the Last.fm market penetration by country. By looking at the higher vigintiles we can see that listeners from most zones are represented in the MLHD. In particular, Northern European, North American, and Australasian countries have the largest proportion of listeners submitting music logs to Last.fm. Also, some countries in South America show similar penetration levels to some Mediterranean countries in Europe. People from Africa, South Asia, and Far East Asia are not extensively represented in our dataset.

Finally, pair-wise mean age comparison using balanced groups of listeners per country (N= 4.5K) showed significant differences between listeners from some of the top countries. For example, Brazilian listeners are younger on average ($\mu = 22.6$) than all other top countries ($p < .001$), except for Poland, Russia, and Ukraine. On the other hand, Japanese listeners are older on average ($\mu = 29.0$) than users from all the other top countries ($p < .001$), except for Spain and France.

3.6 Time Zone Normalization

Last.fm collects scrobbles using the Unix time stamp format no matter where the logs were generated. Therefore, all music logs within the Last.fm database have the same temporal point of reference. Beyond the timestamp and the MBID for the three music entities, the logs do not store any additional geographical information such as city, country, or the time zone where they were generated.

The lack of information about where the logs were actually generated can be a problem. If the researcher wants to find trends in people's daily, weekly, and monthly music listening behavior, it is necessary to aggregate their music listening histories over time. However, the aggregated listening patterns from people in different time zones is shifted depending on where they are. As a result, it would be misleading to directly compare their patterns. The country information could be used to estimate a listener's time zone, but many countries span their territories over several time zones.

In previous studies with similar data, the researchers have hand-picked listeners within the same time zone [2, 3, 16] or are just compared their daily listening patterns directly [20]. However, a research dataset to perform studies at the global level must provide this information in order to properly compare the music listening histories.

We followed an approach for time zone normalization based on the assumption that people share hours of sleep at night [21], and computed the time shift of the listening histories in the MLHD. In Figure 6 we show the estimated distribution of the time zones of all listeners in the dataset. We can observe a peak in the estimated time zone from where people submitted music logs at time zone GMT +0, with about 17 percent of the dataset users. Additionally, a large proportion of the listeners were estimated to be within time zones corresponding to Western Europe, but also spread out throughout the different time zones in America.

4. CONCLUSION

All in all, the MLHD provides three sources of data for each user: (i) demographic metadata, (ii) sanitized full music listening histories, and (iii) low-dimensional feature vectors describing the full listening histories in terms

[3] Population data for the year 2012 taken from the World Bank Open Data repository, available at `http://data.worldbank.org/indicator/SP.POP.TOTL`

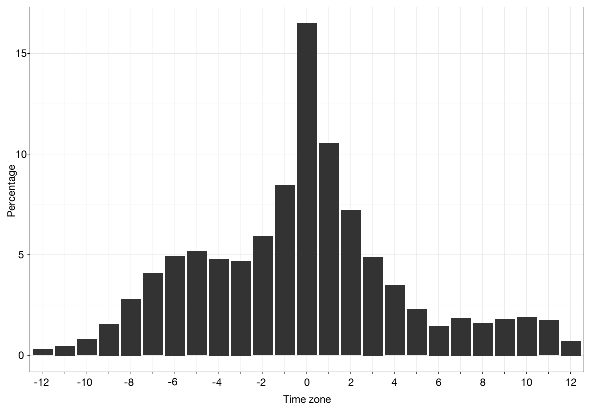


Figure 6. Distribution of the time zones for all users in the dataset (N = 583K).

of user activity. As a result, the full music listening histories compiled in the MLHD dataset offer a large amount of information. On top of having a very fine time granularity—providing second-accurate data about the music item played back in a media player by a specific user—their aggregation into different spans of time may provide clues about the people's listening behavior characteristics and their listening trends over time.

A big advantage of the MLHD dataset over other datasets for listening behavioral research is that it is based on MusicBrainz identifiers (MBID). This feature allows the easy linkage of each log to, for example, all services of the MetaBrainz Foundation ecosystem (i.e., MusicBrainz, AcousticBrainz, ListenBrainz, and CritiqueBrainz) and to other services that provide additional data accessible through these IDs (e.g., Last.fm provides folksonomy tags for artists, albums, and tracks, and DBPedia links Wikipedia open music data to MusicBrainz by means of MBIDs). Therefore, music listening histories can be linked to resources from other repositories, thus enabling the aggregation, linkage, and expansion of the data and the knowledge about people's music listening behavior.

In terms of possible uses of the dataset, data aggregations extracted from the MLHD have already been used in combination with other sources of data. In particular, it has been used as part of the datasets for "Sound and music recommendation with knowledge graphs" [15]. In these datasets, a subset of music listening histories from the MLHD were aggregated into playcounts and used in combination with additional song data collected from Songfacts.com to enable the study of hybrid music recommendation models using additional user-provided factual information describing songs and artists [14]. Additionally, it has been used to find listening behavioral patterns in four different age groups and to evaluate the improvement of a music recommendation model by using demographic, profiling, and contextual features [22].

We plan to expand the MLHD by collecting more listening data. This is a good idea in the eventual case that Last.fm stops providing this data or a full shutdown of the service. Also, the data collected may be added to the ListenBrainz project, an initiative of the MetaBrainz Foundation with the goal of allowing listeners to preserve their existing music listening histories in Last.fm.

Although we aimed to collect data from a large group of listeners of varied demographics—thus helping to overcome biases from previous user-driven and data-driven research—the listening data we collected may be also biased. For example, the age distribution of listeners show that the dataset is skewed towards late adolescent and early adult listeners. However, since this group will be older in a few years from now, and younger generations are already born into a digital era, we suppose that this trend may be different in a few years, and the large skew towards listeners in their early twenties may be less significant. In any case, the MLHD has a much larger amount of data than any of the studies of the datasets reviewed in Section 2, and so it allows for the undertaking of studies with balanced populations of listeners of each age.

We also acknowledge that a limitation of conducting data-centric studies using data collected from listening interactions with media players and music streaming services is the fact that it is hard to know if listeners actually chose the music item they were exposed to, or it was the recommendation engine or shuffle algorithm of a music streaming service the one that suggested the music item. As a result, it is hard to say if a specific scrobble reflected the actual preference of a listener, or if it registered what was recommended by a recommendation or shuffle algorithm. However, Wikström [23] pointed out that ubiquitous access to music services with recommendation algorithms is how the majority of people are actually experiencing music in the new music economy. Hence, the study of music preference nowadays cannot separate self-chosen music from algorithmically generated playlists and suggestions. These two approaches are occurring at the same time, and so both have to be considered in order to obtain insights about listening behaviors and music preferences. We hope the dataset we introduced will be useful for doing large-scale research on user modeling, music preference, and recommendation.

The MLHD can be accessed and downloaded at `http://ddmal.music.mcgill.ca/research/musiclisteninghistoriesdataset`.

5. ACKNOWLEDGEMENTS

We are extremely grateful of all users of Last.fm that have agreed to make their data available for non-commercial use, and also to the Last.fm service, which has collected and offered this data since 2002 uninterruptedly, helping the field of music informatics research to move forward.

This research has been supported by BecasChile Bicentenario, Comisión Nacional de Ciencia y Tecnología, Gobierno de Chile, and the Social Sciences and Humanities Research Council of Canada. Important parts of this work used ComputeCanada's High Performance Computing resources.

6. REFERENCES

[1] Dominikus Baur, Jennifer Büttgen, and Andreas Butz. Listening factors: A large-scale principal components analysis of long-term music listening histories. In *Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems*, pages 1273–6, Austin, TX, 2012.

[2] Pauwke Berkers. Gendered scrobbling: Listening behaviour of young adults on Last.fm. *Interactions: Studies in Communication & Culture*, 2(3):279–96, 2010.

[3] Jennifer Büttgen. What's in a history? A large-scale statistical analysis of Last.fm data. Master's thesis, Ludwig-Maximilians-Universität München, München, Germany, 2010.

[4] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Last.fm web 2.0 dataset. In *2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems*, RecSys 2011, Chicago, IL, 2011.

[5] Òscar Celma. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer-Verlag Berlin Heidelberg, Berlin, Germany, 2010.

[6] Wei Chai and Barry Vercoe. Using user models in music information retrieval systems. In *Proceedings of the 1st International Symposium on Music Information Retrieval*, Plymouth, MA, 2000.

[7] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The Yahoo! music dataset and KDD-cup'11. *Journal of Machine Learning Research*, 18:3–18, 2011.

[8] EMI Group Limited. EMI Million Interview Dataset, 2012. http://musicdatascience.com/emi-million-interview-dataset/. Accessed 18 February 2017.

[9] David Hauger, Markus Schedl, Andrej Košir, and Marko Tkalčič. The million musical tweets dataset: What can we learn from microblogs. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.

[10] Perfecto Herrera, Zuriñe Resa, and Mohamed Sordo. Rocking around the clock eight days a week: An exploration of temporal patterns of music listening. In *Proceedings of the 1st ACM RecSys Workshop on Music Recommendation and Discovery,*, Barcelona, Spain, 2010.

[11] IFPI. IFPI global music report 2015. Annual report, International Federation of the Phonographic Industry, 2015.

[12] IFPI. IFPI global music report 2016. Annual report, International Federation of the Phonographic Industry, 2016.

[13] Brian McFee, Thierry Bertin-Mahieux, Daniel P. W. Ellis, and Gert R. G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference on World Wide Web*, pages 909–16, Lyon, France, 2012.

[14] Sergio Oramas, Vito Claudio Ostuni, Tomasso Di Noia, Xavier Serra, and Eugenio Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology*, 8(2):1–21, 2016.

[15] Sergio Oramas, Vito Claudio Ostuni, and Gabriel Vigliensoni. Sound and music recommendation with knowledge graphs (dataset), 2016. http://hdl.handle.net/10230/27495. Accessed 18 April 2017.

[16] Chan Ho Park and Minsuk Kahng. Temporal dynamics in music listening behavior: A case study of online music service. In *9th IEEE International Conference on Computer and Information Science*, pages 573–8, Kaminoyama, Japan, 2010.

[17] Robert Prey. *Musica analytica: The datafication of listening*, chapter 3, pages 31–48. Palgrave Macmillan UK, London, United Kingdom, 2016.

[18] Paul Rutter. *The Music Industry Handbook*. Media Practice. Routledge, Oxfordshire, United Kingdom, 2 edition, 2016.

[19] Markus Schedl. *Leveraging microblogs for spatiotemporal music information retrieval*, volume 7814 of *Lecture Notes in Computer Science*, pages 796–9. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[20] Markus Schedl. The LFM-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, pages 103–10. ACM, 2016.

[21] Gabriel Vigliensoni and Ichiro Fujinaga. Identifying time zones in a large dataset of music listening logs. In *Proceedings of the 1st International Workshop on Social Media Retrieval and Analysis*, pages 27–32. ACM, 2014.

[22] Gabriel Vigliensoni and Ichiro Fujinaga. Automatic music recommendation systems: Do demographic, profiling, and contextual features improve their performance? In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 94–100, New York City, NY, 2016.

[23] Patrik Wikström. *The Music Industry: Music in the Cloud*. Polity Press, Cambrige, UK, 2nd edition, 2013.

[24] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. #nowplaying music dataset: Extracting listening behavior from twitter. In *Proceedings of the 1st International Workshop on Internet-Scale Multimedia Management*, pages 21–6, Orlando, FL, 2014.

ARTIST PREFERENCES AND CULTURAL, SOCIO-ECONOMIC DISTANCES ACROSS COUNTRIES: A BIG DATA PERSPECTIVE

Meijun Liu
The University of Hong Kong
meijun@connect.hku.hk

Xiao Hu
The University of Hong Kong
xiaoxhu@hku.hk

Markus Schedl
Johannes Kepler University Linz
markus.schedl@jku.at

ABSTRACT

Users in different countries may have different music preferences, possibly due to geographical, economic, linguistic, and cultural factors. Revealing the relationship between music preference and cultural socio-economic differences across countries is of great importance for music information retrieval in a cross-country or cross-cultural context. Existing works are usually based on small samples in one or several countries or take only one or two socio-economic aspects into account. To bridge the gap, this study makes use of a large-scale music listening dataset, LFM-1b with more than one billion music listening logs, to explore possible associations between a variety of cultural and socio-economic measurements and artist preferences in 20 countries. From a big data perspective, the results reveal: 1) there is a highly uneven distribution of preferred artists across countries; 2) the linguistic differences among these countries are positively associated with the distances in artist preferences; 3) country differences in three of the six cultural dimensions considered in this study have positive influences on the difference of artist preferences among the countries; and 4) geographical and economic distances among the countries have no significant relationship with their artist preferences.

1. INTRODUCTION

Probing the relationship between cultural and socio-economic difference and the cross-country difference in music preferences not only matters in music information retrieval (MIR), but also brings important cues to understand the difference in cultural and socio-economic aspects among countries. Against the background of the world's diversity in many cultural and socio-economic aspects, research aiming to uncover cross-country differences in the field of music recommendation and retrieval is seeing increasing attention [22, 30]. It is widely acknowledged that music information behavior is inherently a kind of cultural behavior, shaped by the culture and other socio-economic factors [32, 56]. A growing body of literature demonstrated that different cultures or different countries have disparity in music information behaviors, e.g. music retrieval, management and consumption, and music mood judgment [30, 39, 44]. This is also true in music preferences [25, 56]. In this case, a question naturally arises: which kind of cultural and socio-economic background might possibly be responsible for the difference of music preferences among countries? It is thus necessary to have an in-depth understanding of the differences in music preferences across different countries and of how these differences are mirrored by cultural and socio-economic factors. Answers to these questions can facilitate constructing cross-cultural MIR systems, and promoting music recommendation and retrieval results by taking into account cultural and the socio-economic background of users [56]. Furthermore, this paper also contributes to improving the knowledge of the differences in customs, traditions, cultural values, and other socio-economic factors among countries.

Existing literature provides little evidence of the exact relationships between the cross-country differences in a variety of cultural and socio-economic factors and those in artist preference. Furthermore, limited literature investigated which cultural dimension reflects the inter-country difference in artist preferences. Even fewer previous studies were based on large-scale user-generated datasets. This situation calls for more studies in this regard. Therefore, we investigate in this paper the following research questions:

RQ1: How do artist preferences differ across countries?

RQ2: Does the inter-country difference in artist preferences depend on the geographic, economic, linguistic, and cultural distances among countries?

RQ3: Which cultural dimension can reflect the difference in artist preferences across countries?

Inspired by this research gap and the scientific importance, this paper seeks to probe whether the differences among countries in music taste rely on any factors in the cultural and socio-economic dimensions, through applying descriptive analysis, Kruskal-Wallis variance analysis and Quadratic Assignment Procedure (QAP) on a large dataset with more than one billion listening records, the LFM-1b dataset [49]. To our knowledge, this is a first work that explores relationship between the inter-country difference in artist preferences and a variety of cultural and socio-economic differences among countries.

2. RELATED WORK

Related work can be categorized into research that investigates the connection between music preferences and socio-economic factors, and that between music preferences and cultural dimensions. A recent study analyzed the country-specific music preferences. However, it did not investigate the influential factors of music preferences [50].

In recent years, due to the availability of large-scale music listening data, users geospatial context for music recommendation has received increasing attention [53]. However, there is limited literature on directly exploring the relationship between music preferences and geographic locations. Before large-scale music listening datasets have been published, several works involved location-related information. Researchers proposed a mobile music recommender system, Lifetrack, that enables a playlist based on the location and other information in the users environment [47]. More recently, researchers found that drawing on the information of listeners geographic location help promote music recommendation [43, 52]. Although some works have been done, the authors suggested that combing cultural regions with geographical distances may better explain differences in music taste [53].

Economic status seems to have a potential influence on musical preferences. Cultural consumption is closely related to individuals social status that in turn is directly interrelated with the amount of income. According to Bourdieu's class theory, high-status groups have more cultural capital which is defined as knowledge and appreciation of highbrow culture, and the possession of high or low income in people's childhood tend to shape their taste [7]. Empirical evidence showed that the cultural taste of the high-status group is distinct from people in other classes. For example, people belonging to the high-status group frequently visit museums, classical concerts, the theater and so forth [15, 31, 33]. In the field of music research, there exist some evidence that support the connection between income and music preferences as well. Cutler found that preference for classical music tended to grow with income [13]. Duncan, Herrington and Capella also suggested that the music taste of upper-income individuals is different from their counterparts with low income or/and with only high school education [18]. It has been found that high socio-economic status positively impacts musical openness that is related to the acceptance for diverse music [57].

Culture is a well-discussed factor in music information research, compared to other socio-economic aspects. From the perspectives of sociology, psychology and behavior science, researchers believe that general behaviors and preferences are shaped by culture [32]. In the field of MIR, retrieval methods that consider cultural differences in music perception and consumption are highly desirable [39]. In recent years, taking cultural factors into account has become a frequently-used strategy in MIR research to explore users music need at the country level [23]. Researchers found that preference for music mood altered significantly between countries, implying that utilizing geographic information of users could facilitate further studies [48]. More recently, it has been suggested that the country-based diversity pattern of music listening is associated with some cultural dimensions presented in Hofstede's theory on cultural dimensions [23]. Specifically, researchers found that users in countries with high scores in the culture dimension of power distance tended to show less diversity in the artists and genres they listened to. Oppositely, individualism dimension was negatively correlated with music diversity. Furthermore, the correlation between long-term orientation and artist diversity was considered negative.

Based on small-sample data obtained from surveys, previous studies provided more direct evidence to show the influence of language on listeners reactions to and comments on music. Empirical results presented that there was a significantly positive correlation between familiarity with a language and attitude toward the language in songs [1]. Specifically, it was reported that some children responded to foreign-language music with negative judgment [38]. In a study which focused on language in the context of songs, it was observed that English-speaking students preferred pop songs performed in English to those with Spanish or Chinese lyrics [2]. By examining undergraduate non-music majors world music preferences, researchers found that the breadth and length of studying foreign languages were related to a high degree of world music preferences [26].

In a nutshell, the cultural and socio-economic variables we selected are thought of as potentially correlative factors of music preferences. However, the exact relationship between these factors and music preferences under a cross-country context still remains unclear. First, current studies are limited to small samples collected from surveys or questionnaires. Besides, such self-reported responses can be subjective. In other words, there is scarce literature that investigates this research question using objective datasets in a large scale. Second, most existing studies simply include one or two socio-economic factors, leaving many potentially relevant aspects unexplored. Third, extant studies ignored the discussion of the association between socio-economic factors and music taste in a context of multiple countries or multiple cultures, since a majority of them paid attention to individuals in a single cultural environment. Fourth, among the few studies on relationship between music preferences and socio-economic factors (e.g. geographical location), the conclusions are often ambiguous and indecisive. To bridge the gaps, this study aims to uncover the relationship between music preferences and cultural and socio-economic factors at the country level using a large-scale and user-generated dataset.

3. DATA AND METHODOLOGY

3.1 LFM-1b Dataset

This study uses the open dataset LFM-1b [1] [49]. This dataset includes more than one billion music listening

[1] www.cp.jku.at/datasets/LFM-1b. The period during which the data was collected ranges from January 2013 to August 2014.

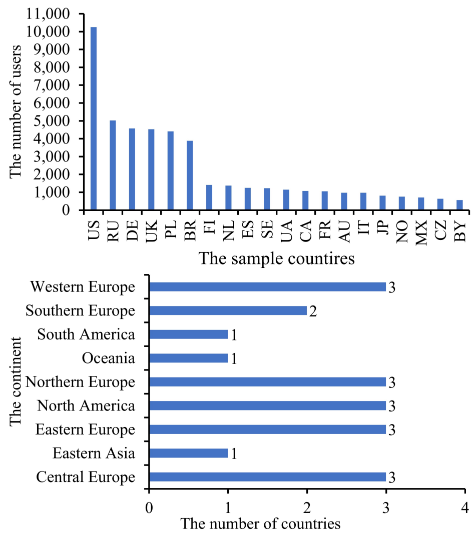


Figure 1: Number of users in the 20 sample countries (top) and the continents where countries are located (bottom).(Country code: US: United States, RU: Russia, DE: Germany, UK: United Kingdom, PL:Poland, BR: Brazil, FI: Finland, NL: Netherlands, ES: Spain, SE: Sweden, UA: Ukraine, CA: Canada, FR: France, AU: Australia, IT: Italy, JP: Japan, NO: Norway, MX: Mexico, CZ: Czech Republish, BY: Belarus)

events created by 120,322 users and enables us to conduct a large-scale analysis in music listening behaviors. It is noteworthy that only 54.13% of users in the LFM-1b dataset provide information on their nationality and the distribution of users across countries is very unbalanced. To avoid possible negative effects on our analysis, we eliminate countries with less than 1% of users in LFM-1b and only use the remaining 20 countries in this study. Finally, we obtained a dataset including 46,619 users with 678,640,512 listening events that cover 2,259,103 unique artists.

The distribution of users in the sampling countries and which continent these countries belong to are shown in Figure 1. It indicates that most of them are located in Europe and America, with one country in Asia, Oceania, and South America respectively.

3.2 Modeling Country-specific Diversity in Artist Preferences

In this study, we used the coefficient of variation (CV) and Gini coefficient to measure and compare the diversity of artist preferences across the countries. Coefficient of variation is a standardized measure of dispersion of the frequency distribution, which is defined as the ratio of the standard deviation to the average of a variable [20]. CV has been frequently used for comparing diversity or inequality in groups [3, 5]. The Gini index enables us to examine the inequality of artist listening frequency in each country [46, 60]. We adapt the definition of the Gini coefficient for a country c to our task and calculate it as shown in Equation 1,

$$G_c = \left| 1 + \frac{1}{N} - \frac{2}{m \times N^2} \sum_i (N - O_i + 1) \cdot y_i \right| \quad (1)$$

where N is the number of artists listened to by users in country c; y_i is the listening count of artist i in country c; O_i is the inverse rank of y_i when sorting the values y_i for all artists i in country c, and m is arithmetic mean of listening counts across the N artists.

We adopted the Kruskal-Wallis (KW) non-parametric analysis of variance as the primary tool to probe whether there is a significant difference among countries in the frequency of artist listening. After performing the Shapiro-Wilk test, it was observed that the data exhibited non-normal distribution, and thus non-parametric analysis of variance was adopted [19]. A follow-up test was carried out to find out which pairs of countries have significant differences [9, 29, 54].

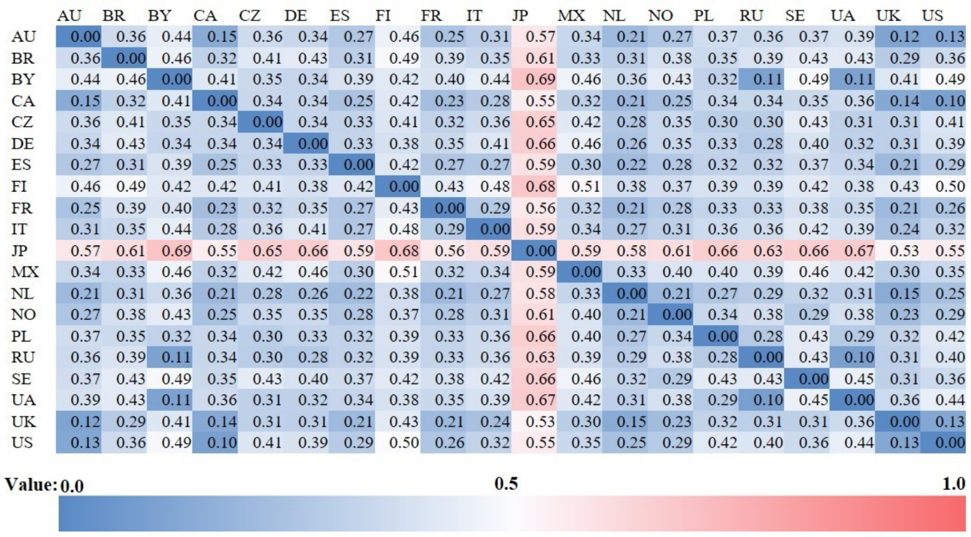
To avoid possible bias caused by the disequilibrium of listening counts across countries, we also normalized the listening frequency of each artist in a country against the total listening count of that country. In other words, we look into not only the raw listening counts but also the normalized listening count of each artist.

3.3 Modeling Country Distances in terms of Artist Preferences, Cultural and Socio-economic Dimensions

The *distance of artist preference* among countries is the dependent variable in this study. Based on the data of listening events in LFM-1b, we calculated the cosine distance of artist preferences among countries. Specifically, each country is represented by a vector of artists, with each dimension of the vector being the number of times the corresponding artist was listened to by users in this country. Then, the cosine distance between each pair of vectors was calculated. The results are shown in Figure 2. Notably, the distances between Japan and all other countries are substantially higher (> 0.5) than those between other pairs of countries, making Japan an outlier, which is in line with previous studies [51].

In this study, the cultural and socio-economic distance between countries is measured by the following aspects: geographic, economic, linguistic, and cultural distance. *Geographic distance* is the geodetic distance between the capital cities calculated by Vincenty's equations and on the basis of the latitude and longitude, i.e., the length of the shortest curve between two points along the surface of the Earth [58]. We define *economic distance* as the difference of gross domestic product (GDP) per capita between countries, calculated based on the data obtained from World Bank [2]. For *linguistic distance* between countries, we regard the language used by the largest population in a country as the main language in that country. On the website of the Central Intelligence Agency (CIA), [3] the language and

[2] http://databank.worldbank.org/data/home.aspx
[3] https://www.cia.gov/library/publications/the-world-factbook/fields/2098.html



	AU	BR	BY	CA	CZ	DE	ES	FI	FR	IT	JP	MX	NL	NO	PL	RU	SE	UA	UK	US
AU	0.00	0.36	0.44	0.15	0.36	0.34	0.27	0.46	0.25	0.31	0.57	0.34	0.21	0.27	0.37	0.36	0.37	0.39	0.12	0.13
BR	0.36	0.00	0.46	0.32	0.41	0.43	0.31	0.49	0.39	0.35	0.61	0.33	0.31	0.38	0.35	0.39	0.43	0.43	0.29	0.36
BY	0.44	0.46	0.00	0.41	0.35	0.34	0.39	0.42	0.40	0.44	0.69	0.46	0.36	0.43	0.32	0.11	0.49	0.11	0.41	0.49
CA	0.15	0.32	0.41	0.00	0.34	0.34	0.25	0.42	0.23	0.28	0.55	0.32	0.21	0.25	0.34	0.34	0.35	0.36	0.14	0.10
CZ	0.36	0.41	0.35	0.34	0.00	0.34	0.33	0.41	0.32	0.36	0.65	0.42	0.28	0.35	0.30	0.30	0.43	0.31	0.31	0.41
DE	0.34	0.43	0.34	0.34	0.34	0.00	0.33	0.38	0.35	0.41	0.66	0.46	0.26	0.35	0.33	0.28	0.40	0.32	0.31	0.39
ES	0.27	0.31	0.39	0.25	0.33	0.33	0.00	0.42	0.27	0.27	0.59	0.30	0.22	0.28	0.32	0.32	0.37	0.34	0.21	0.29
FI	0.46	0.49	0.42	0.42	0.41	0.38	0.42	0.00	0.43	0.48	0.68	0.51	0.38	0.37	0.39	0.39	0.42	0.38	0.43	0.50
FR	0.25	0.39	0.40	0.23	0.32	0.35	0.27	0.43	0.00	0.29	0.56	0.32	0.21	0.28	0.33	0.33	0.38	0.35	0.21	0.26
IT	0.31	0.35	0.44	0.28	0.36	0.41	0.27	0.48	0.29	0.00	0.59	0.34	0.27	0.31	0.36	0.36	0.42	0.39	0.24	0.32
JP	0.57	0.61	0.69	0.55	0.65	0.66	0.59	0.68	0.56	0.59	0.00	0.59	0.58	0.61	0.66	0.63	0.66	0.67	0.53	0.55
MX	0.34	0.33	0.46	0.32	0.42	0.46	0.30	0.51	0.32	0.34	0.59	0.00	0.33	0.40	0.40	0.39	0.46	0.42	0.30	0.35
NL	0.21	0.31	0.36	0.21	0.28	0.26	0.22	0.38	0.21	0.27	0.58	0.33	0.00	0.21	0.27	0.29	0.32	0.31	0.15	0.25
NO	0.27	0.38	0.43	0.25	0.35	0.35	0.28	0.37	0.28	0.31	0.61	0.40	0.21	0.00	0.34	0.38	0.29	0.38	0.23	0.29
PL	0.37	0.35	0.32	0.34	0.30	0.33	0.32	0.39	0.33	0.36	0.66	0.40	0.27	0.34	0.00	0.28	0.43	0.29	0.32	0.42
RU	0.36	0.39	0.11	0.34	0.30	0.28	0.32	0.39	0.33	0.36	0.63	0.39	0.29	0.38	0.28	0.00	0.43	0.10	0.31	0.40
SE	0.37	0.43	0.49	0.35	0.43	0.40	0.37	0.42	0.38	0.42	0.66	0.46	0.32	0.29	0.43	0.43	0.00	0.45	0.31	0.36
UA	0.39	0.43	0.11	0.36	0.31	0.32	0.34	0.38	0.35	0.39	0.67	0.42	0.31	0.38	0.29	0.10	0.45	0.00	0.36	0.44
UK	0.12	0.29	0.41	0.14	0.31	0.31	0.21	0.43	0.21	0.24	0.53	0.30	0.15	0.23	0.32	0.31	0.31	0.36	0.00	0.13
US	0.13	0.36	0.49	0.10	0.41	0.39	0.29	0.50	0.26	0.32	0.55	0.35	0.25	0.29	0.42	0.40	0.36	0.44	0.13	0.00

Figure 2: Heat map of cosine distances among countries based on artist listening frequency

the size of its speakers in a country are obtained to identify the main language in each of the sampling countries. Ethnologue[4] provides the information of the global language family tree, based on which we calculated the linguistic distance between two countries. Consistent with existing literature [21, 37], the linguistic distance between two languages i and j is defined in Equation 2,

$$D_{ij} = 1 - \left(\frac{|N_i \cap N_J|}{\frac{1}{2} \cdot (N_i + N_j)} \right)^{\beta} \qquad (2)$$

where N_i denotes the number of nodes in country i's language tree, N_j analogously. The relative distance between languages which are both included in the same family hinges on the value of β. According to the experience in other studies [21, 37], we set $\beta = 0.5$. For example, in the language family tree, English belongs to the following branch: Indo-European $>$ Germanic $>$ West $>$ English while Swedish is classified into this branch: Indo-European $>$ Germanic $>$ North Germanic $>$ East Scandinavian $>$ Continental Scandinavian $>$ Swedish. The distance between these two languages is approximately 0.368 in that they have four and six nodes separately, sharing two common nodes. In order to quantify the *cultural distance* between countries, we calculate the distance of scores between countries in each of Hofstede's cultural dimensions[5] [28]: **Power distance index** (PDI) refers to the extent to which the less powerful members accept and expect that power is distributed unequally; **Individualism** (IDV) defines the degree of preference for a loosely-knit or tightly-knit social framework; **Masculinity** (MAS) refers to the degree of preference for achievement, heroism, assertiveness, and material rewards for success; **Uncertainty avoidance** (UAI) expresses the attitude of individuals towards uncertainty and ambiguity; **Long-term orientation** (LTO) describes to which degree a society ties the past with the present and future actions or challenges; **Indulgence** (IND) measures the happiness of a society.

[4] `https://www.ethnologue.com/`

[5] `https://geert-hofstede.com/national-culture.html`

3.4 Quadratic Assignment Procedure

In this study, we applied the Quadratic Assignment Procedure (QAP) [36, 55] via Double Dekker Semi-partialling [4, 14] to examine the relationship between distance of artist preference across countries, and geographic, economic, linguistic and cultural proximities across countries. In other words, we explore whether the difference among countries in artist listening has a relationship with their differences in the aspects of geographic location, economy, languages and culture. The primary reason for using QAP in this study is to avoid biases caused by autocorrelation of error in the dyadic dataset [55]. In this study, each observation is a pair of countries (i.e., a dyad in network analysis terminologies). Dyads are non-independent because each node in a dyad is connected to other dyads. Therefore, regression methods that assume independent distribution of data such as ordinary least squares (OLS) regression would lead to biased estimators [4, 6, 41]. In contrast, QAP explicitly takes into account dependence between dyads as well as autocorrelation of errors in the dyadic dataset. It is frequently used in regression analyses on network and relationship datasets [8, 10, 11, 16, 36]. The independent variables are the six matrices of the between-country distances in the six cultural dimensions, whereas the dependent variable is the matrix of inter-country distance on the artist preferences. We control the geographical distance (GEO), economic distance (ECO) and linguistic distance (LAN) among the countries. Besides, we calculated the mean variance inflation factor score (1.79), which is far lower than critical point 10, implying that multicolinearity can be ignored in this study.

4. RESULTS

4.1 Differences among Countries in terms of Artist Preferences

Table 1 presents statistics of the artist listening histories across countries, including the average number of listening events to an artist, the standard deviation (SD), coefficient of variation (CV), the number of unique artists (Uniq.#) listened to by listeners in each country and Gini coefficient (Gini).

As can be seen from Table 1, users from the US and Russia listen to a large number of unique artists, far exceeding other countries. Furthermore, the high CV values for US, BR, RU, PL and UK imply that listeners from these countries listen to a wider range of artists, compared to users from other countries.

Country	Mean	SD	CV	Uniq.#	Gini
US	182.16	2930.63	16.09	747004	96.44%
RU	114.46	1795.28	15.69	632460	96.38%
DE	134.46	1746.64	12.99	474874	96.35%
UK	127.68	1709.56	13.39	456456	95.39%
PL	209.95	3280.39	15.62	362155	95.21%
BR	203.11	3263.22	16.07	267186	95.15%
NL	83.57	833.97	9.98	256895	94.53%
UA	68.38	743.48	10.87	249287	93.98%
SE	102.94	1095.51	10.64	229714	93.38%
FI	114.41	1230.23	10.75	213645	93.10%
FR	71.79	595.26	8.29	207878	93.05%
CA	93.83	817	8.71	191728	92.97%
ES	82.3	717.39	8.72	190671	92.96%
JP	63.44	548.04	8.64	185128	92.95%
BY	51.78	469.26	9.06	166465	92.74%
NO	78.52	669.54	8.53	165663	92.66%
AU	89.02	759.82	8.54	164145	92.52%
IT	81.28	783.94	9.64	156599	92.03%
MX	73.17	753.87	10.3	144930	91.51%
CZ	87.8	743.38	8.47	127726	91.15%
Mean	**105.70**	**1274.32**	**11.05**	**279530**	**93.72%**

Table 1: Statistics of artist listening frequency across sample countries ranked by the number of unique artists

In general, Gini indices are high for all countries, meaning users' preferences for an artist varied a lot. In particular, the inequality of artist listening is most noticeable in the US, Brazil, Poland, and the UK, which is consistent with the CV results.

When comparing frequency of artist listening across countries, the result of the Kruskal-Wallis test shows a statistically significant difference ($p < 0.01$) among the sampling countries. After conducting a follow-up pairwise comparison, we find that significant differences on artist preferences exist between all 190 country pairs, except for BR and AU, CZ and AU, CZ and BR, JP and CZ, MX and FR, PL and CA, RU and NO, SE and FI, SE and FR.

4.2 QAP Correlation and Regression Results

We run two models to test the relationship between artist preference (as represented by artist listening frequencies) distance among countries, and the geographical, economic, linguistic, and cultural distances among them. The QAP correlation coefficients among the variables are reported in Table 2, and the regression results are in Table 3. For comparison, only the control variables are included in model 1 and we added the independent variables to model 2. The adjusted R^2 in the two models are significant: 0.594 and 0.643 in model 1 and 2, respectively. In other words, nearly 59.4% of the variance in the matrix of the artist preference distances among countries can be explained by their distances in the geographic, economic and linguistic aspects; and 64.3% of the variance can be explained in model 2 with the addition of cultural distances.

The distance among countries in term of main languages is positively associated with their distance in artist preferences (r = 0.745 in Table 2). In model 2, the coefficient of linguistic distance among countries is significant and positive ($\beta = 0.68$; $p < 0.001$). Furthermore, three dimensions of cultural distance among countries have positive effects on their artist preference distance: masculinity ($\beta = 0.13$; $p < 0.05$), long-term orientation (β=0.12; $p < 0.01$), and indulgence (β=0.14; $p < 0.05$).

Besides, the regression results in both model 1 and model 2 reveal that economic distance has no significant impact on the artist preference distance on the country level. Geographic distance has a significant impact on the dependent variable in model 1, but becomes insignificant when cultural distances are included in model 2.

5. DISCUSSION

We summarize our main findings in the following. The *distribution of music listening behavior across artists is highly uneven*. In particular, substantial inequality of artist preferences is found in the US, Brazil, Poland, Russia, and the UK. In comparing across the countries, there are significant distinctions in artist preferences within most of country pairs.

The *distance between the main languages used in countries is positively associated with the distance in their artist preferences*. This result could be attributed to the fact that familiarity is a key factor that influences music preference [17, 27]. Familiarity not only refers to having heard a music piece somewhere before, but can also be reflected by the degree of familiarity with the language in the songs [1]. Listeners may be less familiar with music sung in languages they know little about, and thus they may be less likely to listen to that kind of music.

Among the six cultural dimensions, *masculinity, long-term orientation, and indulgence distances between countries have positive correlations with their distances in artist preferences.* First, *masculinity* indicates the degree to which a culture delineates gender roles, and a masculine culture clearly differentiates the social expectations on males and females [42]. Previous literature pointed out that a huge gender difference in both the expression and perception of mood could be found in cultures high in masculinity. Other researchers also demonstrated that masculinity can explain the gender difference in personality traits [12]. It is generally agreed that music listening behavior and

	ARTIST	GEO	ECO	LAN	PDI	IDV	MAS	UAI	LTO	
ARTIST	1									
GEO	0.248	1								
ECO	0.122	-0.017	1							
LAN	0.745***	0.066	0.118	1						
PDI	0.149	-0.034	0.516***	0.211	1					
IDV	0.215	0.354*	0.37**	0.136	0.458**	1				
MAS	0.34*	-0.056	0.102	0.317*	0.005	-0.106	1			
UAI	0.144	-0.101	0.352**	0.241*	0.566**	0.266*	0.12	1		
LTO	0.267**	0.266**	0.016	0.081	0.019	0.112	0.01	0.025	1	
IND	0.269*	0.112	0.334**	0.14	0.416**	0.326**	-0.037	0.398**	0.346**	1

Table 2: QAP correlation coefficients (Note: significance levels: *: $p < 0.05$; **: $p < 0.01$; ***: $p < 0.001$)

Variable	Model 1	Model 2
GEO	0.200*	0.133
ECO	0.040	0.003
LAN	0.727***	0.683***
PDI		-0.035
IDV		0.063
MAS		0.131*
UAI		-0.074
LTO		0.122**
IND		0.140*
Adjusted R^2	0.594 ***	0.642***
N of Obs	380	380

Table 3: The QAP regression result. (Note that all coefficients presented are standardized coefficients. Significance levels: ***$p < 0.001$,**$p < 0.01$, *$p < 0.05$)

emotion strongly interact with each other [34, 35]. Moreover, the correlation between personality and music behavior is documented in empirical studies [24, 45]. Consequently, it is possible that on the country level, the music preference difference and the cultural difference in masculinity interacted through the gender differences in terms of emotion and personality traits.

Second, prior studies offer evidence that people in countries scoring low in *long-term orientation* have a lower preference for listening to diverse artists since they value steadfastness and believe that traditions are to be honored and kept [23]. In other words, people in short-term oriented cultures may prefer to listen to more traditional music, and their music listening behavior is possibly more conservative. Furthermore, it is recently found that individuals in countries with long-term orientation tend to be more patient [59]. This characteristics may not only influence business activities, but also generate different listening behaviors across countries. For instance, in long-term oriented countries, people may be more likely to have the patience to listen to slow and long music. Future studies can further explore and test these hypotheses.

Third, in countries scoring high on *indulgence*, people tend to have more freedom in controlling their daily lives and in choosing the way to enjoy life. Given that listening to music is often regarded as an important entertainment activity, the cultural difference in the dimension of indulgence can possibly affect people's choices of music, and thus bringing about the differences in music preferences across countries.

In the final regression model (model 2 in Table 3), there is no significant association between geographical and economic distance on the music preference distance across countries. Perhaps geographical distance is no longer a barrier for people to access various music in today's highly connected information society. Therefore geolocation plays a less significant role in music preference compared to linguistic and cultural differences among countries. In terms of economic distance, although on the individual level, it is confirmed in the literature [40] that music preferences vary by the income level, this seems questionable on the country level. This discrepancy might be related to the correlation between people's cultural behaviors and social status [40]. On an individual level, income is related to social status which in turn can influence one's music preference. However, on the country level, people's social status ranges a lot in any single country and has virtually no relationship with the GDP per capita of a country. Consequently, economic distance among countries cannot explain differences in music preferences.

6. CONCLUSION AND FUTURE WORK

In this study, we applied descriptive statistical analysis, Krusal-Wallis variance analysis, and Quadratic Assignment Procedure on the LFM-1b dataset, to reveal the association between the distance of a variety of cultural and socio-economic aspects among countries, and the cross-country difference in artist preference.

Findings of this study contribute to the literature of music listening behaviors and preferences, particularly from the cross-country perspective. By analyzing one of the largest datasets in the field, we aim to draw conclusions that are representative and generalizable. Multiple factors in the cultural, linguistic, geographic, and economic aspects were analyzed, and the results can potentially help design new strategies of MIR systems in the cross-country and cross-cultural context. Future studies can compare cross-country differences on other facets of music such as genre and mood.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments. This study is partially supported by the Austrian Science Fund (FWF): P25655 and by a seed research grant funded by University of Hong Kong.

8. REFERENCES

[1] Carlos R Abril. Multicultural dimensions and their effect on children's responses to pop songs performed in various languages. *Bulletin of the Council for Research in Music Education*, pages 37–51, 2005.

[2] Carlos R Abril and Patricia J Flowers. Attention, preference, and identity in music listening by middle school students of different linguistic backgrounds. *Journal of Research in Music Education*, 55(3):204–219, 2007.

[3] Arthur G Bedeian and Kevin W Mossholder. On the use of the coefficient of variation as a measure of diversity. *Organizational Research Methods*, 3(3):285–297, 2000.

[4] Geoffrey G Bell and Akbar Zaheer. Geography, networks, and knowledge flow. *Organization Science*, 18(6):955–972, 2007.

[5] RB Bendel, SS Higgins, JE Teberg, and DA Pyke. Comparison of skewness coefficient, coefficient of variation, and gini coefficient as inequality measures within populations. *Oecologia*, 78(3):394–400, 1989.

[6] Stephen P Borgatti, Martin G Everett, and Linton C Freeman. Ucinet for windows: Software for social network analysis. 2002.

[7] Pierre Bourdieu. *Distinction: A social critique of the judgement of taste*. Harvard University Press, 1984.

[8] Val Burris. Interlocking directorates and political cohesion among corporate elites 1. *American Journal of Sociology*, 111(1):249–283, 2005.

[9] Herve M Caci et al. Kwallis2: Stata module to perform kruskal-wallis test for equality of populations. *Statistical Software Components*, 1999.

[10] Junho H Choi, George A Barnett, and BUM-SOO CHON. Comparing world city networks: a network analysis of internet backbone and air transport intercity linkages. *Global Networks*, 6(1):81–99, 2006.

[11] David T Connolly. An improved annealing scheme for the qap. *European Journal of Operational Research*, 46(1):93–100, 1990.

[12] Paul Costa Jr, Antonio Terracciano, and Robert R McCrae. Gender differences in personality traits across cultures: robust and surprising findings., 2001.

[13] B Cutler. North american demographics. *American Demographics*, 1992.

[14] David Dekker, David Krackhardt, and Tom AB Snijders. Sensitivity of mrqap tests to collinearity and autocorrelation conditions. *Psychometrika*, 72(4):563–581, 2007.

[15] Paul DiMaggio and John Mohr. Cultural capital, educational attainment, and marital selection. *American journal of sociology*, 90(6):1231–1261, 1985.

[16] Michael Dreiling and Derek Darves. Corporate unity in american trade policy: A network analysis of corporate-dyad political action 1. *American Journal of Sociology*, 116(5):1514–63, 2011.

[17] Kevin Droe. Music preference and music education: A review of literature. *Update: Applications of Research in Music Education*, 24(2):23–32, 2006.

[18] J Duncan Herrington and Louis M Capella. Practical applications of music in service settings. *Journal of Services Marketing*, 8(3):50–65, 1994.

[19] Alan C Elliott and Linda S Hynan. A sas® macro implementation of a multiple comparison post hoc test for a kruskal–wallis analysis. *Computer methods and programs in biomedicine*, 102(1):75–80, 2011.

[20] BS Everitt. The cambridge dictionary of statistics cambridge university press. *Cambridge, UK Google Scholar*, 1998.

[21] James D Fearon. Ethnic and cultural diversity by country. *Journal of Economic Growth*, 8(2):195–222, 2003.

[22] Bruce Ferwerda and Markus Schedl. Investigating the relationship between diversity in music consumption behavior and cultural dimensions: A cross-country analysis. In *Proc. of the 1st Workshop on SOAP*, 2016.

[23] Bruce Ferwerda, Andreu Vall, Marko Tkalcic, and Markus Schedl. Exploring music diversity needs across countries. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, pages 287–288. ACM, 2016.

[24] Bruce Ferwerda, Emily Yang, Markus Schedl, and Marko Tkalcic. Personality traits predict music taxonomy preferences. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2241–2246. ACM, 2015.

[25] Flavio Figueiredo, Bruno Ribeiro, Christos Faloutsos, Nazareno Andrade, and Jussara M Almeida. Mining online music listening trajectories. In *ISMIR*, pages 688–694, 2016.

[26] C Victor Fung. Undergraduate nonmusic majors' world music preference and multicultural attitudes. *Journal of Research in Music Education*, 42(1):45–57, 1994.

[27] Russell P Getz. The effects of repetition on listening response. *Journal of Research in Music Education*, 14(3):178–192, 1966.

[28] Geert Hofstede. Cultures and organizations. intercultural cooperation and its importance for survival. software of the mind. *London: Mc Iraw-Hill*, 1991.

[29] DG Housley, KA Berube, TP Jones, S Anderson, FD Pooley, and RJ Richards. Pulmonary epithelial response in the rat lung to instilled montserrat respirable dusts and their major mineral components. *Occupational and environmental medicine*, 59(7):466–472, 2002.

[30] Xiao Hu and Jin Ha Lee. A cross-cultural study of music mood perception between american and chinese listeners. In *ISMIR*, pages 535–540. Citeseer, 2012.

[31] Volker Kirchberg. Museum visitors and non-visitors in germany: A representative survey. *Poetics*, 24(2):239–258, 1996.

[32] Shinobu Kitayama and Hyekyung Park. Cultural shaping of self, emotion, and well-being: How does it work? *Social and Personality Psychology Compass*, 1(1):202–222, 2007.

[33] Wim Knulst and Gerbert Kraaykamp. Trends in leisure reading: Forty years of research on reading in the netherlands. *Poetics*, 26(1):21–41, 1998.

[34] Stefan Koelsch. Brain correlates of music-evoked emotions. *Nature Reviews Neuroscience*, 15(3):170–180, 2014.

[35] Stefan Koelsch, Thomas Fritz, Karsten Müller, Angela D Friederici, et al. Investigating emotion with music: an fmri study. *Human brain mapping*, 27(3):239–250, 2006.

[36] David Krackardt. Qap partialling as a test of spuriousness. *Social networks*, 9(2):171–186, 1987.

[37] David D. Laitin and Rajesh Ramachandran. Language policy and human development. *American Political Science Review*, 110(3):457–480, 2016.

[38] Albert LeBlanc. Effects of style, tempo, and performing medium on children's music preference. *Journal of Research in Music Education*, 29(2):143–156, 1981.

[39] Jin Ha Lee, J Stephen Downie, and Sally Jo Cunningham. Challenges in cross-cultural/multilingual music information seeking. 2005.

[40] Lawrence W Levine. *Highbrow/lowbrow*, volume 1986. Harvard University Press, 1988.

[41] Bing Liu, Songshan Sam Huang, and Hui Fu. An application of network analysis on tourist attractions: The case of xinjiang, china. *Tourism Management*, 58:132–141, 2017.

[42] David Matsumoto. Cultural influences on the perception of emotion. *Journal of Cross-Cultural Psychology*, 20(1):92–105, 1989.

[43] Joshua L Moore, Thorsten Joachims, and Douglas Turnbull. Taste space versus the world: an embedding analysis of listening habits and geography. In *ISMIR*, pages 439–444, 2014.

[44] Esa Nettamo, Mikko Nirhamo, and Jonna Häkkilä. A cross-cultural study of mobile music: retrieval, management and consumption. In *Proceedings of the 18th Australia conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments*, pages 87–94. ACM, 2006.

[45] Jodi L Pearson and Stephen J Dollinger. Music preference correlates of jungian types. *Personality and individual differences*, 36(5):1005–1008, 2004.

[46] Martin Ravallion. Income inequality in the developing world. *Science*, 344(6186):851–855, 2014.

[47] Sasank Reddy and Jeff Mascia. Lifetrak: music in tune with your life. In *Proceedings of the 1st ACM international workshop on Human-centered multimedia*, pages 25–34. ACM, 2006.

[48] Markus Schedl. Leveraging microblogs for spatiotemporal music information retrieval. In *European Conference on Information Retrieval*, pages 796–799. Springer, 2013.

[49] Markus Schedl. The LFM-1b Dataset for Music Retrieval and Recommendation. In *Proceedings of the ACM International Conference on Multimedia Retrieval (ICMR)*, New York, USA, April 2016.

[50] Markus Schedl. Investigating country-specific music preferences and music recommendation algorithms with the lfm-1b dataset. *International Journal of Multimedia Information Retrieval*, 6(1):71–84, 2017.

[51] Markus Schedl. Investigating country-specific music preferences and music recommendation algorithms with the LFM-1b dataset. *International Journal of Multimedia Information Retrieval*, 6(1):71–84, 2017.

[52] Markus Schedl and Dominik Schnitzer. Hybrid retrieval approaches to geospatial music recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 793–796. ACM, 2013.

[53] Markus Schedl, Andreu Vall, and Katayoun Farrahi. User geospatial context for music recommendation in microblogs. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 987–990. ACM, 2014.

[54] Sidney Siegel. Castellan. nonparametric statistics for the social sciences, 1988.

[55] William B. Simpson. Qap: The quadratic assignment procedure. *North American Stata Users Group Meeting*, 2001.

[56] Marcin Skowron, Florian Lemmerich, Bruce Ferwerda, and Markus Schedl. Predicting genre preferences from cultural and socio-economic factors for music retrieval. In *European Conference on Information Retrieval*, pages 561–567. Springer, 2017.

[57] Koen Van Eijck. Social differentiation in musical taste patterns. *Social forces*, 79(3):1163–1185, 2001.

[58] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.

[59] Mei Wang, Marc Oliver Rieger, and Thorsten Hens. How time preferences differ: Evidence from 53 countries. *Journal of Economic Psychology*, 52:115–135, 2016.

[60] Sean D Young, Caitlin Rivers, and Bryan Lewis. Methods of using real-time social media technologies for detection and remote monitoring of hiv outcomes. *Preventive medicine*, 63:112–115, 2014.

Oral Session 4

Multimodal

LEARNING AUDIO – SHEET MUSIC CORRESPONDENCES FOR SCORE IDENTIFICATION AND OFFLINE ALIGNMENT

Matthias Dorfer[*] **Andreas Arzt**[*†] **Gerhard Widmer**[*†]
[*]Department of Computational Perception, Johannes Kepler University Linz, Austria
[†]The Austrian Research Institute for Artificial Intelligence (OFAI), Austria
matthias.dorfer@jku.at

ABSTRACT

This work addresses the problem of matching short excerpts of audio with their respective counterparts in sheet music images. We show how to employ neural network-based cross-modality embedding spaces for solving the following two sheet music-related tasks: retrieving the correct piece of sheet music from a database when given a music audio as a search query; and aligning an audio recording of a piece with the corresponding images of sheet music. We demonstrate the feasibility of this in experiments on classical piano music by five different composers (Bach, Haydn, Mozart, Beethoven and Chopin), and additionally provide a discussion on why we expect multi-modal neural networks to be a fruitful paradigm for dealing with sheet music and audio at the same time.

1. INTRODUCTION

Traditionally, automatic methods for linking audio and sheet music data are based on a common mid-level representation that allows for comparison (i.e., computation of distances or similarities) of time points in the audio and positions in the sheet music. Examples of mid-level representations are symbolic descriptions, which involve the error-prone steps of automatic music transcription on the audio side [2, 4, 12, 20] and optical music recognition (OMR) on the sheet music side [3, 9, 19, 24], or spectral features like pitch class profiles (chroma features), which avoid the explicit audio transcription step but still depend on variants of OMR. For examples of the latter approach see, e.g., [8, 11, 15].

In this paper we present a methodology to *directly* learn correspondences between complex audio data and images of the sheet music, circumventing the problematic definition of a mid-level representation. Given short snippets of audio and their respective sheet music images, a cross-modal neural network is trained to learn an embedding space in which both modalities are represented as 32-dimensional vectors. which can then be compared, e.g., via their cosine distance. Essentially, the neural network replaces the complete feature computation process (on both sides) by learning a transformation of data from the audio and from the sheet music to a common vector space.

The idea of matching sheet music and audio with neural networks was recently proposed in [6]. The approach presented here goes beyond that in several respects. First, the network in [6] requires both sheet music and audio as input at the same time to predict which location in the sheet image best matches the current audio excerpt. We address a more general scenario where both input modalities are required only at training time, for learning the relation between score and audio. This requires a different network architecture that can learn two separate projections, one for embedding the sheet music and one for embedding the audio. These can then be used independently of each other. For example, we can first embed a reference collection of sheet music images using the image embedding part of the network, then embed a query audio and search for its nearest sheet music neighbours in the joint embedding space. This general scenario is referred to as *cross-modality retrieval* and supports different applications (two of which are demonstrated in this paper). The second aspect in which we go beyond [6] is the sheer complexity of the musical material: while [6] was restricted to simple monophonic melodies, we will demonstrate the power of our method on real, complex pieces of classical music.

We demonstrate the utility of our approach via preliminary results on two real-world tasks. The first is *piece identification:* given an audio rendering of a piece, the corresponding sheet music is identified via cross-modal retrieval. (We should note here that for practical reasons, in our experiments the audio data is synthesized from MIDI – see below). The second task is *audio-to-sheet-music alignment*. Here, the trained network acts as a complex distance function for given pairs of audio and sheet music snippets, which in turn is used by a dynamic time warping algorithm to compute an optimal sequence alignment.

Our main contributions, then, are (1) a methodology for learning cross-modal embedding spaces for relating audio data and sheet music data; (2) data augmentation strategies which allow for training the neural network for this complex task even with a limited amount of data; and (3) first results on two important MIR tasks, using this new approach.

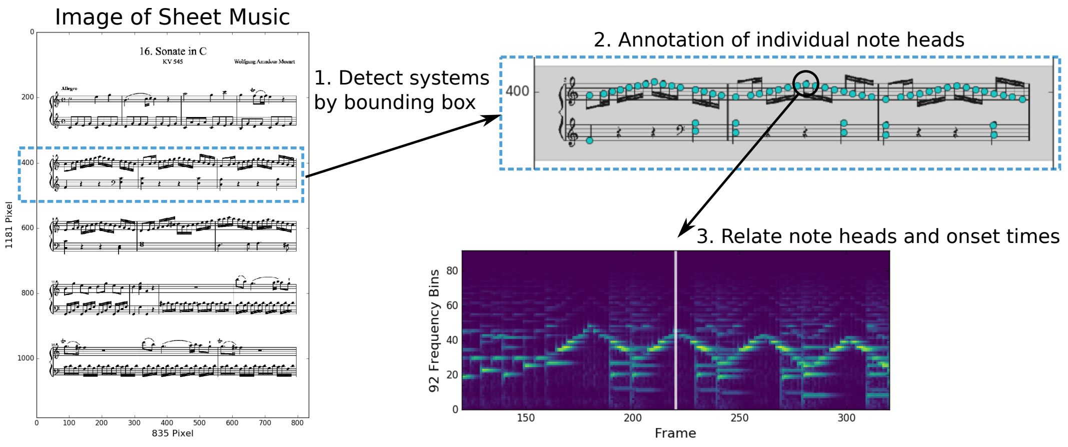


Image of Sheet Music
1. Detect systems by bounding box
2. Annotation of individual note heads
3. Relate note heads and onset times
1181 Pixel
835 Pixel
92 Frequency Bins
Frame

Figure 1. Work flow for preparing the training data (correspondences between sheet music images and the respective music audio). Given the relation between the note heads in the sheet music image and their corresponding onset times in the audio signal we sample audio-sheet-music pairs for training our networks. Figure 2 shows four examples of such training pairs.

2. DESCRIPTION OF DATA

Our approach is built around a neural network designed for learning the relationship between two different data modalities. The network learns its behaviour solely from the examples shown for training. As the presented data is crucial to make this class of models work, we dedicate this section to describing the underlying data as well as the necessary preparation steps needed to generate training examples for optimizing our networks.

2.1 Sheet-Music-Audio Annotation

As already mentioned, we want to address two tasks: (1) sheet music (piece) identification from audio queries and (2) offline alignment of a given audio with its corresponding sheet music image. Both are multi-modal problems involving sheet music images and audio. We therefore start by describing the process of producing the ground truth for learning correspondences between a given score and its respective audio. Figure 1 summarizes the process.

Step one is the localization of staff systems in the sheet music images. In particular, we annotate bounding boxes around the individual systems. Given the bounding boxes we detect the positions of the note heads within each of the systems [1]. The next step is then to relate the note heads to their corresponding onset times in the audio.

Once these relations are established, we know for each note head its location (in pixel coordinates) in the image, and its onset time in the audio. Based on this relationship we cut out corresponding snippets of sheet music images (in our case 180×200 pixels) and short excerpts of audio represented by log-frequency spectrograms (92 bins $\times$ 42 frames). Figure 2 shows four examples of such sheet-music-audio correspondences; these are the pairs presented to our multi-modal networks for training.

[1] We of course do not annotate all of the systems and note heads by hand but use a note head and staff detector to support this tasks (again a neural network trained for this purpose).

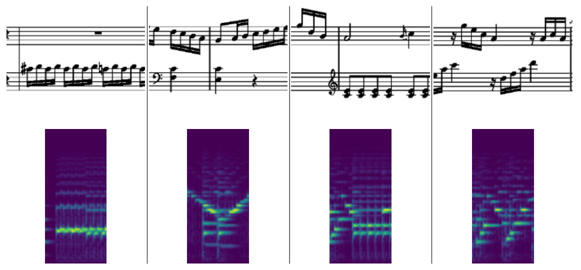


Figure 2. Sheet-music audio correspondences presented to the network for retrieval embedding space learning.

2.2 Composers, Sheet Music and Audio

For our experiments we use classical piano music by five different composers: Mozart (14 pieces), Bach (16), Beethoven (5), Haydn (4) and Chopin (1). To give an impression of the complexity of the music, we have, for instance, Mozart piano sonatas (K.545 1st mvt., K.331 3rd) and symphony transcriptions for piano (Symphony 40 K.550 1st), preludes and fugues from Bach's WTC, Beethoven piano sonata movements and Chopin's Nocturne Op.9 No.1. In terms of experimental setup we will use *only* the 13 pieces of Mozart for training, Mozart's K.545 mvt.1 for validation, and all remaining pieces for testing. This results in 18,432 correspondences for training, 989 for validating, and 11,821 for testing. Our sheet music is collected from *Musescore* [2] where we selected only scores having a 'realistic' layout close to the typesetting of professional publishers [3]. The reason for using Musescore for initial experiments is that along with the sheet music (as *pdf* or image files) Musescore also provides the corresponding *midi* files. This allows us to synthesize the music for each piece of sheet music and to com-

[2] `https://musescore.com`

[3] This is an example of a typical score we used for the experiment (Beethoven Sonata Op.2 No.1): `https://musescore.com/classicman/scores/55331`

pute the exact note onset times from the midis, and thus to establish the required sheet-music audio correspondences.

In terms of audio preparation we compute log-frequency spectrograms of the audios, with a sample rate of 22.05kHz, a FFT window size of 2048 samples, and a computation rate of 20 frames per second. For dimensionality reduction we apply a normalized 16-band logarithmic filterbank allowing only frequencies from $30Hz$ to $16kHz$, which results in 92 frequency bins.

2.3 Data Augmentation

To improve the generalization ability of the resulting networks, we propose several data augmentation strategies specialized to score images and audio. In machine learning, *data augmentation* refers to the application of (realistic) data transformations in order to synthetically increase the effective size of the training set. We already emphasize at this point that data augmentation is a crucial component for learning cross-modality representations that generalize to unseen music, especially when little data is available.

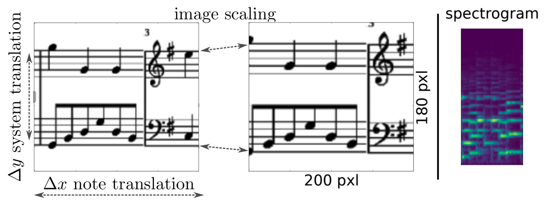


Figure 3. Overview of image augmentation strategies. The size of the sliding image window remains constant (180 × 200 pixels) but its content changes depending on the augmentations applied. The spectrogram remains the same for the augmented image versions.

For **sheet image augmentation** we apply three different transformations, summarized in Figure 3. The first is *image scaling* where we resize the image between 95 and 105% of its original size. This should make the model robust to changes in the overall dimension of the scores. Secondly, in Δy *system translation* we slightly shift the system in the vertical direction by $\Delta y \in [-5, 5]$ pixels. We do this as the system detector will not detect each system in exactly the same way and we want our model to be invariant to such translations. In particular, it should not be the absolute location of a note head in the image that determines its meaning (pitch) but its relative position with respect to the staff. Finally, we apply Δx *note translation*, meaning that we slightly shift the corresponding sheet image window by $\Delta x \in [-5, 5]$ pixels in the horizontal direction.

In terms of **audio augmentation** we render the training pieces with three different sound fonts and additionally vary the tempo between 100 and 130 beats per minute (bpm). The test pieces are all rendered at a rate of 120 bpm using an *additional unseen sound font*. The test set is kept fixed to reveal the impact of the different data augmentation strategies.

3. AUDIO - SHEET MUSIC CORRESPONDENCE LEARNING

This section describes the underlying learning methodology. As mentioned above, the core of our approach is a cross-modality retrieval neural network capable of learning relations between short snippets of audio and sheet music images. In particular, we aim at learning a joint embedding space of the two modalities in which to perform nearest-neighbour search. One method for learning such a space, which has already proven to be effective in other domains such as text-to-image retrieval, is based on the optimization of a pairwise ranking loss [14, 22]. Before explaining this optimization target, we first introduce the general architecture of our correspondence learning network.

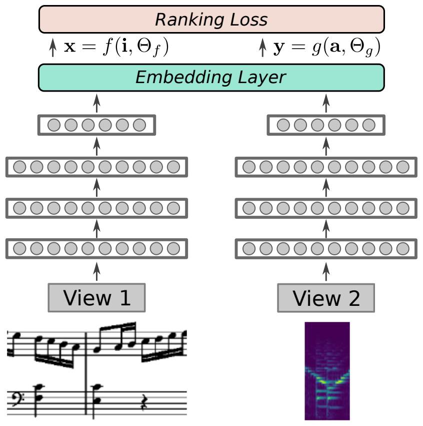

Ranking Loss
$\mathbf{x} = f(\mathbf{i}, \Theta_f)$
$\mathbf{y} = g(\mathbf{a}, \Theta_g)$
Embedding Layer
View 1
View 2

Figure 4. Architecture of correspondence learning network. The network is trained to optimize the similarity (in embedding space) between corresponding audio and sheet image snippets by minimizing a pair-wise ranking loss.

As shown in Figure 4 the network consists of two separate pathways f and g taking two inputs at the same time. Input one is a sheet image snippet $\mathbf{i}$ and input two is an audio excerpt $\mathbf{a}$. This means in particular that network f is responsible for processing the image part of an input pair and network g is responsible for processing the audio. The output of both networks (represented by the *Embedding Layer* in Figure 4) is a k-dimensional vector representation encoding the respective inputs. In our case the dimensionality of this representation is 32. We denote these hidden representations by $\mathbf{x} = f(\mathbf{i}, \mathbf{\Theta_f})$ for the sheet image and $\mathbf{y} = g(\mathbf{a}, \mathbf{\Theta_g})$ for the audio spectrogram, respectively, where Θ_f and Θ_g are the parameters of the two networks.

Given this network design, we now explain the pairwise ranking objective. Following [14] we first introduce a *scoring function* $s(\mathbf{x}, \mathbf{y})$ as the cosine similarity $\mathbf{x} \cdot \mathbf{y}$ between the two hidden representations ($\mathbf{x}$ and $\mathbf{y}$ are scaled to have unit norm). Based on this scoring function we optimize the following pairwise ranking objective (‘hinge loss’):

$$\mathcal{L}_{rank} = \sum_{\mathbf{x}} \sum_{k} \max\{0, \alpha - s(\mathbf{x}, \mathbf{y}) + s(\mathbf{x}, \mathbf{y}_k)\} \quad (1)$$

In our application $\mathbf{x}$ is an embedded sample of a sheet image snippet, $\mathbf{y}$ is the embedding of the matching audio excerpt and $\mathbf{y}_k$ are the embeddings of the *contrastive* (mismatching) audio excerpts (in practice all remaining samples of the current training batch). The intuition behind this loss function is to encourage an embedding space where the distance between matching samples is lower than the distance between mismatching samples. If this condition is roughly satisfied, we can then perform cross-modality retrieval by simple nearest neighbour search in the embedding space. This will be explained in detail in Section 4.

The network itself is implemented as a VGG- style convolution network [21] consisting of 3×3 convolutions followed by 2×2 max-pooling as outlined in detail in Table 1. The final convolution layer computes 32 feature maps and is subsequently processed with a global average pooling layer [16] that produces a 32-dimensional vector for each input image and spectrogram, respectively. This is exactly the dimension of our retrieval embedding space. At the top of the network we put a canonically correlated embedding layer [7] combined with the ranking loss described above. In terms of optimization we use the *adam* update rule [13] with an initial learning rate of 0.002. We watch the performance of the network on the validation set and halve the learning rate if there is no improvement for 30 epochs. This procedure is repeated ten times to finetune the model.

Table 1. Audio-sheet-music model. BN: Batch Normalization [10], ELU: Exponential Linear Unit [5], MP: Max Pooling, Conv(3, pad-1)-16: 3×3 convolution, 16 feature maps and padding 1.

Sheet-Image 180 × 200	Audio (Spectrogram) 92 × 42
2×Conv(3, pad-1)-12	2× Conv(3, pad-1)-12
BN-ELU + MP(2)	BN-ELU + MP(2)
2×Conv(3, pad-1)-24	2× Conv(3, pad-1)-24
BN-ELU + MP(2)	BN-ELU + MP(2)
2×Conv(3, pad-1)-48	2× Conv(3, pad-1)-48
BN-ELU + MP(2)	BN-ELU + MP(2)
2×Conv(3, pad-1)-48	2× Conv(3, pad-1)-48
BN-ELU + MP(2)	BN-ELU + MP(2)
Conv(1, pad-0)-32-BN-LINEAR	Conv(1, pad-0)-32-BN-LINEAR
GlobalAveragePooling	GlobalAveragePooling
Embedding Layer + Ranking Loss	

4. EVALUATION OF AUDIO - SHEET CORRESPONDENCE LEARNING

In this section we evaluate the ability of our model to retrieve the correct counterpart when given an instance of the other modality as a search query. This first set of experiments is carried out on the lowest possible granularity, namely, on sheet image snippets and spectrogram excerpts such as shown in Figure 2. For easier explanation we describe the retrieval procedure from an *audio query point of view* but stress that the opposite direction works in exactly the same fashion. Given a spectrogram excerpt $\mathbf{a}$ as a search query we want to retrieve the corresponding sheet image snippet $\mathbf{i}$. For retrieval preparation we first embed all candidate image snippets $\mathbf{i}_j$ by computing $\mathbf{x}_j = f(\mathbf{i}_j)$ as the output of the image network. In the present case, these candidate snippets originate from the 26 unseen test pieces by Bach, Haydn, Beethoven and Chopin. In a second step we embed the given query audio as $\mathbf{y} = g(\mathbf{a})$ using the audio pathway g of the network. Finally, we select

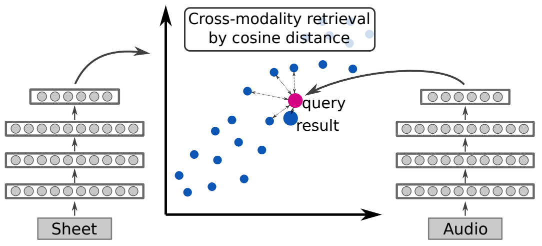


Figure 5. Sketch of sheet-music-from-audio retrieval. The blue dots represent the embedded candidate sheet music snippets. The red dot is the embedding of an audio query. The larger blue dot highlights the closest sheet music snippet candidate selected as retrieval result.

the audio's nearest neighbour $\mathbf{x}_j$ from the set of embedded image snippets as

$$\mathbf{x}_j = \arg\min_{\mathbf{x}_i} \left(1.0 - \frac{\mathbf{x}_i \cdot \mathbf{y}}{||\mathbf{x}_i|| \, ||\mathbf{y}||} \right) \qquad (2)$$

based on their pairwise cosine distance. Figure 5 shows a sketch of this retrieval procedure.

In terms of experimental setup we use the 13 pieces of Mozart for training the network, and the pieces of the remaining composers for testing. As evaluation measures we compute the *Recall@k (R@k)* as well as the *Median Rank (MR)*. The *R@k* rate (high is better) is the percentage of queries which have the correct corresponding counterpart in the first k retrieval results. The *MR* (low is better) is the median position of the target in a cosine-similarity-ordered list of available candidates.

Table 2 summarizes the results for the different data augmentation strategies described in Section 2.3. The unseen synthesizer and the tempo for the test set remain fixed for all settings. This allows us to directly investigate the influence of the different augmentation strategies. The results are grouped into audio augmentation, sheet augmentation, and applying all or no data augmentation at all. On first sight the retrieval performance appears to be very poor. In particular the *MR* seems hopelessly high in view of our target applications. However, we must remember that our query length is only 42 spectrogram frames ($\approx$ 2 seconds of audio) per excerpt and we select from a set of $11,821$ available candidate snippets. And we will see in the following sections that this retrieval performance is still sufficient to perform tasks such as piece identification. Taking the performance of *no augmentation* as a baseline we observe that all data augmentation strategies help improve the retrieval performance. In terms of audio augmentation we observe that training the model with different synthesizers and varying the tempo works best. From the set of image augmentations, the Δy *system translation* has the highest impact on retrieval performance. Overall we get the best retrieval model when applying *all* augmentation strategies. Note also the large gap between *no augmentation* and *full augmentation*. The median rank, for example, drops from 1042 in case of no augmentation to 168 for full augmentation, which is a substantial improvement.

Audio Augmentation	**R@1**	**R@10**	**R@25**	**MR**
1 Synth, 100-130bpm	0.37	3.73	7.05	771
3 Synth, 120bpm	0.75	6.26	11.52	559
3 Synth, 100-130bpm	0.87	8.23	15.29	332
Sheet Augmentation				
image scaling	0.75	5.60	10.14	524
Δy system translation	0.91	6.57	12.21	449
Δx note translation	0.44	3.66	7.19	808
full sheet augmentation	0.70	5.72	11.03	496
no augmentation	0.33	2.88	5.71	1042
full augmentation	1.70	11.67	21.17	168
random baseline	0.00	0.03	0.21	5923

Table 2. Influence of data augmentation on audio-to-sheet retrieval. For the audio augmentation experiments no sheet augmentation is applied and vice versa. *no augmentation* represents 1 Synth, 120bpm without sheet augmentation.

A final note: for space reasons we only present results on audio-to-sheet music retrieval, but that the opposite direction using image snippets as search query works analogously and shows similar performance.

5. PIECE IDENTIFICATION

Given the above model that learns to express similarities between sheet music snippets and audio excerpts, we now describe how to use this to solve our first targeted task: identifying the respective piece of sheet music when given an entire audio recording as a query (despite the relatively poor recall and MR for individual queries).

5.1 Description of Approach

We start by preparing a **sheet music retrieval database** as follows. Given a set of sheet music images along with their annotated systems we cut each piece of sheet music j into a set of image snippets $\{\mathbf{i}_{ji}\}$ analogously to the snippets presented to our network for training. For each snippet we store its originating piece j. We then embed all candidate image snippets into the retrieval embedding space by passing them through the image part f of the multimodal network. This yields, for each image snippet, a 32-dimensional embedding coordinate vector $\mathbf{x}_{ji} = f(\mathbf{i}_{ji})$.

Sheet snippet retrieval from audio: Given a whole audio recording as a search query we aim at identifying the corresponding piece of sheet music in our database. As with the sheet image we start by cutting the audio (spectrogram) into a set of excerpts $\{\mathbf{a}_1, ..., \mathbf{a}_K\}$ again exhibiting the same dimensions as the spectrograms used for training, and embed all query spectrogram excerpts $\mathbf{a}_k$ with the audio network g. Then we proceed as described in Section 4 and select for each audio its nearest neighbour from the set of all embedded image snippets.

Augmentation	**R1**	**R2**	**R3**	**>R3**
no augmentation	4	7	1	14
full augmentation	24	2	0	0

Table 3. Influence of data augmentation on piece retrieval.

Piece selection: Since we know for each of the image snippets its originating piece j, we can now have the retrieval image snippets $\mathbf{x}_{ji}$ *vote* for the piece. The piece achieving the highest count of votes is our final retrieval result. In our experiments we consider for each query excerpt its top 25 retrieval results for piece voting.

5.2 Evaluation of Approach

Table 3 summarizes the piece identification results on our test set of Bach, Haydn, Beethoven and Chopin (26 pieces). Again, we investigate the influence of data augmentation and observe that the trend of the experiments in Section 4 is directly reflected in the piece retrieval results. As evaluation measure we compute Rk as the number of pieces ranked at position k when sorting the result list by the number of votes. Without data augmentation only four of the 26 pieces are ranked first in the retrieval lists of the respective full audio recording queries. When making use of data augmentation during training, this number increases substantially and we are able to recognize 24 pieces at position one; the remaining two are ranked at position two. Although this is not the most sophisticated way of employing our network for piece retrieval, it clearly shows the usefulness of our model and its learned audio and sheet music representations for such tasks.

6. AUDIO-TO-SHEET-MUSIC ALIGNMENT

As a second usage scenario for our approach we present the task of audio-to-sheet-music alignment. Here, the goal is to align a performance (given as an audio file) to its respective score (as images of the sheet music), i.e., computing the corresponding location in the sheet music for each time point in the performance, and vice versa.

6.1 Description of Approach

For computing the actual alignments we rely on Dynamic Time Warping (DTW), which is a standard method for sequence alignment [18], and is routinely used in the context of music processing [17]. Generally, DTW takes two sequences as input and computes an optimal non-linear alignment between them, with the help of a local cost measure that relates points of the two sequences to each other.

For our task the two sequences to be aligned are the sequence of snippets from the sheet music image and the sequence of audio (spectrogram) excerpts, as described in Section 2.2. The neural network presented in Section 3 is then used to derive a local cost measure by computing the pairwise cosine distances between the embedded sheet

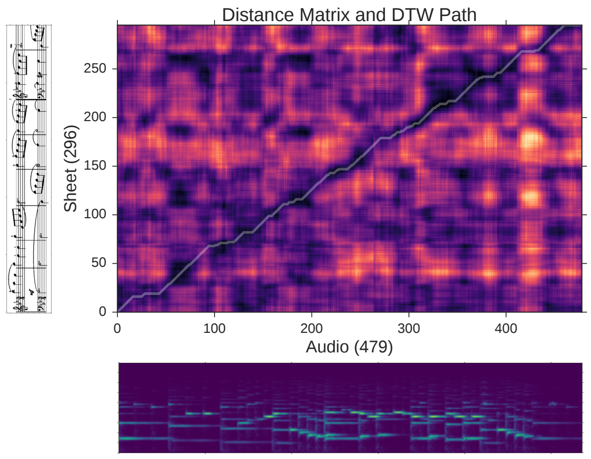


Figure 6. Sketch of audio-to-sheet-music alignment by DTW on a similarity matrix computed on the embedding representation learned by the multi-modal matching network. The white line highlights the path of minimum costs through the sheet music given the audio.

snippets and audio excerpts (see Equation 2). The resulting cost matrix that relates all points of both sequences to each other is shown in Figure 6, for a short excerpt from a simple Bach minuet. Then, the standard DTW algorithm is used to obtain the optimal alignment path.

6.2 Evaluation of Approach

For the evaluation we rely on the same dataset and setup as described above: learning the embedding only on Mozart, then aligning test pieces by Bach, Haydn, Beethoven, Chopin. As evaluation measure we compute the absolute *alignment error* (distance in pixels) of the estimated alignment to its ground truth alignment for each of the sliding window positions. We further normalize the errors by dividing them by the sheet image width to be independent of image resolution. As a naive baseline we compute a linear interpolation alignment which would correspond to a straight line diagonal in the distance matrix in Figure 6. We consider this as a valid reference as we do not consider repetitions for our experiments, yet (in which case things would become somewhat more complicated). We further emphasize that the purpose of this experiment is to provide a proof of concept for this class of models in the context of sheet music alignment tasks, not to compete with existing specialized algorithms for music alignment.

The results are summarized by the boxplots in Figure 7. The median alignment error for the linear baseline is 0.213 normalized image widths ($\approx$ 45 mm in a printed page of sheet music). When computing a DTW path through the distance matrix inferred by our mutimodal audio-sheet-music network this error decreases to 0.041 ($\approx$ 9 mm). Note that values above 1.0 normalized page widths are possible as we handle a piece of sheet music as one single unrolled (concatenated) staff.

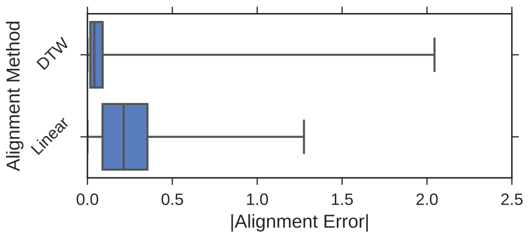


Figure 7. Absolute alignment errors normalized by the sheet image width. We compare the linear baseline with a DTW on the cross-modality distance matrix computed on the embedded audio snippets and spectrogram excerpts.

7. DISCUSSION AND CONCLUSION

We have presented a method for matching short excerpts of audio to their respective counterparts in sheet music images, via a multi-modal neural network that learns relationships between the two modalities, and have shown how to utilize it for two MIR tasks: score identification from audio queries and offline audio-to-sheet-music alignment. Our results provide a proof of concept for the proposed learning-retrieval paradigm and lead to the following conclusions: First, even though little training data is available, it is still possible to use powerful state of the art image and audio models by designing appropriate (task specific) data augmentation strategies. Second, as the best regularizer in machine learning is still a large amount of training data, our results strongly suggest that annotating a truly large dataset will allow us to train general audio-sheet-music-matching models. Recall that for this study we trained on only 13 Mozart pieces, and our model already started to generalize to unseen scores by other composers.

Another aspect of our method is that it works by projecting observations from different modalities into a very low-dimensional joint embedding space. This compact representation is of particular relevance for the task of piece identification as our scoring function – the cosine distance – is a *metric* that permits efficient search in large reference databases [23]. This identification-by-retrieval approach permits us to circumvent solving a large number of local DTW problems for piece identification as done, e.g., in [8].

For now, we have demonstrated the approach on sheet music of realistic complexity, but with synthesized audio (this was necessary to establish the ground truth). The next challenge will be to deal with real audio and real performances, with challenges such as asynchronous onsets, pedal, and varying dynamics.

Finally, we want to stress that our claim is by no means that our proposal in its current stage is competitive with engineered approaches [8, 11, 15] or methods relying on symbolic music or reference performances. These methods have already proven to be useful in real world scenarios, with real performances [1]. However, considering the progress that has been made in terms of score complexity (compared for example to the simple monophonic music used in [6]) we believe it is a promising line of research.

8. ACKNOWLEDGEMENTS

This work is supported by the Austrian Ministries BMVIT and BMWFW, and the Province of Upper Austria via the COMET Center SCCH, and by the European Research Council (ERC Grant Agreement 670035, project CON ESPRESSIONE). The Tesla K40 used for this research was donated by the NVIDIA corporation.

9. REFERENCES

[1] Andreas Arzt, Harald Frostel, Thassilo Gadermaier, Martin Gasser, Maarten Grachten, and Gerhard Widmer. Artificial intelligence in the concertgebouw. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[2] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 121–124, Kyoto, Japan, 2012.

[3] Donald Byrd and Jakob Grue Simonsen. Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. *Journal of New Music Research*, 44(3):169–195, 2015.

[4] Tian Cheng, Matthias Mauch, Emmanouil Benetos, Simon Dixon, et al. An attack/decay model for piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *International Conference on Learning Representations (ICLR) (arXiv:1511.07289)*, 2015.

[6] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. Towards score following in sheet music images. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[7] Matthias Dorfer, Jan Schlüter, Andreu Vall, Filip Korzeniowski, and Gerhard Widmer. End-to-end cross-modality retrieval with cca projections and pairwise ranking loss. *arXiv preprint (arXiv:1705.06979)*, 2017.

[8] Christian Fremerey, Michael Clausen, Sebastian Ewert, and Meinard Müller. Sheet music-audio identification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[9] Jan Hajič jr, Jirı Novotnỳ, Pavel Pecina, and Jaroslav Pokornỳ. Further steps towards a standard testbed for optical music recognition. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[11] Özgür İzmirli and Gyanendra Sharma. Bridging printed music and audio through alignment using a mid-level score representation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[12] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR) (arXiv:1412.6980)*, 2015.

[14] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint (arXiv:1411.2539)*, 2014.

[15] Frank Kurth, Meinard Müller, Christian Fremerey, Yoon-ha Chang, and Michael Clausen. Automated synchronization of scanned sheet music with audio recordings. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 261–266, 2007.

[16] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[17] Meinard Müller. *Fundamentals of Music Processing*. Springer Verlag, 2015.

[18] Lawrence Rabiner and Bing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall Signal Processing Series, 1993.

[19] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre R. S. Marcal, Carlos Guedes, and Jaime S. Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[20] S. Sigtia, E. Benetos, and S. Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations (ICLR) (arXiv:1409.1556)*, 2015.

[22] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218, 2014.

[23] Stijn Van Dongen and Anton J Enright. Metric distances derived from cosine similarity and pearson and spearman correlations. *arXiv preprint arXiv:1208.3145*, 2012.

[24] Cuihong Wen, Ana Rebelo, Jing Zhang, and Jaime Cardoso. A new optical music recognition system based on combined neural network. *Pattern Recognition Letters*, 58:1–7, 2015.

VIDEO-BASED VIBRATO DETECTION AND ANALYSIS FOR POLYPHONIC STRING MUSIC

Bochen Li **Karthik Dinesh** **Gaurav Sharma** **Zhiyao Duan**
Dept. of Electrical and Computer Engineering, University of Rochester, Rochester, NY, USA
{bochen.li, kdinesh, gaurav.sharma, zhiyao.duan}@rochester.edu

ABSTRACT

In music performance, vibrato is an important artistic effect, where slight variations in pitch are introduced to add expressiveness and warmth. Automatic vibrato detection and analysis, although well studied for monophonic music, has rarely been explored for polyphonic music, because of the challenge in multi-pitch analysis. We propose a video-based approach for detecting and analyzing vibrato in polyphonic string music. Specifically, we capture the fine motion of the left hand of string players through optical flow analysis of video frames. We explore two methods. The first uses a feature extraction and SVM classification pipeline, and the second is an unsupervised technique based on autocorrelation analysis of the principal motion component. The proposed methods are compared with audio-only methods applied to individual instrument tracks separated from original audio mixture using the score. Experiments show that the proposed video-based methods achieve a significantly higher vibrato detection accuracy than the audio-based methods especially in high polyphony cases. Further experiments also demonstrate the utility of the approach in vibrato rate and extent analysis.

1. INTRODUCTION

Vibrato is an important artistic effect in musical performance. Instrument players use vibrato to color a tone and express emotions. Physically, vibrato is generated by pitch modulation of a note in a periodic fashion [23]. Important characteristics of vibrato include *rate* and *extent* of this periodic modulation [8]. These characteristics vary significantly across instruments, cultures, and personal styles. Compared to woodwind and brass instruments, vibrato is more pronounced in strings.

Automatic vibrato detection and analysis is an important topic in music information retrieval (MIR) with broad impacts. It is useful in musicological studies to compare different articulation styles of different performers and instruments [2]. It is critical in expressive performance pedagogy for singing [19] and violin [28]. It also facilitates other MIR tasks such as singing voice extraction [12], harmonic-percussive decomposition [21], and audio-visual source association [16]. Vibrato analysis also provides the statistical basis for vibrato synthesis of musical instruments [13], singing voices [11], and bird songs [4], through which the synthesized sounds are more realistic and expressive.

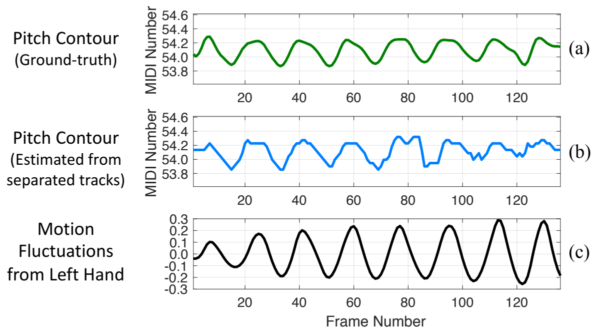


Figure 1. The proposed method tackles the challenging problem of vibrato analysis for polyphonic music by exploiting information from the video to augment audio analysis. (a) The ground-truth pitch contour of a cello vibrato note in a violin-cello duet performance showing a clear vibrato pattern, (b) The estimated pitch contour of this note from the audio mixture using a state-of-the-art score-informed pitch detection method showing corruption due to the interference from the other source, (c) The left hand motion along the fingerboard of the cello player extracted from video analysis is clean and well correlated with the ground-truth pitch contour. The hand motion profile extracted from video is used for vibrato analysis in this paper.

Most of the existing methods for automatic vibrato detection and analysis are audio-based with a focus on monophonic sources, where vibrato can be easily characterized from the pitch trajectory estimated through a monophonic pitch detection algorithm. Methods include thresholding the pitch drift within each note [3], calculating the median distance of the neighboring peaks/troughs of the pitch contour [9], analyzing the spectral peak after a Fourier transform of the pitch contour [25], cross-correlation analysis of frequency/amplitude modulation [26], and a nonlinear sinusoidal decomposition method [27].

Few approaches have focused on polyphonic music, and when they do, they only characterize vibrato of a single source (usually the solo instrument) in the mixture. This is mainly due to the difficulty of reliably estimating simultaneous pitches in polyphonic music [5]. Abeßer et al. [1] proposed a score-informed approach to first estimate the pitch contour of the solo instrument from the audio mixture and then perform vibrato detection and analysis on the pitch contour through autocorrelation. The performance of this approach, however, depends heavily on the pitch estimation performance. Spectrogram-based approaches such as harmonic partial tracking [12] and template convolution [6] reduce the dependency on pitch estimation. However, these operations are still error-prone when harmonics of different sources overlap. To our best knowledge, there is no existing approach for vibrato detection and analysis of multiple simultaneous sources of a polyphonic music mixture, such as a string ensemble. Existing polyphonic audio analysis techniques are not yet sufficient.

Figure 1 shows the limitation of audio-based analysis and motivates the video-based analysis proposed in this paper. In Figure 1 (a), the ground-truth pitch contour of a cello vibrato note in a violin-cello duet performance is shown. This pitch contour is estimated using a monophonic pitch detection algorithm [17] on the isolated (ground truth) signal of the cello note prior to mixing. Vibrato characteristics are clearly observable in this pitch contour. Figure 1 (b) shows the estimated pitch contour of this cello note obtained from a state-of-the-art score-informed source separation and pitch estimation algorithm [7]. Due to the interference from the violin, the estimated pitch contour is corrupted and the vibrato patterns are obscured, especially toward the later time instants represented on the right side of the plot. Note that this example is just a duet of instruments with distinct pitch ranges. For music with higher polyphony using instruments with similar pitch ranges, the estimated pitch contours are further corrupted, making audio-based vibrato detection and analysis unsatisfactory.

For some instruments such as strings, vibrato is often visible from the left hand motion, and this visual information does not degrade as audio information does when polyphony increases. This motivates our proposed approach of vibrato detection and analysis through video-based analysis of the fine motion of the left hand. Figure 1 (c) shows the left hand rolling motion along the principal motion direction (i.e., the fingerboard) of the cello player playing the note. We can see that this motion curve is smooth and it aligns with the ground-truth pitch contour in Figure 1 (a) very well.

The overview of our proposed approach is illustrated in Figure 2. This approach integrates audio, visual and score information, and assumes that the players in the video are well associated with score tracks. Our previous work has addressed the association problem accurately [14]. For each string player, we track the left hand, and then estimate optical flow motion vectors at the pixel level around the left hand. We use audio-score alignment to identify the onset and offset of each note, and perform vibrato detection and analysis on each note from the motion vectors. We develop two approaches for vibrato detection. One uses

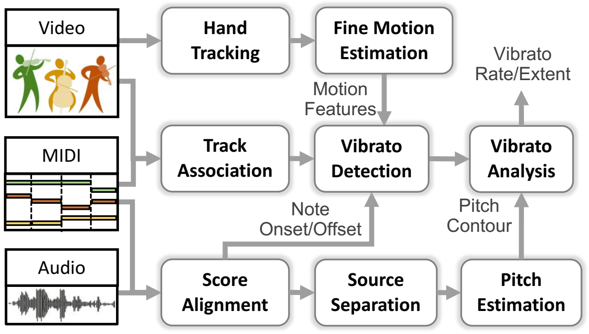


Figure 2. System overview of the proposed video-based vibrato detection and analysis framework.

a Support Vector Machine (SVM) to classify motion features extracted from the pixel-level motion vectors, and the other is based on autocorrelation analysis of the left hand motion along the principal direction (i.e., fingerboard). We further propose a framework to analyze vibrato characteristics: rate and extent. The vibrato rate is estimated from the period of the hand motion curve, and the vibrato extent is estimated from the amplitude of the motion curve after it is scaled to match the estimated noisy pitch contour from score-informed audio analysis.

Experiments are carried on 19 pieces of polyphonic string music from an audio-visual music performance dataset, and the proposed video-based approach is compared with two audio-based baseline methods for vibrato detection. Results show a significant improvement for video-based vibrato detection over the audio-based methods. Further analysis reveals that video-based vibrato detection is robust irrespective of polyphony and instrument types. We further show that the video-based approach is able to estimate the vibrato rate and extent with a deviation from the ground-truth smaller than 1 Hz and 10 musical cents for 90% of the notes, respectively.

2. AUDIO-BASED METHOD

In this section, we introduce an audio-based framework to detect vibrato in polyphonic music to serve as a baseline method. Vibrato can be detected from the pitch contour of each source using either autocorrelation or Fourier transform. However, estimating the pitch contour of each source from the audio mixture is challenging. Inspired by [1], score information can be utilized to alleviate the difficulty of pitch estimation and its assignment to sources.

2.1 Score-informed Pitch Estimation

To utilize the score information for pitch estimation of each source, robust audio-score alignment is required to guarantee the temporal synchronization between the score events and audio articulations. We apply the Dynamic Time Warping (DTW) framework with chroma feature to represent audio and score, as described in [14]. Then the audio mixture is separated using harmonic masking as described in [7]: Pitches of each source are first estimated within two semitones around the quantized score-notated

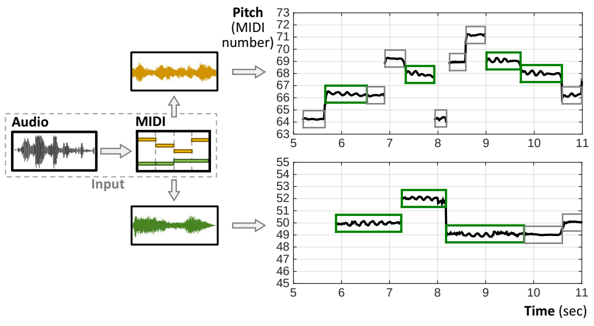


Figure 3. Audio-based vibrato detection. Detected vibrato notes are marked with green rectangles in the pitch trajectories estimated by score-informed pitch estimation.

pitches; Sound sources are then separated by harmonic masking of the pitches in each frame, where the soft masks take into account the harmonic indexes when distributing the mixture signal's energy to overlapping harmonics.

We then re-estimate the pitch contour of each source from its separated signal for vibrato analysis. We again apply the above-mentioned score-informed pitch refinement step to further reduce interference from other sources. The output pitch contour is segmented into notes using the onset/offset information provided by the aligned score. Note that although we can refine the pitches directly from the audio mixture without source separation, it is reported in [16] that the result is more robust on the separated sources. Besides, the availability of separated audio sources is advantageous for other vibrato detection methods that do not rely on pitch contours.

2.2 Vibrato Detection from the Pitch Contour

After obtaining the pitch contour, vibrato detection can be achieved by analyzing the periodic pattern for each note. The pitch contour is analyzed in the MIDI scale, and its DC component is removed by subtracting the average value over the contour. Then we implement two methods to detect the fluctuation rate of the pitch contour: autocorrelation [1] and spectral analysis [25]. For the autocorrelation method, prominent peaks are detected from the autocorrelation function, and the median value of all the neighboring peak distance is used to calculate the fluctuation rate. If the rate is within the range of 3-9 Hz (considering a typical vibrato rate range of [4, 7.5] Hz for strings [10]), the note is detected as vibrato. For the spectral analysis method, we first calculate the magnitude spectrum of the pitch contour of a note through Fourier transform. We then check if the frequency of the maximum peak lies in the rang of 3-9 Hz. Quadratic interpolation is applied in both methods to get a more precise peak location estimation.

The audio-based methods are simple, yet sufficient to detect vibrato in the score-informed fashion. Figure 3 reviews this process and illustrated the detected vibrato notes in green boxes. This approach achieves high detection accuracy in low polyphony settings, but the performance degrades rapidly with increasing polyphony.

3. PROPOSED METHOD

Motivated by the fact that the motion features from the video are correlated with the pitch fluctuations, we propose a video-based vibrato detection and analysis framework. A string instrument player exhibits three kinds of motions: bowing motion to articulate notes, fingering motion to control pitches, and the whole body motion to express musical intentions. Fine periodic fingering motion on the left hand along the fingerboard which changes the length and tension of the string results in vibrato articulations. In this section, we will present the method to extract this fine motion for vibrato detection and analysis.

3.1 Motion Capture

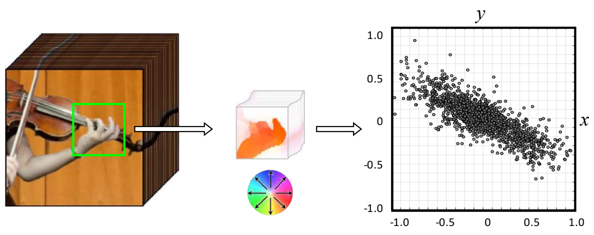


Figure 4. Motion capture results from left hand tracking (left), color encoded pixel velocities (middle), and scatter plot of frame-wise refined motion velocities (right).

The first step is to detect and track the left hand for each player, where the vibrato motions come from. The hand tracking is based on the Kanade-Lucas-Tomasi (KLT) tracker [24] and implemented using the same parameters as presented in [16]. The KLT tracker results in a dynamic region of tracked hand location where we apply the optical flow estimation [22] to obtain the raw motion velocities for each pixel in x and y directions within that region. The motion velocities are spatially averaged as $\mathbf{u}(t) = [u_x(t), u_y(t)]$, where u_x and u_y represents the mean motion velocities in x and y directions respectively, and t is the time index. Notice that the motion velocities in the hand region contain not only the player's fine motion corresponding to vibrato playing, but also his/her large-scale body motions during the performance. In order to eliminate the body movement and obtain a refined motion velocities for vibrato observation, we subtract a moving average of the signal $\mathbf{u}(t)$ from itself, to obtain

$$\mathbf{v}(t) = \mathbf{u}(t) - \bar{\mathbf{u}}(t), \tag{1}$$

where $\bar{\mathbf{u}}(t)$ is the moving average of $\mathbf{u}(t)$ over a 10 frame window. Figure 4 illustrates the original video frame with the tracked hand position, the raw motion velocities from optical flow estimation, and the refined motion velocities $\mathbf{v}(t)$ across all the frames.

3.2 Vibrato Detection from Motion Features

The proposed vibrato detection methods are score informed, where the note onset/offset information from the score is utilized to temporally segment the refined mean motion velocities into $\mathbf{v}^i(t)$, where i is the note index.

To achieve this, each score track needs to be temporally aligned with the video frames, and spatially associated with the players. The first issue is resolved using audio-score alignment, assuming video and audio frames are naturally synchronized. The second issue is addressed as in [14], where player locations are segmented and associated with the score tracks by correlating the bow motions with note events. By utilizing the mean motion velocities and the extracted features, we introduce two methods for vibrato detection. The first method is based on a SVM framework, where each $\mathbf{v}^i(t)$ is classified as vibrato or non-vibrato. The second method is analogous to the audio-based technique, where we perform auto-correlation on the extracted 1-D motion curve along the fingerboard after principal component analysis.

3.2.1 SVM

We train a Support Vector Machine (SVM) as a classification framework for vibrato/non-vibrato detection. We utilize the refined motion velocity segments $\mathbf{v}^i(t) = [v_x^i(t), v_y^i(t)]$ obtained from the procedure explained in Section 3.1. From each $\mathbf{v}^i(t)$, we have velocity components in x and y directions from which 8 dimensional features are extracted. The features are

(a) **Zero crossing rate** (4-D): Vibrato has inherent periodicity when compared to non-vibrato regions. Hence we utilize the zero crossing rate, which is the ratio of total zero crossings to total frame length for $v_x^i(t)$, $v_y^i(t)$ and their auto-correlations, respectively.

(b) **Frequency** (2-D): Vibrato has a typical frequency in the range of 3-9 Hz. Hence we calculate the sum of the absolute value of Fourier coefficients in the 3-9 Hz frequency range for $v_x^i(t)$ and $v_y^i(t)$.

(c) **Auto-correlation peaks** (2-D): Auto-correlation of $v_x^i(t)$ and $v_y^i(t)$ is calculated within a fixed lag of 10 video frames, where total number of local maximum values is utilized as one of the features.

The SVM is trained on tracks which are distinct from the test set using the leave-one-out training strategy. The ground truth vibrato/non-vibrato labels are obtained from ground-truth audio tracks and associated with the corresponding player. For the SVM training algorithm we set the kernel function and scale parameters as radial basis function and automatic scaling, respectively.

3.2.2 PCA

We also propose an unsupervised framework for vibrato detection. From Figure 4, we find that the distribution of the refined motion velocities for vibrato motions are along the fingerboard. So we perform Principal Component Analysis (PCA) on $\mathbf{v}(t)$ across all frames to identify this principal motion direction, and project the motion velocity vectors to this principal direction to obtain a 1-D *motion velocity curve* $V(t)$ as

$$V(t) = \frac{\mathbf{v}(t)^T \tilde{\mathbf{v}}}{\|\tilde{\mathbf{v}}\|}, \quad (2)$$

where $\tilde{\mathbf{v}}$ is the eigenvector corresponding to the largest eigenvalue of the PCA of $\mathbf{v}(t)$. We then perform an integration of the motion velocity curve over time to calculate a *motion displacement curve* as

$$X(t) = \int_0^t V(\tau) d\tau. \quad (3)$$

This displacement curve corresponds to the fluctuation of the vibrating length of the string and hence the pitch fluctuation of the note. Figure 1 (c) shows the motion displacement curve for one vibrato note, which is matched with the ground-truth pitch contour. Similar to the audio-based approach, vibrato is detected through peak picking on the autocorrelation function of the motion displacement curve. Note that different thresholds on the peak picking will affect the sensitivity of the vibrato detection, and we use the uniform threshold for all the notes which yields the best overall results.

3.3 Vibrato Analysis

The video-based method also enables new techniques for analyzing the vibrato features, i.e., vibrato rate and vibrato extent, which describe the speed and the amount by which the pitch is varied. Here extent is defined as the dynamic range of the pitch contour, i.e., the peak-trough difference. Vibrato rate can be directly extracted from video by observing how fast the left hand is rolling along the fingerboard. Again this is solved by analyzing the autocorrelation on the motion displacement curve $X(t)$. Quadratic interpolation is required for peak picking due to the low frame rate of videos. Vibrato extent, however, cannot be estimated by capturing the motion extent, which varies upon different camera distance and angles. Besides, to generate the same vibrato extent, the extent of motion also depends on the vibrato articulation style, the hand position on the fingerboard, and the instrument type. Therefore, we combine the audio analysis together with the extracted motion displacement curve for vibrato extent estimation.

We first estimate the vibration extent of the motion displacement curve as $\hat{w}_e$ by calculating the median of the distance between all the peaks and troughs within each note. We then scale the displacement curve to fit the pitch contour, and the vibrato extent can be calculated from the scaling factor. Specifically, assuming $F(t)$ is the estimated pitch contour (in MIDI number) of the detected vibrato note from audio analysis after subtracting the DC component of itself, the vibrato extent v_e (in musical cents) is estimated as $\hat{v}_e$ as:

$$\hat{v}_e = \arg\min_{v_e} \sum_{t=t^{on}}^{t^{off}} \left| 100 \cdot F(t) - v_e \frac{X(t)}{\hat{w}_e} \right|^2. \quad (4)$$

where $100 \cdot F(t)$ is the pitch contour measured in musical cents; $\frac{X(t)}{\hat{w}_e}$ is the normalized hand displacement curve. Since $X(t)$ is calculated from the video modality, temporal interpolation is applied beforehand to guarantee the same frame rate as the audio, i.e., the hop size for Short-Time Fourier Transform. Note that temporal shift may be applied to $X(t)$ to maximize the cross correlation between $X(t)$ and $F(t)$ to compensate the slight asynchrony between the two modalities (usually within 20ms).

4. EXPERIMENTS

4.1 Dataset and Evaluation Measures

The vibrato detection and analysis system is tested on the URMP dataset [15]. The dataset contains individually recorded audio-visual tracks of various instruments, which are synchronized and assembled to form 44 classical ensemble pieces ranging from duets to quintets. Ground-truth audio tracks and pitch/note annotations are provided in the dataset. The ground-truth annotation of the vibrato rate/extent is acquired by the autocorrelation method as described in Section 2.2 on ground-truth individual audio tracks, and the presence of vibrato is manually examined. For our experiments, we use the 19 ensemble pieces that contains at most one non-string instrument, including 5 duets, 4 trios, 7 quartets, and 3 quintets. Audio is sampled at 48 KHz, and processed with a frame length of 42.7 ms and a hop size of 10 ms for the STFT. Video resolution is 1080P, and the frame rate is 29.97 frames per second.

In the experiments, we evaluate the two proposed video-based methods, i.e., the classification method using SVM framework (Vid-SVM) and autocorrelation analysis on the principal motion component (Vid-PCA). Two audio-based methods described in Section 2.2 are also compared as baseline methods, i.e., peak-picking of the autocorrelation (Aud-AC), and Fourier transform of the pitch contour (Aud-FT). Since the vibrato detection can be viewed as a retrieval task, we compute the note-level precision (P), recall (R), and F-measure (F) using the number of true positives, false positives and false negatives on each track. For the two audio-based methods and the Vid-PCA method, we adjust the peak-picking threshold for a balanced value of precision and recall and fix it for all the tracks. For vibrato rate and extent estimation, we calculate the error between the estimated and ground-truth values on the true positive detections from the Vid-PCA method.

4.2 Results

4.2.1 *Overall Evaluation on Vibrato Detection*

We first evaluate the vibrato detection results using precision, recall and F-measure for the four methods on all of the 57 tracks from the 19 pieces excluding non-string instrument ones, as plotted in Figure 5. Each bar is the average of the 57 tracks. We find that in polyphonic music, both audio-based methods achieve limited performance; lower than 75% for the F-measure. Video-based methods can get a pronounced improvement on the F-measure, which is as high as 90%. The supervised classification method based on SVM further outperforms the unsupervised method, because of the richer features.

4.2.2 *Vibrato Detection Evaluation on Different Cases*

We further investigate how the vibrato detection performance changes along with polyphony and instrument types. Figure 6 illustrates the scatter plot of the vibrato detection F-measure for the four methods (with different colors) in four different polyphony levels corresponding to duets, trios, quartets, and quintets. Each sample point represents the evaluation on one track, and the average

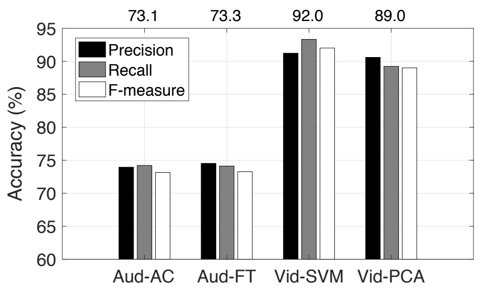


Figure 5. Overall vibrato detection results showing the precision, recall, and F-measure (shown on top) accuracies for 2 audio-based methods and 2 video-based methods.

value in each subset is marked as the red line. We see that the two audio-based methods can reach performance comparable with the video-based methods in low-polyphony pieces, but their performance drops when polyphony increases. This is because of the decreased quality of the pitch contour that is extracted from high-polyphony audio. However, polyphony does not affect the vibrato detection performance for the two video-based methods, since the left hands are always directly observable from visual scene in this dataset. Note that there are several extremely low F-measure values for video-based methods. These come from tracks with plucking-vibrato articulations, where the vibrato is captured from hand motion but is not annotated in the ground truth as its duration and extent are different from the bowing-vibrato articulations.

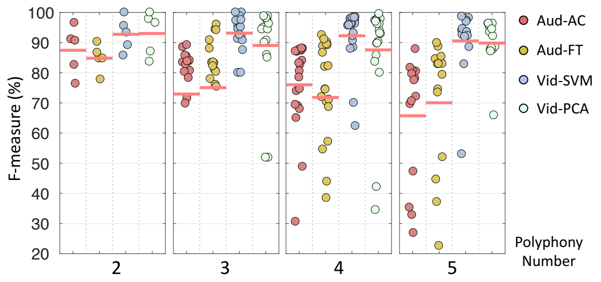


Figure 6. Vibrato detection performance decreases as polyphony increases for audio-based methods, while it stays the same for video-based methods.

Figure 7 further reveals how the vibrato detection results vary for different instruments: violin, viola, cello, and double bass. Again, the audio-based methods are sensitive to instrument types while video-based methods are not. The reason is that the separated track of the low-pitch instrument (such as double bass) is likely to get contaminated by other higher-pitch voices using the harmonic mask method for source separation. In contrast, the vibrato motions for the four different instruments have similar patterns, thus easy to capture by our proposed methods.

4.2.3 *Evaluation of Vibrato Characteristics*

Due to the unsatisfactory performance of audio-based vibrato detection, we evaluate the accuracy of vibrato rate

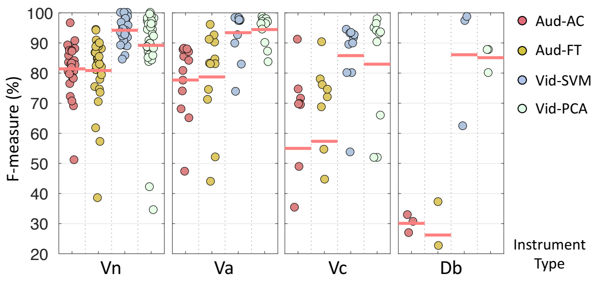


Figure 7. Vibrato detection performance decreases when the fundamental frequency decreases for audio-based methods, while it stays the same for video-based methods.

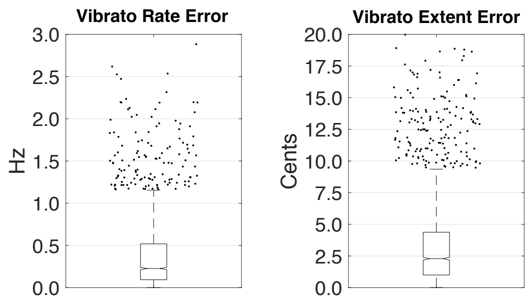


Figure 8. Distribution of vibrato rate and extent estimation error on all notes of all tracks.

and extent estimation only based on the video modality. We conduct this analysis on the true positive detections from the Vid-PCA method, totaling 2290 vibrato notes from the 57 tracks. We calculate the absolute deviation of the estimated value from the ground-truth value for all the notes, and get an average vibrato rate estimation error of 0.38 Hz and median of 0.23 Hz. For vibrato extent, we have an average estimation error of 3.47 cents and a median of 2.29 cents. Figure 8 plots the vibrato rate and extent error distribution for all the notes. We find that for 90% of the vibrato notes, the proposed approach estimates the vibrato rate and extent within an error of 1 Hz and 10 cents, respectively.

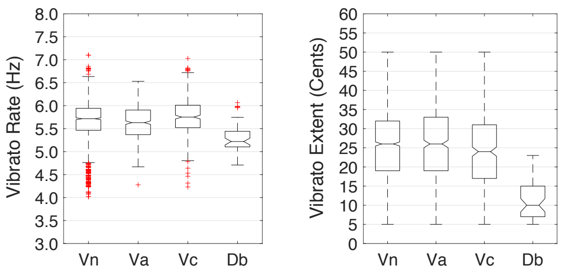


Figure 9. Distributions of vibrato rate and extent for different instruments.

In order to further demonstrate the potential applications of our approach in musicology studies, we analyze how the vibrato rate and extent vary on different instruments and players in this dataset. Figure 9 plots the distributions of rate and extent for the four string instruments, where each sample point represents one track. Similar vibrato rate and extent can be observed for violin and viola whereas, in contrast, we observe a significant drop for the double bass, where a slower rate and subtler extent is inferred. This was explained in [18]; to produce audible pitch fluctuations on the thicker and longer strings on double bass requires more effort to overcome the strength, flexibility, and coordination than other string instruments. Thus vibrato rates of double bass players (4-5 Hz [20]) are typically slower than other string instrumentalists.

We also analyze the vibrato patterns of the four different violinists among the 31 violin tracks, as plotted in Figure 10. Vibrato rate is more dispersed among players than vibrato extent, and both rate and extent show a similar trend among the players. For example, the second player exhibits a slower vibrato rate with a subtler vibrato extent, while the forth player exhibits a faster vibrato rate with a pronounced vibrato extent. This may be because of different players' articulation styles, or different characteristics of the pieces. Detailed discussion is not included in this paper, but our proposed system can provide a powerful tool for further analyses on the musicology side.

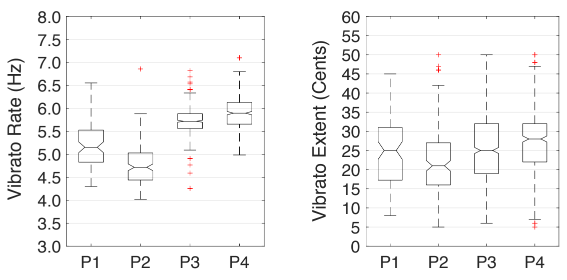


Figure 10. Distributions of vibrato rate and extent of four different violin players.

5. CONCLUSION

We proposed a video-based vibrato detection and analysis framework for polyphonic string music. Specifically, we developed two methods that utilize the motion features from the video for vibrato detection based on the observed correlation between the motion vibrations and the vibrato pitch fluctuations. We also extended the framework to estimate the vibrato rate and extent. Experiments show that the proposed method is successful and offers much better performance than audio-based methods, particularly on pieces with high polyphony, where the strong interference between sources severely degrades the performance of audio-based methods. In future work, it would be helpful to develop a non-score-informed framework for vibrato detection and analysis.

6. ACKNOWLEDGMENT

We thank the Center for Integrated Research Computing (CIRC), University of Rochester for providing computational resources for the project.

7. REFERENCES

[1] Jakob Abeßer, Estefanía Cano, Klaus Frieler, Martin Pfleiderer, and Wolf-Georg Zaddach. Score-informed analysis of intonation and pitch modulation in jazz solos. In *Proc. Intl. Society for Music Information Retrieval (ISMIR)*, pages 823–829, 2015.

[2] Jakob Abeßer, Klaus Frieler, Estefanía Cano, Martin Pfleiderer, and Wolf-Georg Zaddach. Score-informed analysis of tuning, intonation, pitch modulation, and dynamics in jazz solos. *IEEE/ACM Trans. Audio, Speech, and Language Process.*, 25(1):168–177, 2017.

[3] Isabel Barbancho, Cristina de la Bandera, Ana M Barbancho, and Lorenzo J Tardon. Transcription and expressiveness detection system for violin music. In *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Process. (ICASSP)*, pages 189–192. IEEE, 2009.

[4] Jordi Bonada, Robert Lachlan, and Merlijn Blaauw. Bird song synthesis based on hidden markov models. In *Proc. InterSpeech*, volume 2016, 2016.

[5] Karthik Dinesh, Bochen Li, Xinzhao Liu, Zhiyao Duan, and Gaurav Sharma. Visually informed multi-pitch analysis of string ensembles. In *Proc. IEEE Intl. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, 2017.

[6] Jonathan Driedger, Stefan Balke, Sebastian Ewert, and Meinard Müller. Template-based vibrato analysis of music signals. In *Proc. Intl. Society for Music Information Retrieval (ISMIR)*, 2016.

[7] Zhiyao Duan and Bryan Pardo. Soundprism: An online system for score-informed source separation of music audio. *IEEE J. Sel. Topics Signal Process.*, 5(6):1205–1215, 2011.

[8] Harvey Fletcher and Larry C Sanders. Quality of violin vibrato tones. *The Journal of the Acoustical Society of America*, 41(6):1534–1544, 1967.

[9] Anders Friberg, Erwin Schoonderwaldt, and Patrik N Juslin. Cuex: An algorithm for automatic extraction of expressive tone parameters in music performance from acoustic signals. *Acta acustica united with acustica*, 93(3):411–420, 2007.

[10] John M Geringer, Rebecca B MacLeod, and Michael L Allen. Perceived pitch of violin and cello vibrato tones among music majors. *Journal of Research in Music Education*, 57(4):351–363, 2010.

[11] Hung-Yan Gu and Zheng-Fu Lin. Singing-voice synthesis using ann vibrato-parameter models. *J. Inf. Sci. Eng.*, 30(2):425–442, 2014.

[12] Chao-Ling Hsu and Jyh-Shing Roger Jang. Singing pitch extraction by voice vibrato/tremolo estimation and instrument partial deletion. In *Proc. Intl. Society for Music Information Retrieval (ISMIR)*, pages 525–530, 2010.

[13] Hanna Järveläinen. Perception-based control of vibrato parameters in string instrument synthesis. In *Proc. International Computer Music Conference (ICMC)*, 2002.

[14] Bochen Li, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. See and listen: Score-informed association of sound tracks to players in chamber music performance videos. In *Proc. IEEE Intl. Conf. Acoust., Speech, and Signal Process. (ICASSP)*, 2017.

[15] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a classical musical performance dataset for multimodal music analysis: Challenges, insights, and applications. *IEEE Trans. Multimedia*. submitted. Available: `https://arxiv.org/abs/1612.08727`.

[16] Bochen Li, Chenliang Xu, and Zhiyao Duan. Audio-visual source association for string ensembles through multi-modal vibrato analysis. In *Proc. Sound and Music Computing (SMC)*, 2017.

[17] Matthias Mauch and Simon Dixon. PYIN: a fundamental frequency estimator using probabilistic threshold distributions. In *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Process. (ICASSP)*, pages 659–663. IEEE, 2014.

[18] James Paul Mick. *An analysis of double bass vibrato: Rates, widths, and pitches as influenced by pitch height, fingers used, and tempo*. PhD thesis, The Florida State University, 2012.

[19] Tomoyasu Nakano, Masataka Goto, and Yuzuru Hiraga. An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features. In *Proc. InterSpeech*, 2006.

[20] George Papich and Edward Rainbow. A pilot study of performance practices of twentieth-century musicians. *Journal of Research in Music Education*, 22(1):24–34, 1974.

[21] Jeongsoo Park and Kyogu Lee. Harmonic-percussive source separation using harmonicity and sparsity constraints. In *Proc. Intl. Society for Music Information Retrieval (ISMIR)*, pages 148–154, 2015.

[22] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2010.

[23] Johan Sundberg. Acoustic and psychoacoustic aspects of vocal vibrato. *STL-QPSR*, pages 35–62, 1995.

[24] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, School of Computer Science, Carnegie Mellon University, Apr. 1991.

[25] José Ventura, Ricardo Sousa, and Anibal Ferreira. Accurate analysis and visual feedback of vibrato in singing. In *Proc. Intl. Symposium on Communications Control and Signal Process. (ISCCSP)*, pages 1–6. IEEE, 2012.

[26] Henrik Von Coler and Axel Roebel. Vibrato detection using cross correlation between temporal energy and fundamental frequency. In *Proc. 131st Audio Engineering Society Convention*, 2011.

[27] Luwei Yang, Khalid Z Rajab, and Elaine Chew. The filter diagonalisation method for music signal analysis: frame-wise vibrato detection and estimation. *Journal of Mathematics and Music*, pages 1–19, 2017.

[28] Jun Yin, Ye Wang, and David Hsu. Digital violin tutor: an integrated system for beginning violin learners. In *Proc. ACM Intl. Conf. Multimedia*, pages 976–985. ACM, 2005.

DECODING NEURALLY RELEVANT MUSICAL FEATURES USING CANONICAL CORRELATION ANALYSIS

Nick Gang Blair Kaneshiro Jonathan Berger
Center for Computer Research in Music and Acoustics
Stanford University
{ngang,blairbo,brg}@ccrma.stanford.edu

Jacek P. Dmochowski
Department of Biomedical Engineering
City College of New York
jdmochowski@ccny.cuny.edu

ABSTRACT

Music Information Retrieval (MIR) has been dominated by computational approaches. The possibility of leveraging neural systems via brain-computer interfaces is an alternative approach to annotating music. Here we test this idea by measuring correlations between musical features and brain responses in a statistically optimal fashion. Using an extensive dataset of electroencephalographic (EEG) responses to a variety of natural music stimuli, we employed Canonical Correlation Analysis to identify spatial EEG components that track temporal stimulus components. We found multiple statistically significant dimensions of stimulus-response correlation (SRC) for all songs studied. The temporal filters that maximize correlation with the neural response highlight harmonics and subharmonics of that song's beat frequency, with different harmonics emphasized by different components. The most stimulus-driven component of the EEG has an anatomically plausible, symmetric frontocentral topography that is preserved across stimuli. Our results suggest that different neural circuits encode different temporal hierarchies of natural music. Moreover, as techniques for decoding EEG advance, it may be possible to automatically label music via brain-computer interfaces that capture neural responses that are then translated into stimulus annotations.

1. INTRODUCTION

Computationally extracted audio features have been used in Music Information Retrieval (MIR) research to model the perceptual attributes of music. Music-specific features were first introduced by Tzanetakis & Cook for genre classification [43]. These and other features have been used in subsequent work for further study of genre [14, 21] and other music-tagging applications including emotion and mood classification [25, 35, 41] and artist identification [26].

By contrast, the music neuroscience community has historically focused primarily on experimental stimuli consisting of simple tones or short, synthesized instrumental melodies. This controlled paradigm allows for precise experimental manipulations, with the goal of investigating specific musical parameters; it also permits event-related averaging of responses over repeated trials. However, these stimuli lack the complexity and ecological validity of music that is consumed in real life, and preclude the study of global music processing [20].

In recent years, however, this field has increasingly utilized "naturalistic" music stimuli, including complete, real-world musical works. Here, the computationally extracted features developed for MIR research have found direct application as they provide objective, time-varying stimulus representations for which neural correlates can be investigated. To date, this approach has been successfully applied to a variety of brain imaging modalities including functional magnetic resonance imaging (fMRI) [1, 2, 40, 42], electroencephalography (EEG) [6, 22, 30, 38], and electrocorticography (ECoG) [31, 32, 37]. Both encoding (predicting neural activations from stimulus features) and decoding (predicting stimulus features from neural activations) approaches have been explored [2, 27, 40].

While neuroscience is not yet an established subfield of MIR, the approaches and insights of each field are arguably complementary [3, 16]. In the present study, we extend this interdisciplinary approach and investigate the relationship between time-varying features of naturalistic music and their EEG responses. We employ a hybrid encoding-decoding model to derive features and brain signals that maximally covary. The model temporally filters musical features while spatially filtering the EEG to learn a multidimensional mapping between stimulus and response, implemented here by Canonical Correlation Analysis. We uncover multiple statistically significant dimensions of stimulus-response correlation, with the first dimension showing a consistent EEG filter across different songs. Moreover, the temporal filters that maximize SRC emphasize harmonics and subharmonics of the beat frequency, with different harmonics selected by different dimensions of SRC. Our findings suggest that musical features can potentially be annotated by processing neural responses, opening up an entirely novel approach to MIR. Finally, all data and code will be made publicly available.

The remainder of the paper is organized as follows. In Section 2, we describe the EEG dataset, audio stimulus feature extraction, and analysis procedures. We present the results of our analyses in Section 3, and conclude with a discussion in Section 4.

2. METHODS

All analyses were performed using Matlab. [1]

2.1 EEG Dataset

Seeking ready-to-use EEG data reflecting natural music listening and for which we could obtain the stimuli, we used the publicly available NMED-H dataset [18]. This dataset contains EEG responses to intact and scrambled versions of full-length "Bollywood" songs, each approximately 4.5 minutes long. We used the responses to intact songs only, which comprise data from 48 unique participants (12 per song), who each heard their song twice—a total of 24 EEG trials per song. The data frames have been filtered and cleaned of ocular and noise artifacts, and contain recordings from 125 electrodes at a sampling rate of 125 Hz with average reference. Full details of data acquisition and preprocessing are given in Kaneshiro (2016) [15]. As the downloaded data contained missing values, we imputed missing data using a spatial average from neighboring electrodes before proceeding with analysis.

2.2 Stimulus Feature Extraction

The NMED-H documentation provides links to purchase the songs from iTunes, and instructions for converting them to the intact versions of the experimental stimuli [18]. After following those procedures, we extracted acoustical features using the MIR Toolbox, Version 1.5 [19]. We extracted the same collection of 20 short-term features that were used in a recent fMRI study by Alluri et al. [1]: Zero crossing rate, spectral centroid, high/low energy ratio, spectral spread, spectral rolloff, spectral entropy, spectral flatness, roughness, RMS energy, broadband spectral flux, and spectral flux for 10 octave-wide subbands. Features were extracted in 25-msec analysis windows with a 50% overlap between frames (standard parameters for short-term features [1, 43]), yielding a feature sampling frequency of 80 Hz. As in the Alluri study, we also orthogonalized the features using PCA, providing a lower-dimensional stimulus representation that contains contributions of all features under consideration [1]. We performed all subsequent analyses using PC1, as well as two individual features. RMS and spectral flux were chosen as they reflect amplitude envelope and timbre, respectively, and have been used in previous studies mapping music stimulus features to brain responses [1, 2, 30, 42].

As a reference for interpreting results, we extracted beat and tempo information from the stimulus audio files using a publicly available Matlab implementation [8]. [2] From the global tempo estimates, we computed frequencies relevant to processing hierarchical timescales in music, namely those corresponding to the beat (quarter note), as well as one fourth (whole note), half (half note), twice (eighth note), and four times (sixteenth note) the beat frequency. Previous studies have investigated contributions of beat frequencies to stimulus amplitude envelopes [28]; here we have taken a similar approach, visualizing low-frequency magnitude spectra of the three features used for analysis.

The audio waveforms, spectrograms, low-frequency magnitude spectra, and PC1 loadings of the four stimuli are shown in Fig. 1. By visual inspection, it is apparent that the four songs have different structures, and a variety of tempos. Furthermore, the feature FFTs show spectral peaks at both beat-relevant frequencies and other frequencies not occurring at multiples of the beat. Interestingly, PC1 loadings computed across the full set of 20 features are mostly consistent from song to song.

2.3 Canonical Correlation Analysis

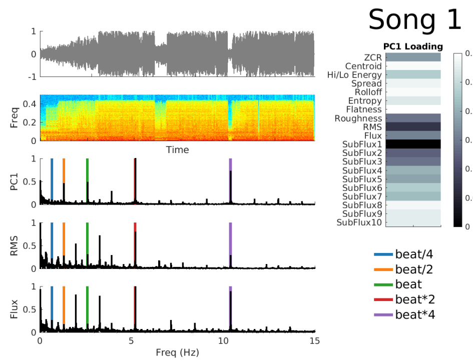
Canonical Correlation Analysis (CCA) involves projecting two data sets onto subspaces such that the projections are maximally correlated across time [9, 10, 13]. It has been used extensively in neuroscience, most recently as a technique for investigating links between visual stimuli and their EEG responses [7]. This approach may be thought of "hybrid encoding-decoding", in that the stimulus is temporally filtered (encoded) and the neural response spatially filtered (decoded), with the filtering optimized by CCA. The result is a multidimensional measure of the stimulus-response correlation (SRC), where each dimension emphasizes a different temporal component of the stimulus and a different spatial component of the EEG.

The inputs to the CCA are two matrices. For the present application, $X \in \mathbb{R}^{L\times T}$ is a convolution matrix of the stimulus feature where the row dimension spans time delays ("lags") and the column dimension spans time. In this construction, temporal filtering of the stimulus feature is achieved by multiplication with X. Matrix $Y \in \mathbb{R}^{D\times T}$ is the EEG data, where the row dimension spans electrodes and the column dimension spans time. CCA on X and Y produces a matrix of temporal filters $H \in \mathbb{R}^{L\times K}$ and a corresponding matrix of spatial filters $W \in \mathbb{R}^{D\times K}$ that extract temporal and spatial components from the stimulus and EEG, respectively, where K is the number of components. Therefore we obtain $U = H^TX$ and $V = W^TY$, where U is a matrix of temporally filtered stimulus components, and V is a matrix of spatially filtered EEG components. The filters H and W are computed to maximize the correlation among corresponding rows of U and V (i.e., the components), under the constraint that the rows of U and V are temporally uncorrelated. The components are sorted in descending order of correlation, such that the first component pair (first rows of U and V) are most strongly correlated.

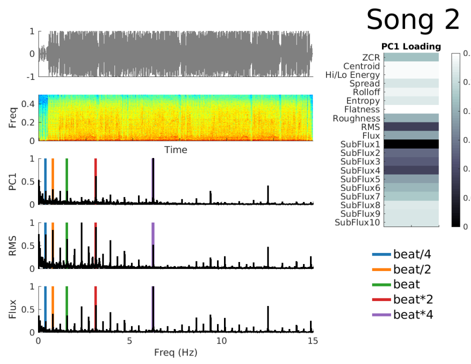
On a per-song basis, we pooled the data across trials to learn the model parameters. As the input sampling rates of the EEG and acoustical feature were not identical, we resampled the EEG to the sampling rate of the acoustical

[1] https://www.mathworks.com/

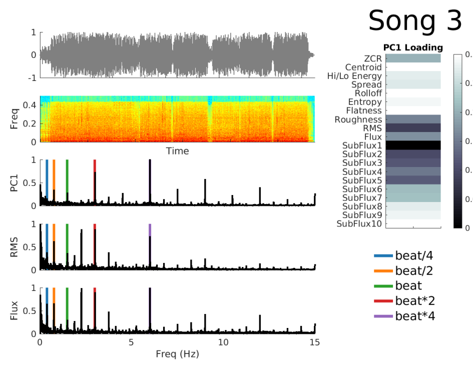
[2] https://labrosa.ee.columbia.edu/projects/beattrack/



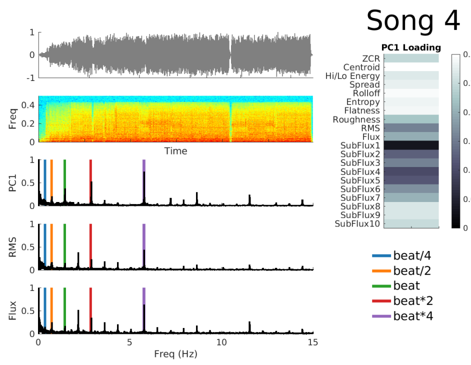
(a) Song 1: “Ainvayi Ainvayi”.



(b) Song 2: “Daaru Desi”.



(c) Song 3: “Haule Haule”.



(d) Song 4: “Malang”.

Figure 1: Features of the songs used here as stimuli. From top left to bottom left in each pane are the waveform, spectrogram, and low frequency spectrum of each individual feature used (PC1, RMS energy, and spectral flux). On the right is the loading vector for the first PC computed across the 20 short-term features for that song.

features (80 Hz) beforehand. We performed separate CCA computations for each acoustical feature for each song, considering samplewise shifts up to 2 seconds in the construction of the feature input matrix X.

In sum, the CCA procedure outputs a delay-by-component stimulus filter matrix H, an electrodes-by-component response filter W, as well as the time samples-by-component filtered data outputs V and U. The Matlab code used to perform these analyses is made publicly available through GitHub. [3]

2.4 Visualizing the CCA Filters

While the columns of W provide the spatial filter weights, a “forward model” is recommended for visualizing component topographies on the scalp [12]. Thus, we used the EEG covariance matrix $R = YY^T$ to compute the forward-model projection $A = RW(W^TRW)^{-1}$ [29]. The columns of A represent the projection of the component onto the scalp and are visualized topographically.

For the temporal filters, we are interested primarily in their spectral characteristics, particularly at musically relevant (beat-related) frequencies. Therefore, we computed the FFT of each temporal filter and plotted its magnitude spectrum.

[3] `http://jd-lab.org/resources/`

2.5 Stimulus-to-Response Correlations

The CCA procedure described above outputs U and V matrices containing the filtered data on a per-song, per-feature basis. We computed SRC for the first 5 components on a per-trial basis across the full duration of the trial. We report the mean correlation coefficient across trials, on a per-component, per-feature, per-song basis.

Due to autocorrelation characteristics of the stimulus and response data [37], we assessed statistical significance using a permutation test approach [39]. This was done by implementing the following procedure for each CCA computation performed above: First, we disrupted the temporal structure of individual trials of input EEG (while preserving aggregate spectral content) by phase scrambling the data from each electrode. Following that, the CCA and SRC computations were performed using the phase-scrambled EEG and intact acoustical feature as inputs. This procedure was repeated 500 times. We compared the SRC from intact data to the distribution of SRC across permutation iterations for computation of p-values. We corrected for multiple comparisons using False Discovery Rate (FDR) [4]. Reported statistical significance ($p < 0.05$) and marginal significance ($0.05 \le p < 0.1$) reflect FDR correction.

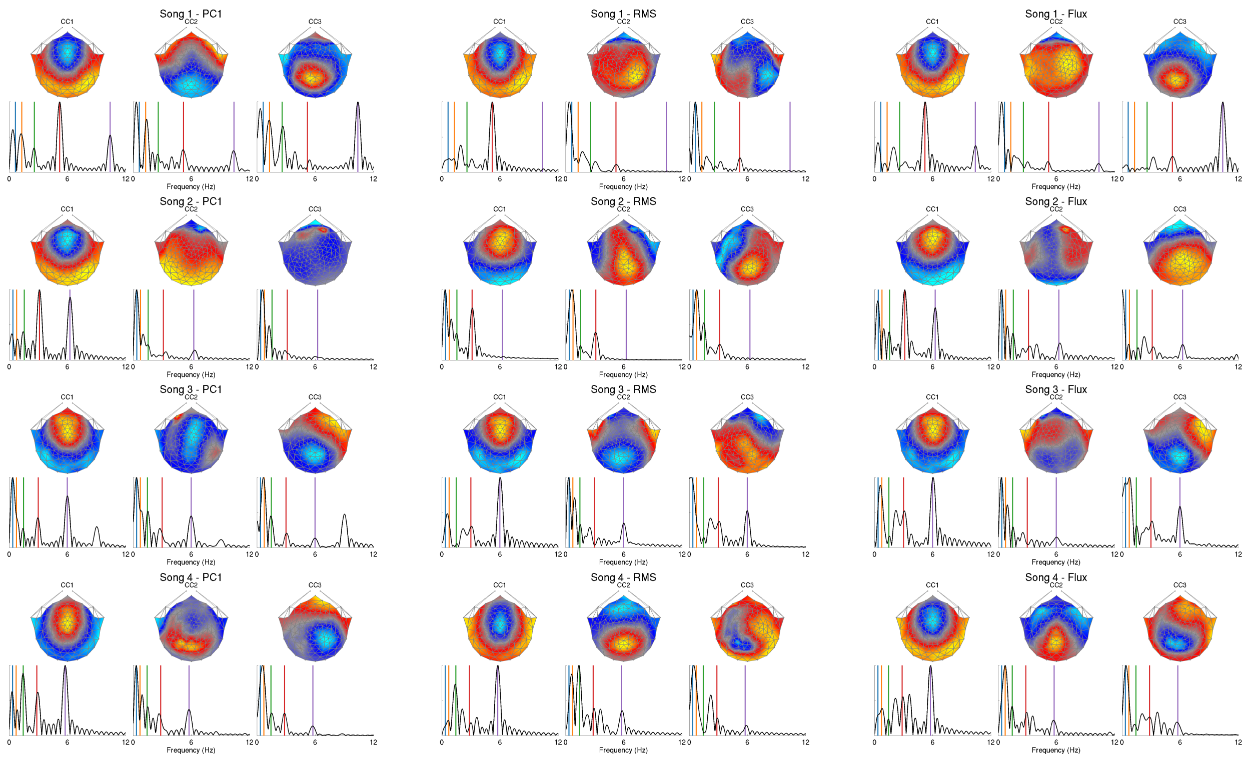


Figure 2: CCA filters. The spatial and temporal filters comprising each CCA pair are visualized for all songs and input features. Shown are the component topographies (spatial filters), as well as the frequency-domain representations of the temporal filters. The first 3 CCs are plotted. In each spectrum, vertical lines denote one fourth the beat frequency (blue), half the beat (orange), beat (green), twice the beat (red), and four times the beat (purple).

3. RESULTS

3.1 Spatial and Temporal Filters

We first probed the spatial topographies of the EEG components that best represented the musical features. Fig. 2 shows that the topography of component 1 is common across songs and features, up to a sign ambiguity inherent to CCA. The symmetric frontocentral topography of CC1 matches various past results involving spatial decomposition of brain responses during natural music listening [17, 33, 38]. Unlike the first CC, the second and third components tend to vary with the stimulus, but possess smooth and broad topographies consistent with the projections of cortical sources onto the scalp.

Interestingly, the temporal filters of each component are focused on harmonics and subharmonics of the song's beat frequency. In the case of CC1, the frequency responses of these filters tend to show peaks at higher beat-related frequencies (eighth and sixteenth notes). Subsequent CCs tend to show peaks at lower beat related frequencies (whole and half notes). Two exceptions to this are the filters for PC1 and spectral flux in Song 1. In both cases CC3 heavily emphasizes the sixteenth note frequency.

While some temporal filters within a single song and feature show similar frequency responses, these components can be differentiated by their phase. For example, Fig. 3 shows the time-domain representation of the temporal filters output for Song 1 RMS and Song 2 RMS. In both cases, the second and third filters emphasize whole note frequencies, but with a different phase. In general, all temporal filters show far more energy in beat-related frequencies than elsewhere.

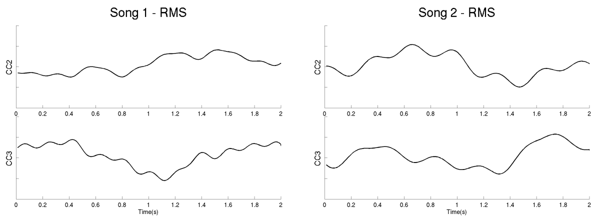


Figure 3: Time-domain examples of temporal filters with similar spectra but different phasing. Left: CC2 and CC3 for Song 1 RMS. Right: CC2 and CC3 for Song 2 RMS.

3.2 Stimulus-to-EEG Correlations

The results of our CCA procedure show multiple dimensions of significant correlation between the stimulus and brain response. As shown in Table 1, CC1 produces statistically significant SRC ($p < 0.05$ after FDR) for all songs and stimulus features. For the remaining components, the coefficients vary in strength across songs and features, but multiple dimensions of significance and marginal significance ($0.05 \leq p < 0.1$) are observed. We note that there is not a universal correspondence between correlation co-

	Comp.	PC1	Flux	RMS
Song 1	CC1	0.0652**	0.0690**	0.0630**
	CC2	0.0280**	0.0242**	0.0275*
	CC3	0.0212**	0.0179**	0.0153
	CC4	0.0177**	0.0179**	0.0123
	CC5	0.0135**	0.0102**	0.0115
Song 2	CC1	0.0522**	0.0573**	0.0524**
	CC2	0.0301**	0.0244**	0.0268**
	CC3	0.0183*	0.0177**	0.0203**
	CC4	0.0158**	0.0109**	0.0137*
	CC5	0.0119**	0.0062	0.0104**
Song 3	CC1	0.0460**	0.0530**	0.0437**
	CC2	0.0225	0.0306**	0.0254**
	CC3	0.0171*	0.0213**	0.0254**
	CC4	0.0139**	0.0119	0.0110
	CC5	0.0099*	0.0084	0.0074
Song 4	CC1	0.0536**	0.0511**	0.0475**
	CC2	0.0248	0.0324**	0.0261*
	CC3	0.0194*	0.0192**	0.0203**
	CC4	0.0170**	0.0178**	0.0135
	CC5	0.0114**	0.0102*	0.0089

Table 1: Multidimensional stimulus-response correlations captured by CCA. '**' denotes statistical significance ($p < 0.05$) and '*' denotes marginal significance ($0.05 \leq p < 0.1$) after correcting for FDR.

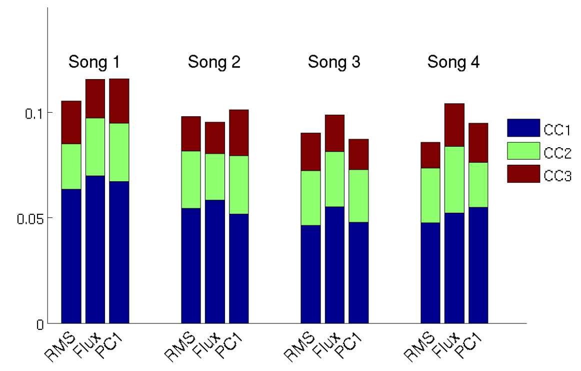


Figure 4: Total stimulus-response correlation for the first 3 CCs of each song and feature. The stacked bar graphs depict the proportion of stimuls-response correlation contributed by each CCA dimension.

efficients and statistical significance. For example, in CC5 of Song 1, the correlation coefficient of $\rho = 0.0102$ for Flux is significant, while the slightly larger $\rho = 0.0115$ for RMS is not. This is due to the fact that separate permutation tests were performed, and surrogate EEG data generated, for each song and audio feature.

4. DISCUSSION

The technique outlined here provides a way to study music processing by direct comparison of an auditory stimulus and its corresponding brain response. Using CCA, matching spatial and temporal filters emerge that maximally correlate the stimulus and response in time. We found multiple dimensions of statistically significant correlation between stimulus and response. While the magnitudes of these correlations are small, the fact that they are not confined to a single dimension suggests that multiple brain areas process distinct portions of the stimulus. Such a multidimensional correlation could not be detected using sensor-space processing.

In past CCA studies using audio-visual stimuli [7], analysis of temporal filter resonances lacked clear relationships to the stimuli. However, the music studied here is organized by a hierarchy of beat- and measure-related periodicities, providing direct references with which to compare the temporal filter frequency responses. Here we found that the temporal filters that extract neurally relevant musical features are focused at harmonics of the beat frequency, independent of the song or feature.

Each CCA dimension emphasizes different brain sources (e.g., spatial topographies) and different combinations of harmonics. These results thus suggest that different temporal hierarchies of music are processed by distinct neural circuits. The topographical consistency of the strongest component, CC1, across all songs suggests a common mechanism for natural music listening. This consistency across songs is especially intriguing since the EEG data reflect disjoint sets of listeners for each song.

From the analysis of SRC significance, PC1 does not appear to outperform other input features. In fact, spectral flux is the only feature that produces at least three significant components in each of the four songs. This may seem counterintuitive, but there is no guarantee that a feature explaining the most variance in the audio data will do the same for brain data. Indeed, the objective of CCA is to maximize covariance between the two data sets.

The correlations between musical features and EEG responses found here, while statistically significant, are fairly low (< 0.1). The low signal-to-noise ratio (SNR) of EEG severely limits the magnitudes of stimulus-response correlations, particularly with linear techniques as were used here. Moreover, the EEG recorded during music listening is driven mostly by sources unrelated to the auditory stimulus. The response to the stimulus comprises only a fractional component of the overall neural activity. Even with more sophisticated imaging modalities such as fMRI, correlation coefficients on the order of 0.1 are typically observed [11]. In order to increase the correlations between stimulus features and brain responses, nonlinear techniques such as deep neural networks could be employed in order to account for higher-order correlations and complex relationships not captured by linear CCA.

In research combining acoustical feature extraction and brain responses, it is important to consider the relative time scales on which stimuli and corresponding brain responses are sampled. Acoustical features are broadly separated into short- and long-term features. Past research has used the short-term features described above, as well as long-term "texture windows" with temporal resolution of around 1 Hz (e.g., 3-sec window with 33% overlap) [43]. Recording modalities for cortical responses can similarly be grouped by their temporal resolution. For example, EEG provides high temporal resolution (typically up to 1000 Hz for cortical responses), while fMRI offers a sampling frequency of only around 0.5 Hz [1]. Thus, in terms of time scales, EEG is amenable to short-term features and fMRI to long-term. Interestingly, however, many studies to date seem mismatched in this regard. There exist both fMRI studies utilizing short-term features [1, 42] and EEG studies utilizing long-term features [6,22,44], meaning significant upsampling or downsampling was needed to compare stimulus features with responses. The present study is the first to our knowledge to non-invasively examine stimulus to response mapping in natural listening using exclusively matching timescales.

When choosing brain response recording modalities for this type of research, it is important to understand the tradeoffs in temporal and spatial resolution. Unlike hemodynamic signals of fMRI, the electrical signals recorded by EEG have been refracted through the skull and scalp; thus, observed topographies represent signals at specific electrodes, but not necessarily activations of specific underlying brain regions. ECoG methods solve this problem by placing electrodes directly on the cortex, but require invasive procedures and generally record from a smaller number of electrodes over a small region of the brain.

The present study correlated time-domain representations of both the acoustical features and EEG responses. The CCA approach could also be applied to transforms of either input. Past EEG and ECoG studies have examined time-frequency representations [6, 22, 31, 32, 37] and compared audio features with oscillatory band power in brain responses. Alternative stimulus input representations can also be considered. Time frequency transforms of the audio such as the Constant-Q Transform or other filterbank decompositions could be used as long- or short-term input features depending on the temporal resolution of interest. Using predetermined and hand-engineered features, as we did here, can also be limiting. The features used here are well represented in past research, but it could be beneficial for a system to learn the audio features themselves with the goal of improving the output of the optimization—for example with deep neural network approaches that have been applied to learn features for music tagging and signal processing systems [5, 34].

Here we have chosen to average SRC coefficients for each song and feature across the full duration of the stimulus, producing a global correlation measure for each set of components. It is also possible to compute a time-varying measure of SRC and further investigate the musical events corresponding to moments of especially high or low SRC. Past research has even linked time-varying SRC to the attentional state of participants [7], pointing to application as a surrogate measure of listener attention. This approach could prove useful in an MIR context, providing a continuous, objective (brain-based) measure of attention to a real-world musical work.

While public access to naturalistic listening data remains limited, additional options exist. Given the limitations of the NMED-H dataset, it would be helpful to test this method on EEG datasets that reflect a wider range of musical genres [36] and tempos [23, 24].

Future research may also consider differing stimuli across participants. Here, each CCA computation operated over concatenated EEG responses to a shared stimulus (e.g., all responses to Song 1). However, CCA has also been used to derive correlated components for unique perceptual experiences such as video game play [7]. MIR applications of this approach could involve pooling responses to different performances of the same song, or allowing participants to choose personal favorites. In addition, it will be interesting to investigate further the composition of the temporal stimulus filters, which for the present analyses are tightly coupled to beat frequencies, when songs of various tempos are analyzed together.

5. ACKNOWLEDGMENTS

This research was supported by the Wallenberg Network Initiative: Culture, Brain, Learning; the Roberta Bowman Denning Fund for Humanities and Technology; and the Patrick Suppes Gift Fund. The authors thank Anthony Norcia for helpful feedback on the analyses presented here.

6. REFERENCES

[1] V. Alluri, P. Toiviainen, I. P. Jääskeläinen, E. Glerean, M. Sams, and E. Brattico. Large-scale brain networks emerge from dynamic processing of musical timbre, key and rhythm. *NeuroImage*, 59(4):3677–3689, 2012.

[2] V. Alluri, P. Toiviainen, T. E. Lund, M. Wallentin, P. Vuust, A. K. Nandi, T. Ristaniemi, and E. Brattico. From Vivaldi to Beatles and back: predicting lateralized brain responses to music. *NeuroImage*, 83:627–636, 2013.

[3] J. J. Aucouturier and E. Bigand. Seven problems that keep MIR from attracting the interest of cognition and neuroscience. *Journal of Intelligent Information Systems*, 41(3):483–497, 2013.

[4] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001.

[5] K. Choi, G. Fazekas, and M. Sandler. Automatic tagging using deep convolutional neural networks. In *ISMIR*, pages 805–811, 2016.

[6] F. Cong, V. Alluri, A. K. Nandi, P. Toiviainen, R. Fa, B. Abu-Jamous, L. Gong, B. G. W. Craenen, H. Poikonen, M. Huotilainen, and T. Ristaniemi. Linking brain responses to naturalistic music through analysis of ongoing EEG and stimulus features. *IEEE Transactions on Multimedia*, 15(5):1060–1069, 2013.

[7] J. P. Dmochowski, J. Ki, P. De Guzman, P. Sajda, and L. C. Parra. Extracting multidimensional stimulus-response correlations using hybrid encoding-decoding of neural activity. *NeuroImage*, 2017.

[8] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[9] H. R. Glahn. Canonical correlation and its relationship to discriminant analysis and multiple regression. *Journal of the Atmospheric Sciences*, 25(1):23–31, 1968.

[10] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, 16(12):2639–2664, 2004.

[11] Uri Hasson, Yuval Nir, Ifat Levy, Galit Fuhrmann, and Rafael Malach. Intersubject synchronization of cortical activity during natural vision. *science*, 303(5664):1634–1640, 2004.

[12] S. Haufe, F. Meinecke, K. Görgen, S. Dähne, J.-D. Haynes, B. Blankertz, and F. Bießmann. On the interpretation of weight vectors of linear models in multivariate neuroimaging. *NeuroImage*, 87:96–110, 2014.

[13] H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

[14] Y. F. Huang, S. M. Lin, H. Y. Wu, and Y. S. Li. Music genre classification based on local feature selection using a self-adaptive harmony search algorithm. *Data & Knowledge Engineering*, 92:60–76, 2014.

[15] B. Kaneshiro. *Toward an Objective Neurophysiological Measure of Musical Engagement*. PhD thesis, Stanford University, 2016.

[16] B. Kaneshiro and J. P. Dmochowski. Neuroimaging methods for music information retrieval: Current findings and future prospects. In *ISMIR*, pages 538–544, 2015.

[17] B. Kaneshiro, J. P. Dmochowski, A. M. Norcia, and J. Berger. Toward an objective measure of listener engagement with natural music using inter-subject EEG correlation. In *ICMPC13*, 2014.

[18] B. Kaneshiro, D. T. Nguyen, J. P. Dmochowski, A. M. Norcia, and J. Berger. Naturalistic music EEG dataset—Hindi (NMED-H). In *Stanford Digital Repository*, 2016.

[19] O. Lartillot and P. Toiviainen. A Matlab toolbox for musical feature extraction from audio. In *DAFx*, pages 237–244, 2007.

[20] D. J. Levitin and V. Menon. Musical structure is processed in language areas of the brain: a possible role for brodmann area 47 in temporal coherence. *NeuroImage*, 20(4):2142–2152, 2003.

[21] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *SIGIR*, pages 282–289, 2003.

[22] Y.-P. Lin, J.-R. Duann, W. Feng, J.-H. Chen, and T.-P. Jung. Revealing spatio-spectral electroencephalographic dynamics of musical mode and tempo perception by independent component analysis. *Journal of Neuroengineering and Rehabilitation*, 11(1):18, 2014.

[23] S. Losorelli, D. T. Nguyen, J. P. Dmochowski, and B. Kaneshiro. Naturalistic music EEG dataset—Tempo (NMED-T). In *Stanford Digital Repository*, 2017.

[24] S. Losorelli, D. T. Nguyen, J. P. Dmochowski, and B. Kaneshiro. NMED-T: A tempo-focused dataset of cortical and behavioral responses to naturalistic music. In *ISMIR*, 2017.

[25] L. Lu, D. Liu, and H. J. Zhang. Automatic mood detection and tracking of music audio signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):5–18, 2006.

[26] M. I. Mandel and D. P. W. Ellis. Song-level features and support vector machines for music classification. In *ISMIR*, pages 594–599, 2005.

[27] T. Naselaris, K. N. Kay, S. Nishimoto, and J. L. Gallant. Encoding and decoding in fMRI. *NeuroImage*, 56(2):400–410, 2011.

[28] S. Nozaradan, I. Peretz, and A. Mouraux. Selective neuronal entrainment to the beat and meter embedded in a musical rhythm. *The Journal of Neuroscience*, 32(49):17572–17581, 2012.

[29] L. C. Parra, C. D. Spence, A. D. Gerson, and P. Sajda. Recipes for the linear analysis of EEG. *NeuroImage*, 28(2):326–341, 2005.

[30] H. Poikonen, V. Alluri, E. Brattico, O. Lartillot, M. Tervaniemi, and M. Huotilainen. Event-related brain responses while listening to entire pieces of music. *Neuroscience*, 312:58–73, 2016.

[31] C. Potes, P. Brunner, A. Gunduz, R. T. Knight, and G. Schalk. Spatial and temporal relationships of electrocorticographic alpha and gamma activity during auditory processing. *NeuroImage*, 97:188–195, 2014.

[32] C. Potes, A. Gunduz, P. Brunner, and G. Schalk. Dynamics of electrocorticographic (ECoG) activity in human temporal and frontal cortical areas during music listening. *NeuroImage*, 61(4):841–848, 2012.

[33] R. S. Schaefer, J. Farquhar, Y. Blokland, M. Sadakata, and P. Desain. Name that tune: Decoding music from the listening brain. *NeuroImage*, 56(2):843–849, 2011.

[34] S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In *ICASSP*, pages 6959–6963, 2014.

[35] Y. Song, S. Dixon, and M. Pearce. Evaluation of musical features for emotion classification. In *ISMIR*, pages 523–528, 2012.

[36] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn. Towards music imagery information retrieval: Introducing the OpenMIIR dataset of EEG recordings from music perception and imagination. In *ISMIR*, 2015.

[37] I. Sturm, B. Blankertz, C. Potes, G. Schalk, and G. Curio. ECoG high gamma activity reveals distinct cortical representations of lyrics passages, harmonic and timbre-related changes in a rock song. *Frontiers in Human Neuroscience*, 8:798, 2014.

[38] I. Sturm, S. Dähne, B. Blankertz, and G. Curio. Multi-variate EEG analysis as a novel tool to examine brain responses to naturalistic music stimuli. *PloS one*, 10(10):e0141281, 2015.

[39] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer. Testing for nonlinearity in time series: the method of surrogate data. *Physica D: Nonlinear Phenomena*, 58(1):77–94, 1992.

[40] P. Toiviainen, V. Alluri, E. Brattico, M. Wallentin, and P. Vuust. Capturing the musical brain with lasso: Dynamic decoding of musical features from fMRI data. *Neuroimage*, 88:170–180, 2014.

[41] K. Trohidis, G. Tsoumakas, G. Kalliris, and Ioannis P. Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, pages 325–330, 2008.

[42] 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*, page nsv060, 2015.

[43] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[44] D. Wang, F. Cong, Q. Zhao, P. Toiviainen, A. K. Nandi, M. Huotilainen, T. Ristaniemi, and A. Cichocki. Exploiting ongoing EEG with multilinear partial least squares during free-listening to music. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

Oral Session 5

Rhythm

TRANSFER LEARNING FOR MUSIC CLASSIFICATION AND REGRESSION TASKS

Keunwoo Choi, György Fazekas, Mark Sandler
Centre for Digital Music
Queen Mary University of London, London, UK
keunwoo.choi@qmul.ac.uk

Kyunghyun Cho
Center for Data Science
New York University, New York, NY, USA
kyunghyun.cho@nyu.edu

ABSTRACT

In this paper, we present a transfer learning approach for music classification and regression tasks. We propose to use a *pre-trained convnet feature*, a concatenated feature vector using the activations of feature maps of multiple layers in a trained convolutional network. We show how this convnet feature can serve as general-purpose music representation. In the experiments, a convnet is trained for music tagging and then transferred to other music-related classification and regression tasks. The convnet feature outperforms the baseline MFCC feature in all the considered tasks and several previous approaches that are aggregating MFCCs as well as low- and high-level music features.

1. INTRODUCTION

In the field of machine learning, transfer learning is often defined as *re-using parameters* that are trained on a *source* task for a *target* task, aiming to transfer knowledge between the domains. A common motivation for transfer learning is the lack of sufficient training data in the target task. When using a neural network, by transferring pre-trained weights, the number of trainable parameters in the target-task model can be significantly reduced, enabling effective learning with a smaller dataset.

A popular example of transfer learning is semantic image segmentation in computer vision, where the network utilises rich information, such as basic shapes or prototypical templates of objects, that were captured when trained for image classification [37]. Another example is pre-trained word embeddings in natural language processing. Word embedding, a vector representation of a word, can be trained on large datasets such as Wikipedia [35] and adopted to other tasks such as sentiment analysis [27].

This work has been part funded by FAST IMPACt EPSRC Grant EP/L019981/1 and the European Commission H2020 research and innovation grant AudioCommons (688382). Mark Sandler acknowledges the support of the Royal Society as a recipient of a Wolfson Research Merit Award. Kyunghyun Cho thanks the support by eBay, TenCent, Facebook, Google and NVIDIA.

There have been several works on transfer learning in Music Information Retrieval (MIR). Hamel et al. proposed to directly learn music features using linear embedding [57] of mel-spectrogram representations and genre/similarity/tag labels [20]. Oord et al. outlines a large-scale transfer learning approach, where a multi-layer perceptron is combined with the spherical K-means algorithm [16] trained on tags and play-count data [54]. After training, the weights are transferred to perform genre classification and auto-tagging with smaller datasets. In music recommendation, Choi et al. used the weights of a convolutional neural network for feature extraction in playlist generation [10], while Liang et al. used a multi-layer perceptron for feature extraction of content-aware collaborative filtering [29].

2. TRANSFER LEARNING FOR MUSIC

In this section, our proposed transfer learning approach is described. A convolutional neural network (convnet) is designed and trained for a source task, and then, the network with trained weights is used as a feature extractor for target tasks. The schematic of the proposed approach is illustrated in Figure 1.

2.1 Convolutional Neural Networks for Music Tagging

We choose music tagging as a source task because *i)* large training data is available and *ii)* its rich label set covers various aspects of music, e.g., *genre*, *mood*, *era*, and *instrumentations*. In the source task, a mel-spectrogram ($\mathbf{X}$), a two-dimensional representation of music signal, is used as the input to the convnet. The mel-spectrogram is selected since it is psychologically relevant and computationally efficient. It provides a mel-scaled frequency representation which is an effective approximation of human auditory perception [36] and typically involves compressing the frequency axis of short-time Fourier transform representation (e.g., 257/513/1025 frequency bins to 64/96/128 Mel-frequency bins). In our study, the number of mel-bins is set to 96 and the magnitude of mel-spectrogram is mapped to decibel scale ($\log_{10}\mathbf{X}$), following [8] since it is also shown to be crucial in [7].

In the proposed system, there are five layers of convolutional and sub-sampling in the convnet as shown in Figure 1. This convnet structure with 2-dimensional 3×3 kernels and 2-dimensional convolution, which is often called *Vggnet* [44], is expected to learn hierarchical time-frequency

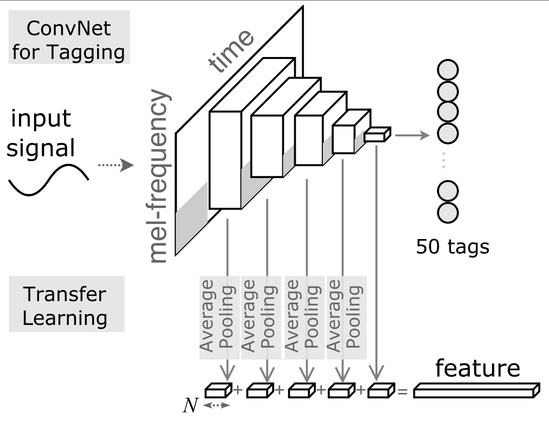


Figure 1: A block diagram of the training and feature extraction procedures. Exponential linear unit (ELU) is used as an activation function in all convolutional layers [15]. Max-pooling of (2, 4), (4, 4), (4, 5), (2, 4), (4, 4) is applied after every convolutional layer respectively. In all the convolutional layers, the kernel sizes are (3, 3), numbers of channels N is 32, and Batch normalisation is used [24]. The input has a single channel, 96-mel bins, and 1360 temporal frames. After training, the feature maps from 1st–4th layers are subsampled using average pooling while the feature map of 5th layer is used as it is, since it is already scalar (size 1×1). Those 32-dimensional features are concatenated to form a *convnet feature*.

patterns. This structure was originally proposed for visual image classification and has been found to be effective and efficient in music classification [1] [11].

2.2 Representation Transfer

In this section, we explain how features are extracted from a pre-trained convolutional network. In the remainder of the paper, this feature is referred to as *pre-trained convnet feature*, or simply *convnet feature*.

It is already well understood how deep convnets learn *hierarchical features* in visual image classification [58]. By convolution operations in the forward path, lower-level features are used to construct higher-level features. Subsampling layers reduce the size of the feature maps while adding local invariance. In a deeper layer, as a result, the features become more invariant to (scaling/location) distortions and more relevant to the target task.

This type of hierarchy also exists when a convnet is trained for a music-related task. Visualisation and sonification of convnet features for music genre classification has shown the different levels of hierarchy in convolutional layers [13], [9].

Such a hierarchy serves as a motivation for the proposed transfer learning. Relying solely on the last hidden layer may not maximally extract the knowledge from a pre-trained network. For example, low-level information such as tempo, pitch, (local) harmony or envelop can be captured in early layers, but may not be preserved in deeper layers due to the constraints that are introduced by the network structure: aggregating local information by discarding less-relevant information in subsampling. For the same reason, deep scattering networks [6] and a convnet for music tagging introduced in [28] use multi-layer representations.

Based on this insight, we propose to use not only the activations of the final hidden layer but also the activations of (up to) *all* intermediate layers to find the most effective representation for each task. The final feature is generated by concatenating these features as demonstrated in Figure 1, where all the five layers are concatenated to serve as an example.

Given five layers, there are $\sum_{n=1}^{5} {}_5C_n = 31$ strategies of layer-wise combination. In our experiment, we perform a nearly exhaustive search and report all results. We designate each strategy by the indices of layers employed. For example, a strategy named '`135`' refers to using a $32 \times 3 = 96$-dimensional feature vector that concatenates the first, third, and fifth layer convnet features.

During the transfer, average-pooling is used for the 1st–4th layers to reduce the size of feature maps to 1×1 as illustrated in Figure 1. Averaging is chosen instead of `max` pooling because it is more suitable for summarising the global statistics of large regions, as done in the last layer in [30]. Max-pooling is often more suitable for capturing the existence of certain patterns, usually in small and local regions [2].

Lastly, there have been works suggesting random-weights (deep) neural networks including deep convnet can work well as a feature extractor [22] [59] (Not identical, but a similar approach is transferring knowledge from an irrelevant domain, e.g., visual image recognition, to music task [19].) We report these results from random convnet features and denote it as *random convnet feature*. Assessing performances of random convnet feature will help to clarify the contributions of the pre-trained knowledge transfer versus the contributions of the convnet structure and nonlinear high-dimensional transformation.

2.3 Classifiers and Regressors of Target Tasks

Variants of support vector machines (SVMs) [45, 50] are used as a classifier and regressor. SVMs work efficiently in target tasks with small training sets, and outperformed K-nearest neighbours in our work for all the tasks in a preliminary experiment. Since there are many works that use hand-written features and SVMs, using SVMs enables us to focus on comparing the performances of features.

3. PREPARATION

3.1 Source Task: Music Tagging

In the source task, 244,224 preview clips of the Million Song Dataset [5] are used (201,680/12,605/25,940 for training/validation/test sets respectively) with top-50 *last.fm* tags including genres, eras, instrumentations, and moods. Mel-spectrograms are extracted from music signals in real-time on the GPU using *Kapre* [12]. Binary cross-entropy is used as the loss function during training.

[1] For more recent information on kernel shapes for music classification, please see [40].

[2] Since the average is affected by zero-padding which is applied to signals that are shorter than 29 seconds, those signals are repeated to create 29-second signals. This only happens in Task 5 and 6 in the experiment.

Task	**Dataset name**	**#clips**	**Metric**	**#classes**
T1. Ballroom dance genre classification	Extended ballroom [32]	4,180	Accuracy	13
T2. Genre classification	Gtzan genre [53]	1,000	Accuracy	10
T3. Speech/music classification	Gtzan speech/music [52]	128	Accuracy	2
T4. Emotion prediction	EmoMusic (45-second) [46]	744	Coefficient of determination (r^2)	N/A (2-dimensional)
T5. Vocal/non-vocal classification	Jamendo [41]	4,086	Accuracy	2
T6. Audio event classification	Urbansound8K [42]	8,732	Accuracy	10

Table 1: The details of the six tasks and datasets used in our transfer learning evaluation.

The ADAM optimisation algorithm [25] is used for accelerating stochastic gradient descent. The convnet achieves 0.849 AUC-ROC score (Area Under Curve - Receiver Operating Characteristic) on the test set. We use the *Keras* [14] and *Theano* [51] frameworks in our implementation.

3.2 Target Tasks

Six datasets are selected to be used in six target tasks. They are summarised in Table 1.

- Task 1: The Extended ballroom dataset consists of specific Ballroom dance sub-genres.
- Task 2: The Gtzan genre dataset has been extremely popular, although some flaws have been found [48].
- Task 3: The dataset size is smaller than the others by an order of magnitude.
- Task 4: Emotion predition on the arousal-valence plane. We evaluate arousal and valence separately. We trim and use the first 29-second from the 45-second signals.
- Task 5. Excerpts are subsegments from tracks with binary labels (*'vocal'* and *'non-vocal'*). Many of them are shorter than 29s. This dataset is provided for benchmarking frame-based vocal detection while we use it as a pre-segmented classification task, which may be easier than the original task.
- Task 6: This is a non-musical task. For example, the classes include *air conditioner*, *car horn*, and *dog bark*. All excerpts are shorter than 4 seconds.

3.3 Baseline Feature and Random Convnet Feature

As a baseline feature, the means and standard deviations of 20 Mel-Frequency Cepstral Coefficients (MFCCs), and their first and second-order derivatives are used. In this paper, this baseline feature is called *MFCCs* or *MFCC vectors*. MFCC is chosen since it has been adopted in many music information retrieval tasks and is known to provide a robust representation. *Librosa* [34] is used for MFCC extraction and audio processing.

The random convnet feature is extracted using the identical convnet structure of the source task and after random weights initialisation with a normal distribution [21] but without a training.

4. EXPERIMENTS

4.1 Configurations

For Tasks 1-4, the experiments are done with 10-fold cross-validation using stratified splits. For Task 5, pre-defined training/validation/test sets are used. The experiment on Task 6 is done with 10-fold cross-validation without replacement to prevent using the sub-segments from the same recordings in training and validation. The SVM parameters are optimised using grid-search based on the validation results. Kernel type/bandwidth of radial basis function and the penalty parameter are selected from the ranges below:

- Kernel type: [*linear*, *radial*]
  - Bandwidth γ in radial basis function : $[1/2^3, 1/2^5, 1/2^7, 1/2^9, 1/2^{11}, 1/2^{13}, 1/N_f]$
- Penalty parameter C : $[0.1, 2.0, 8.0, 32.0]$

A radial basis function is $\exp(-\gamma|x-x'|^2)$, and γ and N_f refer to the radial kernel bandwidth and the dimensionality of feature vector respectively. With larger C, the penalty parameter or regularisation parameter, the loss function gives more penalty to misclassified items and vice versa. We use *Scikit-learn* [38] for these target tasks. The code for the data preparation, experiment, and visualisation are available on GitHub [3] .

4.2 Results and Discussion

Figure 2 shows a summary of the results. The scores of the *i)* best performing convnet feature, *ii)* concatenating '12345' [4] convnet feature and MFCCs, *iii)* MFCC feature, and *iv)* state-of-the-art algorithms for all the tasks.

In all the six tasks, the majority of convnet features outperforms the baseline feature. Concatenating MFCCs

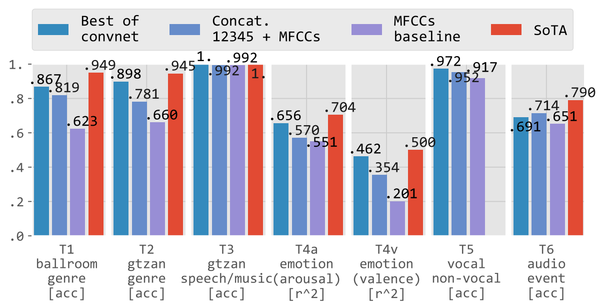


Figure 2: Summary of performances of the convnet feature (blue), MFCCs (purple), and state-of-the-art (red) for Task 1-6 (State-of-the-art of Task 5 does not exist).

[3] https://github.com/keunwoochoi/transfer_learning_music

[4] Again, '12345' refers to the convnet feature that is concatenated from 1st–5th layers. For another example, '135' means concatenating the features from first, third, and fifth layers.

with '12345' convnet feature usually does not show improvement over a pure convnet feature except in Task 6, audio event classification. Although the reported state-of-the art is typically better, almost all methods rely on musical knowledge and hand-crafted features, yet our features perform competitively. An in-depth look at each task is therefore useful to provide insight.

In the following subsections, the details of each task are discussed with more results presented from (almost) exhaustive combinations of convnet features as well as random convnet features at all layers. For example, in Figure 3, the scores of 28 different convnet feature combinations are shown with blue bars. The narrow, grey bars next to the blue bars indicate the scores of random convnet features. The other three bars on the right represent the scores of the concatenation of '12345' + MFCC feature, MFCC feature, and the reported state-of-the-art methods respectively. The rankings within the convnet feature combinations are also shown *in* the bars where top-7 and lower-7 are highlighted.

We only briefly discuss the results of random convnet features here. The best performing random convnet features do not outperform the best-performing convnet features in any task. In most of the combinations, convnet features outperformed the corresponding random convnet features, although there are few exceptions. However, random convnet features also achieved comparable or even better scores than MFCCs, indicating *i)* a significant part of the strength of convnet features comes from the network structure itself, and *ii)* random convnet features can be useful especially if there is not a suitable source task.

4.2.1 Task 1. Ballroom Genre Classification

Figure 3 shows the performances of different features for Ballroom dance classification. The highest score is achieved using the convnet feature '123' with 86.7% of accuracy. The convnet feature shows good performances, even outperforming some previous works that explicitly use rhythmic features.

The result clearly shows that low-level features are crucial in this task. All of the top-7 strategies of convnet feature include the *second* layer, and 6/7 of them include the *first* layer. On the other hand, the lower-7 are ['5', '4', '3', '45', '35', '2', '25'], none of which includes the first layer. Even '1' achieves a reasonable performance (73.8%).

The importance of low-level features is also supported by known properties of this task. The ballroom genre labels are closely related to rhythmic patterns and tempo [32] [49]. However, there is no label directly related to tempo in the source task. Moreover, deep layers in the proposed structure are conjectured to be mostly invariant to tempo. As a result, high-level features from the fourth and fifth layers poorly contribute to the task relative to those from the first, second, and third layers.

The state-of-the-art algorithm which is also the only algorithm that used the same dataset due to its recent release uses *2D scale transform*, an alternative representation of music signals for rhythm-related tasks [33], and

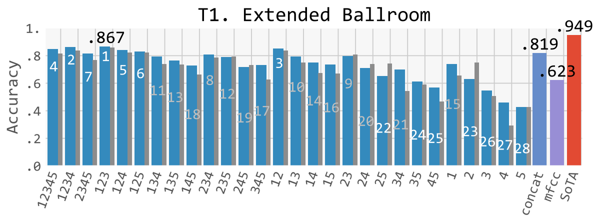


Figure 3: Performances of Task 1 - Ballroom dance genre classification of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method. (Note the exception that the SoTA is reported in weighted average recall.)

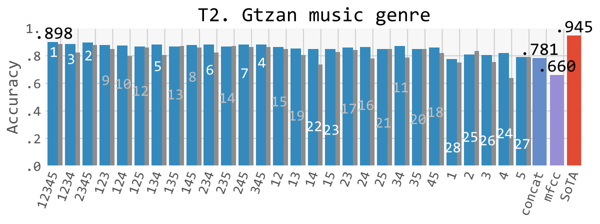


Figure 4: Performances of Task 2 - Gtzan music genre classification of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method.

reports 94.9% of weighted average recall. For additional comparisons, there are several works that use the Ballroom dataset [18]. This has 8 classes and it is smaller in size than the Extended Ballroom dataset (13 classes). Laykartsis and Lerch [31] combines beat histogram and timbre features to achieve 76.7%. Periodicity analysis with SVM classifier in Gkiokas et al. [17] respectively shows 88.9%/85.6 - 90.7%, before and after feature selection.

4.2.2 Task 2. Gtzan Music Genre Classification

Figure 4 shows the performances on Gtzan music genre classification. The convnet feature shows 89.8% while the concatenated feature and MFCCs respectively show only 78.1% and 66.0% of accuracy. Although there are methods that report accuracies higher than 94.5%, we set 94.5% as the state-of-the-art score following the dataset analysis in [48], which shows that the perfect score cannot surpass 94.5% considering the noise in the Gtzan dataset.

Among a significant number of works that use the Gtzan music genre dataset, we describe four methods in more detail. Three of them use an SVM classifier, which enables us to focus on the comparison with our feature. Arabi and Lu [1] is most similar to the proposed convnet features in a way that it combines low-level and high-level features and shows a similar performance. Beniya et al. [4] and Huang et al. [23] report the performances with many low-level features before and after applying feature selection algorithms. Only the latter outperforms the proposed method and only after feature selection.

- Arabi and Lu [1] uses not only low-level features such as {spectral centroid/flatness/roll-off/flux}, but also high-level musical features such as {beat, chord distribution and chord progressions}. The best combination of the features shows 90.79% of accuracy.
- Beniya et al. [4] uses a particularly rich set of statistics such as {mean, standard deviation, skewness, kurtosis,

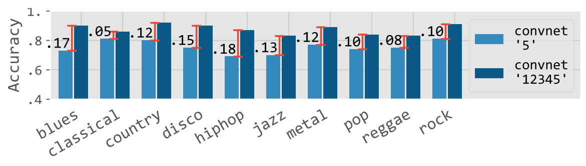


Figure 5: Comparison of per-label results of two convnet feature strategies, '12345' and '5' for Gtzan music genre classification. Numbers denote the differences of scores.

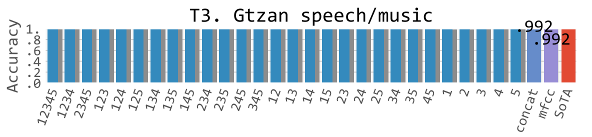


Figure 6: Performances of Task 3 - Speech/music classification of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method. All scores of convnet features and SoTA are 1.0 and omitted in the plot.

covariance} of many low-level features including {RMS energy, attack, tempo, spectral features, zero-crossing, MFCC, dMFCC, ddMFCC, chromagram peak and centroid}. The feature vector dimensionality is reduced by MRMR (max-relevance and min-redundancy) [39] to obtain the highest classification accuracy of 87.9%.

- Huang et al. [23] adopts another feature selection algorithm, self-adaptive harmony search [55]. The method uses statistics such as {mean, standard deviation} of many features including {energy , pitch, and timbral features} and their derivatives. The original 256-dimensional feature achieved 84.3% of accuracy which increases to 92.2% and 97.2% after feature selection.
- Reusing AlexNet [26], a pre-trained convnet for visual image recognition achieved 78% of accuracy [19].

In summary, the convnet feature achieves better performance than many approaches which use extensive music feature sets without feature selection as well as some of the approaches with feature selection. For this task, it turns out that combining features from all layers is the best strategy. In the results, '12345', '2345', and '1234' are three best configurations, and all of the top-7 scores are from those strategies that use more than three layers. On the contrary, all lower-7 scores are from those with only 1 or 2 layers. This is interesting since the majority (7/10) of the target labels already exists in source task labels, by which it is reasonable to assume that the necessary information can be provided only with the last layer for those labels. Even in such a situation, however, low-level features contribute to improving the genre classification performance [5] .

Among the classes of target task, *classical* and *disco*, *reggae* do not exist in the source task classes. Based on this, we consider two hypotheses, *i)* the performances of those three classes may be lower than the others, *ii)* low-level features may play an important role to classify them since high-level feature from the last layer may be biased to the other 7 classes which exist in the source task. However, both hypotheses are rebutted by comparing the performances for each genres with convnet feature '5' and '12345' as in Figure 5. First, with '5' convnet feature, *classical* shows the highest accuracy while both *disco* and *reggae* show accuracies around the average accuracy reported over the classes. Second, aggregating early-layer features affects all the classes rather than the three omitted classes. This suggests that the convnet features are not strongly biased towards the genres that are included in the source task and can be used generally for target tasks with music different from those genres.

[5] On the contrary, in Task 5 - music emotion classification, high-level feature plays a dominant role (see Section 4.2.4).

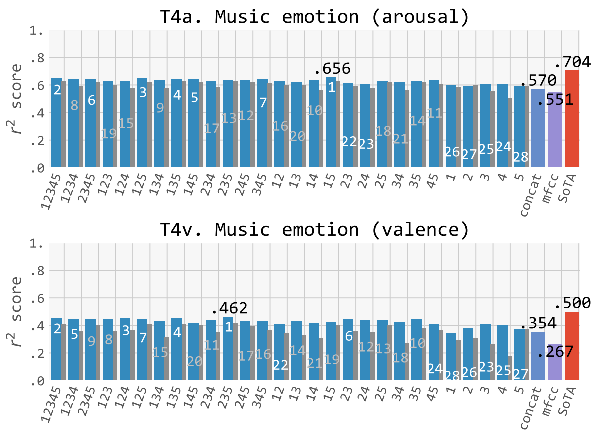


Figure 7: Performances of Task 4a (arousal) and 4v (valence) - Music emotion prediction of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method.

4.2.3 *Task 3. Gtzan Speech/music Classification*

Figure 6 shows the accuracies of convnet features, baseline feature, and state-of-the-art [47] with low-level features including MFCCs and sparse dictionary learning for Gtzan music/speech classification. A majority of the convnet feature combinations achieve 100% accuracy. MFCC features achieve 99.2%, but the error rate is trivial (0.8% is one sample out of 128 excerpts).

Although the source task is only about music tags, the pre-trained feature in any layer easily solved the task, suggesting that the nature of music and speech signals in the dataset is highly distinctive.

4.2.4 *Task 4. Music Emotion Prediction*

Figure 7 shows the results for music emotion prediction (Task 4). The best performing convnet features achieve 0.633 and 0.415 r^2 scores on arousal and valence axes respectively.

On the other hand, the state-of-the-art algorithm reports 0.704 and 0.500 r^2 scores using music features with a recurrent neural network as a classifier [56] that uses 4,777 audio features including many functionals (such as *quantiles, standard deviation, mean, inter peak distances*) of *12 chroma features*, *loudness*, *RMS Energy*, *zero crossing rate*, *14 MFCCs*, *spectral energy*, *spectral roll-off*, etc.

For the prediction of arousal, there is a strong dependency on the last layer feature. All top-7 performances are from the feature vectors that include the fifth layer. The first layer feature also seems important, since all of the top-5 strategies include the first and fifth layer features. For

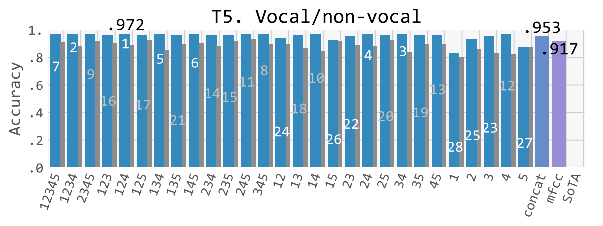


Figure 8: Performances of Task 5 - Vocal detection of convnet features (with random convnet features in grey) and MFCCs.

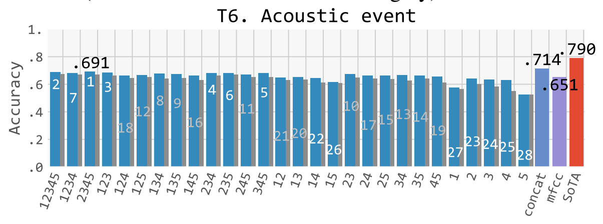


Figure 9: Performances of Task 6 - Acoustic event detection of convnet features (with random convnet features in grey), MFCCs, and the reported state-of-the-art method.

valence prediction, the third layer feature seems to be the most important one. The third layer is included in all of the top-6 strategies. Moreover, '3' strategy was found to be best performing among strategies with single layer feature.

To summarise the results, the predictions of arousal and valence rely on different layers, for which they should be optimised separately. In order to remove the effect of the choice of a classifier and assess solely the effect of features, we compare our approach to the baseline method of [56] which is based on the same 4,777 features with SVM, not a recurrent neural network. The baseline method achieves .541 and .320 r^2 scores respectively on arousal and valence, both of which are lower than those achieved by using the proposed convnet feature. This further confirms the effectiveness of the proposed convnet features.

4.2.5 *Task 5. Vocal/non-vocal Classification*

Figure 8 presents the performances on vocal/non-vocal classification using the Jamendo dataset [41]. There is no known state-of-the-art result, as the dataset is usually used for *frame-based* vocal *detection/segmentation*. Pre-segmented *Excerpt classification* is the task we formulate in this paper. For this dataset, the fourth layer plays the most important role. All the 14 combinations that include the fourth layer outperformed the other 14 strategies without the fourth layer.

4.2.6 *Task 6. Acoustic Event Detection*

Figure 9 shows the results on acoustic event classification using Urbansound8K dataset [42]. Since this is not a music-related task, there are no common tags between the source and target tasks, and therefore the final-layer feature is not expected to be useful for the target task.

The strategy of concatenating '12345' convnet features and MFCCs yields the best performance. Among convnet features, '2345', '12345', '123', and '234' achieve good accuracies. In contrast, those with only one or two layers do not perform well. We were not able to observe any particular dependency on a certain layer.

Since the convnet features are trained on music, they do not outperform a dedicated convnet trained for the target task. The state-of-the-art method is based on a deep convolutional neural network with data augmentation [43]. Without augmenting the training data, the accuracy of convnet in the same work is reported to be 74%, which is still higher than our best result (71.4%). [6]

The convnet feature still shows better results than conventional audio features, demonstrating its versatility even for non-musical tasks. The method in [42] with {*minimum, maximum, median, mean, variance, skewness, kurtosis*} of 25 MFCCs and {*mean and variance*} of the first and second MFCC derivatives (225-dimensional feature) achieved only 68% accuracy using the SVM classifier. This is worse than the performance of the best performing convnet feature.

It is notable again that unlike in the other tasks, concatenating convnet feature and MFCCs results in an improvement over either a convnet feature or MFCCs (71.4%). This suggests that they are complementary to each other in this task.

5. CONCLUSIONS

We proposed a transfer learning approach using deep learning and evaluated it on six music information retrieval and audio-related tasks. The pre-trained convnet was first trained to predict music tags and then aggregated features from the layers were transferred to solve genre classification, vocal/non-vocal classification, emotion prediction, speech/music classification, and acoustic event classification problems. Unlike the common approach in transfer learning, we proposed to use the features from every convolutional layers after applying an average-pooling to reduce their feature map sizes.

In the experiments, the pre-trained convnet feature showed good performance overall. It outperformed the baseline MFCC feature for all the six tasks, a feature that is very popular in music information retrieval tasks because it gives reasonable baseline performance in many tasks. It also outperformed the random-weights convnet features for all the six tasks, demonstrating the improvement by pre-training on a source task. Somewhat surprisingly, the performance of the convnet feature is also very competitive with state-of-the-art methods designed specifically for each task. The most important layer turns out to differ from task to task, but concatenating features from all the layers generally worked well. For all the five *music* tasks, concatenating MFCC feature onto convnet features did not improve the performance, indicating the music information in MFCC feature is already included in the convnet feature. We believe that transfer learning can alleviate the data sparsity problem in MIR and can be used for a large number of different tasks.

[6] Transfer learning targeting audio event classification was recently introduced in [2, 3] and achieved a state-of-the-art performance.

6. REFERENCES

[1] Arash Foroughmand Arabi and Guojun Lu. Enhanced polyphonic music genre classification using high level features. In *Signal and Image Processing Applications (ICSIPA), 2009 IEEE International Conference on*, pages 101–106. IEEE, 2009.

[2] Relja Arandjelović and Andrew Zisserman. Look, listen and learn. *arXiv preprint arXiv:1705.08168*, 2017.

[3] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. Soundnet: Learning sound representations from unlabeled video. In *Advances in Neural Information Processing Systems*, pages 892–900, 2016.

[4] Babu Kaji Baniya, Joonwhoan Lee, and Ze-Nian Li. Audio feature reduction and analysis for automatic music genre classification. In *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 457–462. IEEE, 2014.

[5] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596. University of Miami, 2011.

[6] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[7] Keunwoo Choi, György Fazekas, Kyunghyun Cho, and Mark Sandler. The effects of noisy labels on deep convolutional neural networks for music classification. *arXiv:1706.02361*, 2017.

[8] Keunwoo Choi, György Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. In *The 17th International Society of Music Information Retrieval Conference, New York, USA*. International Society of Music Information Retrieval, 2016.

[9] Keunwoo Choi, György Fazekas, and Mark Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016.

[10] Keunwoo Choi, György Fazekas, and Mark Sandler. Towards playlist generation algorithms using rnns trained on within-track transitions. In *Workshop on Surprise, Opposition, and Obstruction in Adaptive and Personalized Systems (SOAP), Halifax, Canada, 2016*, 2016.

[11] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[12] Keunwoo Choi, Deokjin Joo, and Juho 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.

[13] Keunwoo Choi, Jeonghee Kim, György Fazekas, and Mark Sandler. Auralisation of deep convolutional neural networks: Listening to learned features. In *International Society of Music Information Retrieval (ISMIR), Late-Breaking/Demo Session, Malaga, Spain*. International Society of Music Information Retrieval, 2015.

[14] François Chollet. Keras: Deep learning library for theano and tensorflow. https://github.com/fchollet/keras, 2015.

[15] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[16] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pages 561–580. Springer, 2012.

[17] Aggelos Gkiokas, Vassilis Katsouros, and György Carayannis. Towards multi-purpose spectral rhythm features: An application to dance style, meter and tempo estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):1885–1896, 2016.

[18] Fabien Gouyon, Simon Dixon, Elias Pampalk, and Gerhard Widmer. Evaluating rhythmic descriptors for musical genre classification. In *25th AES International Conference, London, UK*, 2004.

[19] Grzegorz Gwardys and Daniel Grzywczak. Deep image features in music information retrieval. *International Journal of Electronics and Telecommunications*, 60(4):321–326, 2014.

[20] Philippe Hamel, Matthew EP Davies, Kazuyoshi Yoshii, and Masataka Goto. Transfer learning in mir: Sharing learned latent representations for music audio classification and similarity. Curitiba, Brazil, 2013.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[22] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *IEEE International Joint Conference on Neural Networks*, volume 2, pages 985–990. IEEE, 2004.

[23] Yin-Fu Huang, Sheng-Min Lin, Huan-Yu Wu, and Yu-Siou Li. Music genre classification based on local feature selection using a self-adaptive harmony search algorithm. *Data & Knowledge Engineering*, 92:60–76, 2014.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[27] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, volume 14, pages 1188–1196, 2014.

[28] Jongpil Lee and Juhan Nam. Multi-level and multi-scale feature aggregation using pre-trained convolutional neural networks for music auto-tagging. *arXiv preprint arXiv:1703.01793*, 2017.

[29] Dawen Liang, Minshu Zhan, and Daniel PW Ellis. Content-aware collaborative music recommendation using pre-trained neural networks. In *Conference of the International Society for Music Information Retrieval (ISMIR 2015)*, pages 295–301. Malaga, Spain, 2015.

[30] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[31] Athanasios Lykartsis and Alexander Lerch. Beat histogram features for rhythm-based musical genre classification using multiple novelty functions. In *Proceedings of the 16th ISMIR Conference*, pages 434–440, 2015.

[32] Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. *Conference of the International Society for Music Information Retrieval (ISMIR 2016) late-breaking session*, 2016.

[33] Ugo Marchand and Geoffroy Peeters. Scale and shift invariant time/frequency representation using auditory statistics: Application to rhythm description. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*, pages 1–6. IEEE, 2016.

[34] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi YAMAMOTO, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, and Adrian Holovaty. librosa: 0.4.1, October 2015.

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[36] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012.

[37] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[38] 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*, 12:2825–2830, 2011.

[39] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[40] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2017.

[41] Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal detection in music with support vector machines. In *Acoustics, Speech and Signal Processing (ICASSP), 2008 IEEE International Conference on*, pages 1885–1888, March 31 - April 4 2008.

[42] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. In *22st ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014.

[43] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 2017.

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[45] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[46] Mohammad Soleymani, Micheal N Caro, Erik M Schmidt, Cheng-Ya Sha, and Yi-Hsuan Yang. 1000 songs for emotional analysis of music. In *Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia*, pages 1–6. ACM, 2013.

[47] M Srinivas, Debaditya Roy, and C Krishna Mohan. Learning sparse dictionaries for music and speech classification. In *Digital Signal Processing (DSP), 2014 19th International Conference on*, pages 673–675. IEEE, 2014.

[48] Bob L Sturm. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint arXiv:1306.1461*, 2013.

[49] Bob L Sturm et al. Revisiting priorities: Improving mir evaluation practices. In *Proc. 17th International Society for Music Information Retrieval Conference (ISMIR'16), New York, NY, USA*, 2016.

[50] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[51] The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[52] George Tzanetakis. Gtzan musicspeech. *availabe online at http://marsyas.info/download/data sets*, 1999.

[53] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE transactions on*, 10(5):293–302, 2002.

[54] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. Transfer learning by supervised pretraining for audio-based music classification. In *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[55] Chia-Ming Wang and Yin-Fu Huang. Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37(4):2826–2837, 2010.

[56] Felix Weninger, Florian Eyben, and Bjorn Schuller. On-line continuous-time music mood regression with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 5412–5416. IEEE, 2014.

[57] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: Scaling up to large vocabulary image annotation. *International Joint Conference on Artificial Intelligence*, 2011.

[58] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[59] Yujun Zeng, Xin Xu, Yuqiang Fang, and Kun Zhao. Traffic sign recognition using extreme learning classifier with deep convolutional features. In *The 2015 international conference on intelligence science and big data engineering (IScIDE 2015), Suzhou, China*, 2015.

DRUM TRANSCRIPTION VIA JOINT BEAT AND DRUM MODELING USING CONVOLUTIONAL RECURRENT NEURAL NETWORKS

Richard Vogl[1,2] **Matthias Dorfer**[2] **Gerhard Widmer**[2] **Peter Knees**[1]
[1] Institute of Software Technology & Interactive Systems, Vienna University of Technology, Austria
[2] Dept. of Computational Perception, Johannes Kepler University Linz, Austria
{richard.vogl, peter.knees}@tuwien.ac.at

ABSTRACT

Existing systems for automatic transcription of drum tracks from polyphonic music focus on detecting drum instrument onsets but lack consideration of additional meta information like bar boundaries, tempo, and meter. We address this limitation by proposing a system which has the capability to detect drum instrument onsets along with the corresponding beats and downbeats. In this design, the system has the means to utilize information on the rhythmical structure of a song which is closely related to the desired drum transcript. To this end, we introduce and compare different architectures for this task, i.e., recurrent, convolutional, and recurrent-convolutional neural networks. We evaluate our systems on two well-known data sets and an additional new data set containing both drum and beat annotations. We show that convolutional and recurrent-convolutional neural networks perform better than state-of-the-art methods and that learning beats jointly with drums can be beneficial for the task of drum detection.

1. INTRODUCTION

The automatic creation of symbolic transcripts from music in audio files is an important high-level task in music information retrieval. Automatic music transcription systems (AMT) aim at solving this task and have been proposed in the past (cf. [1]), but there is yet no general solution to this problem. The transcription of the drum instruments from an audio file of a song is a sub-task of automatic music transcription, called automatic drum transcription (ADT). Usually, such ADT systems focus solely on the detection of drum instrument note onsets. While this is the necessary first step, for a full transcript of the drum track more information is required. Sheet music for drums—equally to sheet music for other instruments—contains additional information required by a musician to perform a piece. This information comprises (but is not limited to): meter, overall tempo, indicators for bar boundaries, indications for local changes in tempo, dynamics, and playing style of the piece. To obtain some of this information, beat and downbeat detection methods can be utilized. While beats provide tempo information, downbeats add bar boundaries, and the combination of both provides indication for the meter within the bars.

In this work, neural networks for joint beat and drum detection are trained in a multi-task learning fashion. While it is possible to extract drums and beats separately using existing work and combine the results afterwards, we show that it is beneficial to train for both tasks together, allowing a joint model to leverage commonalities of the two problems. Additionally, recurrent (RNN), convolutional (CNN) and convolutional-recurrent neural network (CRNN) models for drum transcription and joint beat and drum detection are evaluated on two well-known, as well as a new data set.

The remainder of this work is structured as follows. In the next section, we discuss related work. In sec. 3, we describe the implemented drum transcription pipeline used to evaluate the network architectures, followed by a section discussing the different network architectures (sec. 4). In sec. 5, we explain the experimental setup to evaluate the joint learning approach. After that, a discussion of the results follows in sec. 6 before we draw conclusions in sec. 7.

2. RELATED WORK

While in the past many different approaches for ADT have been proposed [11,13,15,16,22,24,25,34,38], recent work focuses on end-to-end approaches calculating activation functions for each drum instrument. These methods utilize non-negative matrix factorization (NMF, e.g. adaptive-NMF in Dittmar et al. [7] and partially fixed NMF in Wu et al. [37]) as well as RNNs (RNNs with label time-shift in Vogl et al. [35, 36] and bidirectional RNNs in Southall et al. [31]) to extract the activation functions from spectrograms of the audio signal. Such activation-function-based end-to-end ADT systems circumvent certain issues associated with other architectures. Methods which first segment the song (e.g. using onset detection) and subsequently classify these segments [22, 23, 38] suffer from a loss of information after the segmentation step—i.e. whenever the system fails to detect a segment, this information is lost. Such systems heavily depend on the accuracies of the single components, and can never perform better than the weakest component in the pipeline. Additionally, information of the input signal which is discarded after a processing step might still be of value for later steps.

Since RNNs, especially long short-term memory (LSTM) [17] and gated recurrent unit (GRU) [5] networks, are designed to model long term relationships, one might suspect that systems based on RNNs [31,35,36] can leverage the repetitive structure of the drum tracks and make use of this information. Contrary to this intuition this is not the case for RNN-based systems proposed so far. Both the works of Vogl et al. [35, 36] and Southall et al. [31] use snippets with length of only about one second to train the RNNs. This prohibits learning long-term structures of drum rhythms which are typically in the magnitude of two or more seconds. In [35], it has been shown that RNNs with time-shift perform equally well as bidirectional RNNs, and that backward directional RNNs perform better than forward directional RNNs. Combining these findings indicates that the learned models actually mostly consider local features. Therefore, RNNs trained in such a manner seem to learn only an acoustic, but not a structural model for drum transcription.

Many works on joint beat and downbeat tracking have been published in recent years [2,9,10,19–21,26]. A discussion of all the different techniques would go beyond the scope of this work. One of the most successful methods by Böck et al. [2] is a joint beat and downbeat tracking system using bidirectional LSTM networks. This approach achieves top results in the 2016 MIREX task for beat detection and can be considered the current state of the art. [1]

In this work, a multi-task learning strategy is used to address the discussed issues of current drum transcription systems, cf. [4]. The use of a model jointly trained on drum and beat annotations, combined with longer training snippets, allows the model to learn long-term relations of the drum patterns in combination with beats and downbeats. Furthermore, learning multiple related tasks simultaneously at once can improve results for the single tasks. To this end, different architectures of RNNs, CNNs, and a combination of both, convolutional-recurrent neural networks (CRNNs) [8,27,39], are evaluated.

The rationale behind selecting these three methods for comparison is as follows. RNNs have proven to be well-suited for both drum and beat detection, as well as learning long-term dependencies for music language models [30]. CNNs are among the best performing methods for many image processing and other machine learning tasks, and have been used on spectrograms of music signals in the past. For instance, Schlüter and Böck [28] use CNNs to improve onset detection results, while Gajhede et al. [12] use CNNs to successfully classify samples of three drum sound classes on a non-public data set. CRNNs should result in a model, in which the convolutional layers focus on acoustic modeling of the events, while the recurrent layers learn temporal structures of the features.

3. DRUM TRANSCRIPTION PIPELINE

The implemented method is an ADT system using a similar pipeline as presented in [31] and [36]. Fig. 1 visualizes the overall structure of the system. The next subsections discuss the single blocks of the system in more detail.

[1] `http://www.music-ir.org/mirex/wiki/2016:MIREX2016_Results`

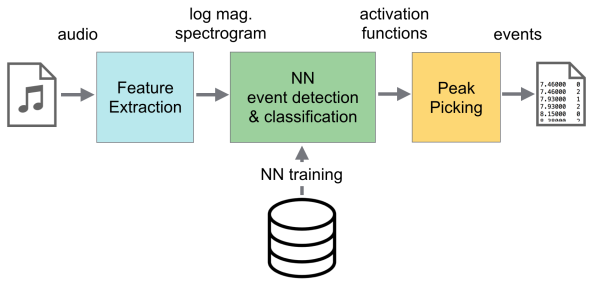


Figure 1. System overview of the implemented drum transcription pipeline used to evaluate the different neural network architectures.

3.1 Feature Extraction

First, a logarithmic magnitude spectrogram is calculated using a 2048-samples window size and a resulting frame rate of 100Hz from a 44.1kHz 16bit mono audio signal input. Then, the frequency bins are transformed to a logarithmic scale using triangular filters (twelve per octave) in a frequency range from 20 to 20,000 Hz. Finally, the positive first-order-differential over time of this spectrogram is calculated and concatenated. This results in feature vectors with a length of 168 values (2x84 frequency bins).

3.2 Activation Function Calculation

The central block in fig. 1 represents the activation function calculation step. This task is performed using a neural network (NN) trained on appropriate training data (see sec. 4). As in most of the related work, we only consider three drum instruments: bass- or kick drum, snare drum, and hi-hat.

While the architectures of the single NNs are different, they share certain commonalities: *i.* all NNs are trained using the same input features; *ii.* the RNN architectures are implemented as bidirectional RNNs (BRNN) [29]; *iii.* the output layers consist of three or five sigmoid units, representing three drum instruments under observation (drum only) or three drum instruments plus beat and downbeat (drum and beats), respectively; and *iv.* the NNs are all trained using the RMSprop optimization algorithm proposed by Tieleman et al. [33], using mini-batches of size eight. For training, we follow a three-fold cross validation strategy on all data sets. Two splits are used for training, 15% of the training data is separated and used for validation after each epoch, while testing/evaluation is done on the third split. The NNs are trained using a fixed learning rate with additional refinement if no improvement on the validation set is achieved for 10 epochs. During refinement the learning rate is reduced and training continues using the parameters of the best performing model so far.

More details on the individual NN architectures are provided in sec. 4.

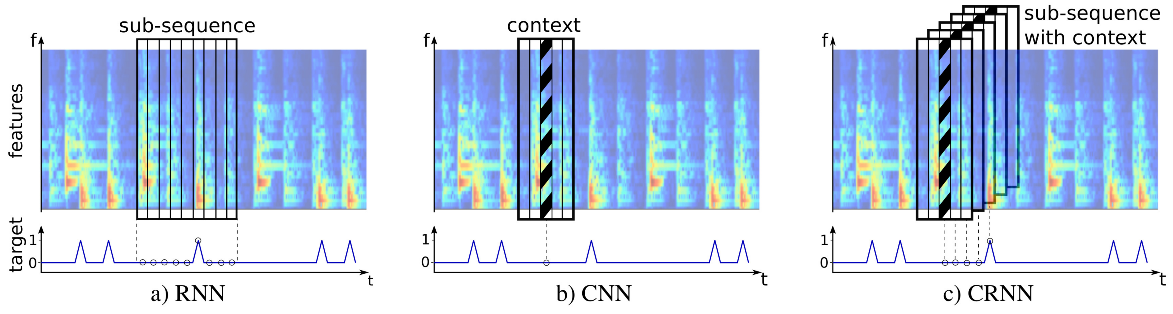


Figure 2. Comparison of mode of operation of RNNs, CNNs, and CRNNs on spectrograms of audio signals. RNNs process the input in a sequential manner. Usually, during training, only sub-sequences of the input signal are used to reduce the memory footprint of the networks. CNNs process the signal frame by frame without being aware of sequences. Because of this, a certain spectral context is added for each input frame. CRNNs, like RNNs, process the input sequentially, but additionally, a spectral context is added to every frame on which convolution is performed by the convolutional layers.

3.3 Preparation of Target Functions

For training the NNs, target functions of the desired output are required besides the input features. These target functions are generated by setting frames of a signal with the same frame rate as the input features to 1 whenever an annotation is present and to 0 otherwise. A separate target function is created for each drum instrument as well as for beats and downbeats.

3.4 Peak Picking

In the last step of our pipeline (rightmost block of fig. 1), the drum instrument onsets (and beats if applicable) are identified using a simple peak picking method introduced for onset detection in [3]: A point n in the activation function $f_a(n)$ is considered a peak if these terms are fulfilled:

1. $f_a(n) = max(f_a(n-m), \cdots, f_a(n))$,
2. $f_a(n) \geq mean(f_a(n-a), \cdots, f_a(n)) + \delta$,
3. $n - n_{lp} > w$,

where δ is a variable threshold. A peak must be the maximum value within a window of size $m+1$, and exceeding the mean value plus a threshold within a window of size $a+1$. Additionally, a peak must have at least a distance of $w+1$ to the last detected peak (n_{lp}). Values for the parameters were tuned on a development data set to be: $m = a = w = 2$.

The threshold for peak picking is determined on the validation set. Since the activation functions produced by the NN contain little noise and are quite spiky, rather low thresholds ($0.1 - 0.2$) give best results.

4. NEURAL NETWORK MODELS

In this section, we explore the properties of the neural network models considered more closely. Of the NN categories mentioned before, we investigate three different types: bidirectional recurrent networks (BRNN), convolutional networks (CNN), and convolutional bidirectional recurrent networks (CBRNN). For every class of networks, two different architectures are implemented: *i.* a smaller network, with less capacity, trained on shorter subsequences (with focus only on acoustic modeling), and *ii.* a larger network, trained on longer subsequences (with additional focus on pattern modeling).

Even though we previously showed that RNNs with label time-shift achieve similar performance as BRNNs [35, 36], in this work, we will not use time-shift for target labels. This is due to three reasons: *i.* the focus of this work is not real-time transcription but a comparison of NN architectures and training paradigms, therefore using a bidirectional architecture has no downsides; *ii.* it is unclear how label time-shift would affect CNNs; *iii.* in [2], the effectiveness of BRNNs (BLSTMs) for beat and downbeat tracking is shown. Thus, in the context of this work, using BRNNs facilitates combining state-of-the-art drum and beat detection methods while allowing us to compare CNNs and RNNs in a fair manner.

4.1 Bidirectional Recurrent Neural Network

Gated recurrent units (GRUs [5]) are similar to LSTMs in the sense that both are gated RNN-cell types that facilitate learning of long-term relations in the data. While LSTMs feature forget, input, and output gates, GRUs only exhibit two gates: update and output. This makes the GRU less complex in terms of number of parameters. It has been shown that both are equally powerful [6], with the difference that more GRUs are needed in an NN layer to achieve the same model capacity as with LSTMs, resulting in more or less equal number of total parameters. An advantage of using GRUs is that hyperparameter optimization for training is usually easier compared to LSTMs.

In this work, two bidirectional GRU (BGRU) architectures are used. The small model (BGRU-a) features two layers of 50 nodes each, and is trained on sequences of 100 frames; the larger model (BGRU-b) consists of three layers of 30 nodes each, and is trained on sequences of 400 frames. For training an initial learning rate of 0.007 is used.

	Frames	Context	Conv. Layers	Rec. Layers	Dense Layers
BGRU-a	100	—	—	2 x 50 GRU	—
BGRU-b	400	—	—	3 x 30 GRU	—
CNN-a	—	9	1xA + 1xB	—	2 x 256
CNN-b	—	25	1xA + 1xB	—	2 x 256
CBGRU-a	100	9	1xA + 1xB	2 x 50 GRU	—
CBGRU-b	400	13	1xA + 1xB	3 x 60 GRU	—

Table 1. Overview of used neural network model architectures and parameters. Every network additionally contains a dense sigmoid output layer. Conv. block A consists of 2 layers with 32 3x3 filters and 3x3 max-pooling; conv. block B consists of 2 layers with 64 3x3 filters and 3x3 max-pooling; both use batch normalization.

4.2 Convolutional Neural Network

Convolutional neural networks have been successfully applied not only in image processing, but also many other machine learning tasks. The convolutional layers are constructed using two different building blocks: block *A* consists of two layers with 32 3x3 filters and block *B* consists of two layers with 64 3x3 filters; both in combination with batch normalization [18], and each followed by a 3x3 max pooling layer and a drop-out layer ($\lambda = 0.3$) [32].

For both CNN models, block *A* is used as input, followed by block *B*, and two fully connected layers of size 256. The only difference between the small (CNN-a) and the large (CNN-b) model is the context used to classify a frame: 9 and 25 frames are used for CNN-a and CNN-b respectively. While plain CNNs do not feature any memory, the spectral context allows the CNN to access surrounding information during training and classification. However, a context of 25 frames (250ms) is not enough to find repetitive structures in the rhythm patterns. Therefore, the CNN can only rely on acoustic, i.e., spectral features of the signal. Nevertheless, with advanced training methods like batch normalization, as well as the advantage that CNNs can easily learn pitch invariant kernels, CNNs are well-equipped to learn a task adequate acoustic model. For training an initial learning rate of 0.001 is used.

4.3 Convolutional Bidirectional RNN

Convolutional recurrent neural networks (CRNN) represent a combination of CNNs and RNNs. They feature convolutional layers as well as recurrent layers. Different implementations are possible. In this work, the convolutional layers directly process the input features, i.e. spectrogram representations, meant to learn an acoustic model (cf. 2D image processing tasks). The recurrent layers are placed after the convolutional layers and are supposed to serve as a means for the network to learn structural patterns.

For this class of NN, the two versions differ in the following aspects: CBGRU-a features 2 recurrent layers with 30 GRUs each, uses a spectral context of 9 frames for convolution, and is trained on sequences of length 100; while CBGRU-b features 3 recurrent layers with 60 GRUs each, uses a spectral context of 13 frames, and is trained on sequences of length 400. For training an initial learning rate of 0.0005 is used.

Table 1 recaps the information of the previous sections in a more compact form. Figure 2 visualizes the modes of operation of the different NN architectures on the input spectrograms.

5. EVALUATION

For evaluation of the introduced NN architectures, the different models are individually trained on single data sets in a three-fold cross-validation manner. For data sets which comprise beat annotations, three different experiments are performed (explained in more detail in section 5.2); using data sets only providing drum annotations, just the drum detection task is performed.

5.1 Data Sets

In this work, the different methods are evaluated using three different data sets, consisting of two well-known and a newly introduced set.

5.1.1 IDMT-SMT-Drums v.1 (SMT)

Published along with [7], the IDMT-SMT-Drums [2] data set comprises tracks containing three different drum-set types. These are: *i.* real-world, acoustic drum sets (titled *RealDrum*), *ii.* drum synthesizers (*TechnoDrum*), and *iii.* drum sample libraries (*WaveDrum*). It consists of 95 simple drum tracks containing bass drum, snare drum and hi-hat only. The tracks have an average length of 15s and a total length of 24m. Also included are additional 285 shorter, single-instrument training tracks as well as 180 single instrument tracks for 60 of the 95 mixture tracks (from the *WaveDrum02* subset)—intended to be used for source separation experiments. These additional single instrument tracks are used as additional training samples (together with their corresponding split) but not for evaluation.

5.1.2 ENST Drums (ENST)

The ENST-Drums set [14] contains real drum recordings of three different drummers performing on different drum kits. [3] Audio files for separate solo instrument tracks

[2] `https://www.idmt.fraunhofer.de/en/business_units/m2d/smt/drums.html`

[3] `http://perso.telecom-paristech.fr/~grichard/ENST-drums/`

	Input Features		Target Functions	
	Spectrogram	Beats	Drums	Beats
DT	✓		✓	
BF	✓	✓	✓	
MT	✓		✓	✓

Table 2. Overview of experimental setup. Rows represent individual tasks and show their input feature and target function combinations.

as well as for two mixtures are included. Additionally, accompaniment tracks are available for a subset of the recordings—the so called minus-one tracks. In this work, the wet mixes (contains standard post-processing like compression and equalizing) of the minus-one tracks were used. They make up 64 tracks of 61s average length and a total length of 1h.

Evaluation was performed on the drum-only tracks (ENST solo) as well as the mixes with their accompaniment tracks (ENST acc.). Since the ENST-Drums data set contains more than the three instruments under observation, only the snare, bass, and hi-hat annotations were used.

5.1.3 RBMA Various Assets 2013 (RBMA13)

This new data set consists of the 30 tracks of the freely available 2013 Red Bull Music Academy Various Assets sampler.[4] The sampler covers a variety of electronically produced music, which encompasses electronic dance music (EDM) but also singer-songwriter tracks and even fusion-jazz styled music. Three tracks on the sampler do not contain any drums and are therefore ignored. Annotations for drums, beats, and downbeats were manually created. Tracks in this set have an average length of 3m 50s. The total length of the data set is 1h 43m.

This data set is different from the other two data sets in three aspects: *i.* it contains quite diverse drum sounds, *ii.* the drum patterns are arranged in the usual song-structure within a full length track, and *iii.* most of the tracks contain singing voice, which showed to be a challenge for systems solely trained on music without singing voice. The annotations for drums and beats have been manually created and are publicly available for download.[5]

5.2 Experimental Setup

To compare the different NN architectures, and evaluate them in the context of ADT using joint learning of beat and drum activations, the following experiments were performed.

5.2.1 Drum Detection (DT)

In this set of experiments, the features as explained in sec. 3.1 and target functions generated from the drum annotations described in sec. 3.3 are used for NN training.

[4] https://rbma.bandcamp.com/album/various-assets-not-for-sale-red-bull-music-academy-new-york-2013

[5] http://ifs.tuwien.ac.at/~vogl/datasets/

	SMT	ENST		RBMA13		
		solo	acc.	DT	BF	MT
GRUts [36]	*92.5*	*83.3*	*75.0*	-	-	-
BGRU-a	93.0	80.9	70.1	59.8	63.6	64.6
BGRU-b	93.3	82.9	72.3	61.8	64.5	64.3
CNN-a	87.6	78.6	70.8	66.2	66.7	63.3
CNN-b	93.4	**85.0**	78.3	66.8	65.2	64.8
CBGRU-a	**95.2**	84.6	76.4	65.2	66.1	66.9
CBGRU-b	93.8	83.9	**78.4**	**67.3**	**68.4**	**67.2**

Table 3. F-measure results for the evaluated models on different data sets. The columns DT, BF, and MT show results for models trained only for drum detection, trained using oracle beats as additional input features, and simultaneously trained on drums and beats, respectively. Bold values represent the best performance for an experiment across models. The baseline can be found in the first row.

These experiments are comparable to the ones in the related work, since we use a similar setup. As baseline, the results in [36] are used. The results of this set of experiments allow to compare the performance of different NN architectures for drum detection.

5.2.2 Drum Detection with Oracle Beat Features (BF)

For this set of experiments, in addition to the input features explained in sec. 3.1, the annotated beats, represented as the target functions for beats and downbeats, are included as input features. As targets for NN training only the drum target functions are utilized. Since beat annotations are required for this experiment, only data sets comprising beat annotations can be used. Using the results of these experiments, it can be investigated if the prior knowledge of beat and downbeat positions is beneficial for drum detection.

5.2.3 Joint Drum and Beat Detection (MT)

This set of experiments represents the multi-task learning investigation. As input for training, again, only the spectrogram features are used. Targets for training of the NNs comprise, in this case, drum and beat activation functions. As discussed in the introduction, in some cases it can be beneficial to train related properties simultaneously. Beats and drums are closely related, because usually drum pattern are repetitive on a bar-level (separated by downbeats) and drum onsets often correlate with beats.

The insight which can be drawn from these experiments is whether simultaneous training of drums, beats, and downbeats is beneficial. It is of interest if the resulting performance is higher than the one achieved for DT; and also if it is below, comparable, or even surpasses the results in the BF experiment series.

Table 2 gives an overview of the properties of the experiments and the used feature/target combination.

5.3 Evaluation Method

To evaluate the performance of the different architectures and training methods, the well-known metrics precision,

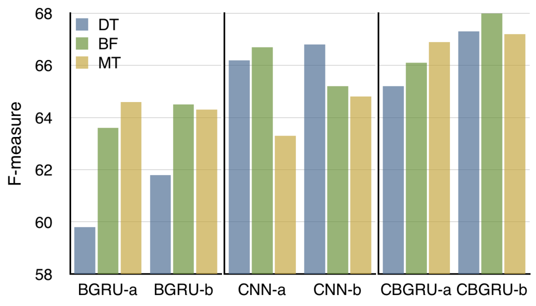


Figure 3. Results for RBMA13 data set, highlighting the influence of oracle beat features (BF) and multi-task learning (MT). While recurrent models (left and right) benefit, convolutional models (center) do not.

recall, and F-measure are used. These are calculated for drum instrument onsets as well as beat positions. True positive, false positive, and false negative onset and beat positions are identified by using a $20ms$ tolerance window. This is in line with the evaluation in [36] which is used as baseline for the experiments of this work. Note that other work, e.g. [7, 25, 31], uses less strict tolerance windows of $30ms$ or $50ms$ for evaluation.

6. RESULTS AND DISCUSSION

Table 3 shows the F-measure results for the individual NN architectures on the data sets used for evaluation. The results for BGRU-a and BGRU-b on the ENST data set are lower than for the baseline, although the models should be comparable. This is due to the fact that in [36] data augmentation is applied. This is especially helpful in the case of the ENST data set, since e.g. the pitches of the base drums vary greatly over the different drum kits. The results for CNN-a are lower than the state of the art, which implies that the context of 9 frames is too small to detect drum events using a CNN. All other results on the ENST and SMT data sets represent an improvement over the state of the art. This shows that CNN with a large enough spectral context (25 frames in this work) can detect drum events better than RNNs. A part of the large increase for the ENST data set can be attributed to the fact that CNNs can model pitch invariance easier than RNNs.

The results for the MT experiments show the following tendencies: For the BGRU-a and BGRU-b models, an improvement can be observed when applying multi-task learning. Compared to using oracle beats (BF) for training, the improvement is higher for BGRU-a and similar in the case of BGRU-b. This result is interesting for two reasons: *i.* although BGRU-a is trained on short sequences, an improvement can be observed, and *ii.* the improvement is comparable to that when using oracle beats (BF) although the beat tracking results are low. This could imply that multi-task learning is also beneficial for the acoustic model of the system. As expected, the CNNs (CNN-a, CNN-b) can not improve when using multi-task learning, but rather the results deteriorate. In case of the convolutional-recurrent models, the result for CBGRU-a is similar to BGRU-a. In case of CBGRU-b no improvement of drum detection performance using multi-task learning can be observed, although it is the case using oracle beats (BF). We attribute this to the fact that CBGRU-b has enough capacity for good acoustic modeling, while the low beat detection results limit the effects of multi-task learning on this level.

BLSTM [2]	*85.6*
BGRU-a	46.4
BGRU-b	46.2
CNN-a	44.9
CNN-b	46.9
CBGRU-a	47.6
CBGRU-b	48.8

Table 4. F-measure results for beat detection for the multi-task learning experiments compared to a state-of-the-art approach (first row) on the RBMA13 set.

Table 4 shows the F-measure results for beat and downbeat tracking. The results are all below the state-of-the-art beat tracker used as baseline [2]. This is due to several factors. In [2], *i.* much larger training sets for beat and downbeat tracking are used, *ii.* the LSTMs are trained on full sequences of the input data, giving the model more context, and *iii.* an additional music language model in the form of a dynamic Bayesian network (DBN) is used.

The results for CNNs and CRNNs show that convolutional feature processing is beneficial for drum detection. The finding considering drum detection results for multi-task learning are also promising. The low results of beat and downbeat tracking are certainly a limiting factor and probably the reason for the lack of improvement for MT over DT in the case of BGRU-b. As a next step, to better leverage multi-task learning effects, beat detection results must be improved using similar techniques as in [2].

7. CONCLUSIONS

In this work, convolutional and convolutional-recurrent NN models for drum transcription were introduced and compared to the state of the art of recurrent models. The evaluation shows that the new models are able to outperform this state of the art. Furthermore, an investigation whether *i.* beat and downbeat input features are beneficial for drum detection, and *ii.* this benefit is also achievable using multi-task learning of drums, beats, and downbeats, was conducted. The results show that this is the case, although the low beat and downbeat detection results achieved with the implemented architectures is a limiting factor. While the goal of this work was not to improve the capabilities of beat and downbeat tracking per se, future work will focus on improving these aspects, as we believe this will have an overall positive impact on the performance of the joint model. The newly created data set consisting of freely available music and annotations for drums, beats and downbeats will be an asset for this line of research to the community.

8. ACKNOWLEDGEMENTS

This work has been partly funded by the Austrian FFG under the BRIDGE 1 project *SmarterJam* (858514), by the EU's seventh Framework Programme FP7/2007-13 for research, technological development and demonstration under grant agreement no. 610591 (*GiantSteps*), as well by the Austrian ministries BMVIT and BMWFW, and the province of Upper Austria via the COMET Center SCCH. We would like to thank Wulf Gaebele and the annotators Marc Übel and Jo Thalmayer from the Red Bull Music Academy, as well as Sebastian Böck for advice and help with beat and downbeat annotations and detection.

9. REFERENCES

[1] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[2] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. 17th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2016.

[3] Sebastian Böck and Gerhard Widmer. Maximum filter vibrato suppression for onset detection. In *Proc. 16th Intl Conf on Digital Audio Effects (DAFx)*, 2013.

[4] Rich Caruana. Multitask learning. In Thrun and Pratt (eds.) *Learning to learn*, pages 95–133. Springer, 1998.

[5] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. Learning phrase representations using rnn encoderdecoder for statistical machine translation. In *Proc. Conf on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. `http://arxiv.org/abs/1412.3555`, 2014.

[7] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on nmf decomposition. In *Proc. 17th Intl Conf on Digital Audio Effects (DAFx)*, 2014.

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. IEEE conference on computer vision and pattern recognition (CVPR)*, 2015.

[9] Simon Durand, Juan P Bello, Bertrand David, and Gaël Richard. Downbeat tracking with multiple features and deep neural networks. In *Proc. 40th IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[10] Simon Durand, Juan P Bello, Bertrand David, and Gaël Richard. Feature adapted convolutional neural networks for downbeat tracking. In *Proc. 41st IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[11] Derry FitzGerald, Bob Lawlor, and Eugene Coyle. Drum transcription in the presence of pitched instruments using prior subspace analysis. In *Proc. Irish Signals & Systems Conf*, 2003.

[12] Nicolai Gajhede, Oliver Beck, and Hendrik Purwins. Convolutional neural networks with batch normalization for classifying hi-hat, snare, and bass percussion sound samples. In *Proc. Audio Mostly Conf*, 2016.

[13] Olivier Gillet and Gaël Richard. Automatic transcription of drum loops. In *Proc. 29th IEEE Intl Conf on Acoustics, Speech, and Signal Processing (ICASSP)*, 2004.

[14] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proc. 7th Intl Conf on Music Information Retrieval (ISMIR)*, 2006.

[15] Olivier Gillet and Gaël Richard. Supervised and unsupervised sequence modelling for drum transcription. In *Proc. 8th Intl Conf on Music Information Retrieval (ISMIR)*, 2007.

[16] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. `http://arxiv.org/abs/1502.03167`, 2015.

[19] Florian Krebs, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer. Downbeat tracking using beat-synchronous features and recurrent neural networks. In *Proc. 17th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2016.

[20] Florian Krebs, Sebastian Böck, and Gerhard Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proc. 15th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2013.

[21] Florian Krebs, Filip Korzeniowski, Maarten Grachten, and Gerhard Widmer. Unsupervised learning and refinement of rhythmic patterns for beat and downbeat tracking. In *Proc. 22nd European Signal Processing Conf (EUSIPCO)*, 2014.

[22] Marius Miron, Matthew EP Davies, and Fabien Gouyon. Improving the real-time performance of a causal audio drum transcription system. In *Proc. 10th Sound and Music Computing Conf (SMC)*, 2013.

[23] Marius Miron, Matthew EP Davies, and Fabien Gouyon. An open-source drum transcription system for pure data and max msp. In *Proc. 38th IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[24] Arnaud Moreau and Arthur Flexer. Drum transcription in polyphonic music using non-negative matrix factorisation. In *Proc. 8th Intl Conf on Music Information Retrieval (ISMIR)*, 2007.

[25] Jouni Paulus and Anssi Klapuri. Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009.

[26] Geoffroy Peeters and Helene Papadopoulos. Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1754–1769, 2011.

[27] Pedro HO Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *Proc. 31st Intl Conf on Machine Learning (ICML)*, Beijing, China, 2014.

[28] Jan Schlüter and Sebastian Böck. Improved musical onset detection with convolutional neural networks. In *Proc. 39th IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[29] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[30] Siddharth Sigtia, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur S d'Avila Garcez, and Simon Dixon. An RNN-based music language model for improving automatic music transcription. In *Proc. 15th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2014.

[31] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bidirectional recurrent neural networks. In *Proc. 17th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2016.

[32] Srivastava, Nitish and Hinton, Geoffrey and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[33] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*, October 2012.

[34] Christian Uhle, Christian Dittmar, and Thomas Sporer. Extraction of drum tracks from polyphonic music using independent subspace analysis. In *Proc. 4th Intl Symposium on Independent Component Analysis and Blind Signal Separation*, 2003.

[35] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proc. 17th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2016.

[36] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proc. 42nd IEEE Intl Conf on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[37] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proc. 16th Intl Society for Music Information Retrieval Conf (ISMIR)*, 2015.

[38] Kazuyoshi Yoshii, Masataka Goto, and Hiroshi G Okuno. Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):333–345, 2007.

[39] Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *Proc. IEEE Conf on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015.

A FORMALIZATION OF RELATIVE LOCAL TEMPO VARIATIONS IN COLLECTIONS OF PERFORMANCES

Jeroen Peperkamp **Klaus Hildebrandt** **Cynthia C. S. Liem**
Delft University of Technology, Delft, The Netherlands
jbpeperkamp@gmail.com {k.a.hildebrandt, c.c.s.liem}@tudelft.nl

ABSTRACT

Multiple performances of the same piece share similarities, but also show relevant dissimilarities. With regard to the latter, analyzing and quantifying variations in collections of performances is useful to understand how a musical piece is typically performed, how naturally sounding new interpretations could be rendered, or what is peculiar about a particular performance. However, as there is no formal ground truth as to what these variations should look like, it is a challenge to provide and validate analysis methods for this. In this paper, we focus on relative local tempo variations in collections of performances. We propose a way to formally represent relative local tempo variations, as encoded in warping paths of aligned performances, in a vector space. This enables using statistics for analyzing tempo variations in collections of performances. We elaborate the computation and interpretation of the mean variation and the principal modes of variation. To validate our analysis method despite the absence of a ground truth, we present results on artificially generated data, representing several categories of local tempo variations. Finally, we show how our method can be used for analyzing to real-world data and discuss potential applications.

1. INTRODUCTION

When performing music that is written down in a score, musicians produce sound that subtly differs from what is written. For example, to create emphasis, they can vary the time between notes, the dynamics, or other instrument-specific parameters, such as which strings to use on a violin or how to apply the pedals on a piano. In this paper, we focus on variations in timing, contributing a method to detect local tempo variations in a collection of performances.

Solving this problem is made difficult by the fact that it is not clear what we are trying to find: there is generally no ground truth that tells us what salient variations there are for a given piece. Furthermore, it is difficult to discern whether a given performance is 'common' or 'uncommon'.

To overcome this, we propose an approach for statistical analysis of relative local tempo variations among performances in a collection. To this end, we elaborate the computation of the mean variation and the principal modes of variation. The basis of the approach is the insight that after normalization, the set of possible tempo variations, represented by temporal warping paths, forms a convex subset of a vector space. We test our approach on artificially generated data (with controllable variations in a collection), and on recorded real performances. We discuss two applications: analysis of tempo variations and example-guided synthesis of performances.

2. RELATED WORK

2.1 Performance Analysis

Most closely related to the present work are the works in [9, 11] and [21, 22], focusing on statistical comparison of performances, targeting local tempo variations without ground truth. [9, 11] focus especially on temporal warping paths with respect to a reference performance. Furthermore, [10] analyzes main modes of variation in comparative analysis of orchestral recordings. We differ from these works in offering a more formalized perspective on variation, a more thorough and controlled validation procedure on artificially generated data, and ways to perform analyses with respect to a full collection of performances, beyond a single reference performance.

Further work in comparative performance analysis considered features such as dynamics [6]: here, it was shown that dynamic indications in a score do not lead to absolute realizations of loudness levels. [8] and [1] provide comparative analyses on many expressive features, although the latter work also finds that musicians find it difficult to think about the aspects of their performance in the quantitative fashion that is common in the MIR literature.

The absence of a clear-cut ground truth also poses challenges when automatically creating a natural-sounding rendition of a piece of music, as noted in [3] as well as [26]. Indeed, the system in the latter work explicitly relies "on a 'correct' or 'appropriate' phrase structure analysis", suggesting it is not trivial to get such an analysis.

Quite some work has also gone into the task of structure analysis, e.g. [12, 14–16, 18, 19, 23]. It turns out, however, that for some genres, the structure may be perceived ambiguously, as observed with professional annotators [23], performers [17] and listeners [24].

2.2 Dynamic Time Warping

For obtaining temporal warping paths between performances, we use Dynamic Time Warping (DTW). In a nutshell, DTW matches points from one time series to points from another time series such that the cumulative distance between the matched points is as small as possible, for some suitable distance function; the matching can then be interpreted as a warping path. A thorough overview of DTW is given in [13].

3. FORMAL ANALYSIS FRAMEWORK

We start with a formalization of tempo variations and then describe the proposed statistical analysis. The tempo variations we consider can be described by warping paths, which can be obtained from recordings of performances by using DTW.

3.1 Formal Properties

We wish to compare tempo variations between different performances of a piece. In this section, we consider an idealized setting in which only the local tempo is varied. In the next section, we will discuss how this can be used for analyzing variations in actual performances.

For our formal framework, we first need a representation of a performance. We will call the reference performance $g : [0, l_g] \to \mathbb{R}^d$, with l_g the length of the performance and d the dimensionality of some suitable feature space in which the performance can be represented. Other performances in a collection, displaying tempo variations with respect to the reference performance, can be defined as follows:

Definition 1. *A performance of g with varied tempo is a function $f = g \circ \psi : [0, l_f] \to \mathbb{R}^d$, with l_f and d defined as above, and $\psi : [0, l_f] \to [0, l_g]$ a function with nonnegative derivative, i.e., $\dot{\psi} \geq 0$. We call ψ a tempo variation.*

For the analysis of tempo variations between f and g, we distinguish between average and relative tempo variation. The average tempo variation can be observed by looking at the length of the interval over which the functions are parametrized; it is simply the difference in average overall tempo of each performance. Clearly, the longer the interval, the slower the performance is on average. There is more structure in the details, of course, which is what the relative variations attempt to capture. Specifically, this refers to an analysis of tempo variations given that the performances are parametrized over an interval of the same length, for instance, the unit interval.

Now, to implement the concept of relative tempo variations, we first reparametrize the performances over the unit interval. Given $f : [0, l_f] \to \mathbb{R}^d$, we consider the normalized performance $f^* = f \circ \rho : [0, 1] \to \mathbb{R}^d$, where $\rho : [0, 1] \to [0, l_f]$ is given by $\rho(t) = l_f t$. Now we can go into more detail about these relative tempo variations.

3.1.1 Structure of the Set of Relative Tempo Variations

Relative tempo variations can be described by reparametrizations that relate the performances in question. Due to the normalization of the performances, the reparametrizations map the unit interval to itself. The relative tempo variations φ and their derivatives $\dot{\varphi}$ are characterized by the following two properties:

Property 1. $\varphi(0) = 0$, $\varphi(1) = 1$.

Property 2. $\dot{\varphi}(n) \geq 0$ *for any* $n \in [0, 1]$.

Examples of such relative tempo variations are shown in Figure 1 (left), along with insets to see what happens when one zooms in. When working with the normalized performances, every performance with varied tempo f^* of a reference performance g^* has the form $f^* = g^* \circ \varphi$.

The benefit of splitting average and relative variation is that the set of relative variations has a geometric structure: the following lemma shows that it is a convex set in an vector space. This enables us to use classical methods from statistical analysis to analyze the relative tempo variations, as explained in Section 3.2.

Lemma 1. *Convex combinations of relative tempo variations are relative tempo variations.*

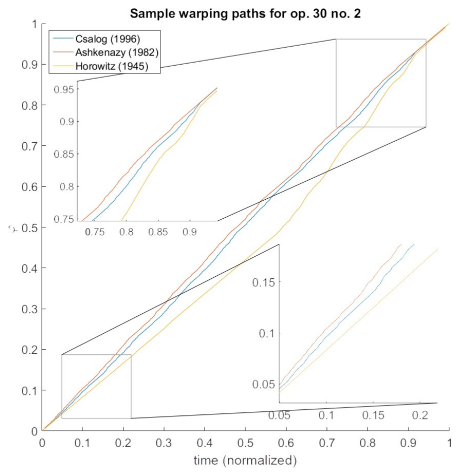
Proof. Let $\alpha = (\alpha_1, \dots, \alpha_m)$ be a vector of nonnegative numbers, $\alpha_i \geq 0$, with unit ℓ_1 norm, $\sum_{i=1}^m \alpha_i = 1$, and let $\varphi_i : [0, 1] \mapsto [0, 1]$ be relative tempo variations ($1 \leq i \leq m$). We show that $\varphi = \sum_{i=1}^m \alpha_i \varphi_i$ is a relative tempo variation. As a sum of functions on the unit interval, φ is also a function on the unit interval. Since the α_i sum to 1, $\sum_{i=1}^m \alpha_i \varphi_i(0) = 0$ and $\sum_{i=1}^m \alpha_i \varphi_i(1) = 1$, which means that Property 1 holds. Finally, since all α_i are nonnegative, $\dot{\varphi} \geq 0$ is also maintained. □

3.2 Analysis of Prominent Variations

In the following, we consider a set of performances (with varied tempo) and show how our approach allows us to compute statistics on the set. Explicitly, we take the mean and perform principal component analysis (PCA). As a first step, we reparametrize the performances over the unit interval $[0, 1]$, as described above. We distinguish two settings for our analysis. First, we describe a setting in which we consider one reference performance. An example of such a reference performance in practice is a rendered MIDI, which has a linear timing to which we relate the actual performances in the set. In the second setting, we avoid the use of a reference performance by incorporating all pairwise comparisons between performances.

3.2.1 Comparing to the Reference Performance

Comparing a set of performances $\{f_1, f_2, \dots, f_n\}$ to a reference g^* means obtaining for each normalized performance f_i^* the corresponding relative tempo variation φ_i, such that $f_i^* = g^* \circ \varphi_i$. Lemma 1 shows that we can build a continuous set of relative tempo variations by building convex combinations. Geometrically speaking, we consider the simplex spanned by the φ_i. Though not needed



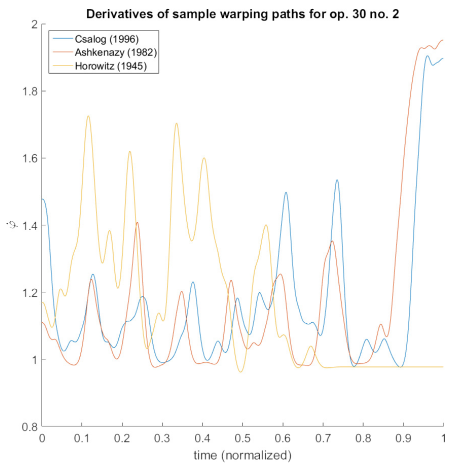


Figure 1. Several reparametrizations φ relating professional human performances of Chopin's Mazurka op. 30 no. 2 to a deadpan MIDI version. Original φ with zoomed insets (left) and their derivatives $\dot{\varphi}$ (right).

for our analysis, extrapolation out of the simplex is possible, as long as Property 2 is satisfied.

A particularly interesting convex combination for our purposes is the mean of the set of performances. The mean relative tempo variation $\bar{\varphi}$ can be computed by setting all the α_i to the same value in Lemma 1 above. The mean of the normalized performances $\{f_i^*\}$ is given as $g^* \circ \bar{\varphi}$. To obtain the mean of the performances, $g^* \circ \bar{\varphi}$ is linearly rescaled to the average length of the performances f_i. The mean $\bar{\varphi}$ gives information about which local tempo variations away from g^* are the most prevalent among the performances under analysis. Of course, the mean does not capture the variance in the set, for example, deviations in opposite directions, as when some performers speed up and others slow down, which would be evened out.

The variance in a set can be analyzed using PCA. To perform a PCA on the set φ_i, we need a scalar product on the space of relative tempo variations. Since these are functions on the unit interval, any scalar product on this function space can be used. For our experiments, we used the L^2-scalar product of the derivatives of the functions (in other words the Sobolev H_0^1-scalar product). The reason for using a scalar product of the derivatives, rather than the function values, is that the derivatives describe the variations in tempo, and the function values encode the alignment of the performance. See Figure 1 (right) for an example of how this brings out the variation. Once a scalar product is chosen, we construct the covariance matrix, whose entries are the mutual scalar products of the functions $\varphi_i - \bar{\varphi}$ (the distance of the tempo variations to the mean). The eigenvectors of the covariance matrix yield the principal modes of variation in the set φ_i. These express the main variations away from the mean in the set and the eigenvalues indicate how much variance there is in the set of performances by how much of the variance is explained by the corresponding modes. The modes express the tendency of performers to speed up or slow down observed in the set of performances.

3.2.2 Incorporating All Pairwise Comparisons

When using a reference performance, one has to choose which performance to use as g^*, or to produce an artificial performance for g^* (as we do in Section 4). This way, the comparison becomes dependent on the choice of g^*, which may not be desirable, as there may be 'outlier' performances that would not necessarily be the best choice for a reference performance (though other things can be learned from them [17]).

To avoid the need to choose g^*, we propose an alternative analysis using all pairwise comparisons. This means obtaining reparametrizations φ for every pair of performances f^* and g^* such that $f^* = g^* \circ \varphi$. This makes sense, as it is not guaranteed that for three normalized performances f^*, g^* and h^* and reparametrizations φ_i and φ_j such that $g^* = f^* \circ \varphi_i$ and $h^* = g^* \circ \varphi_j$, we would get $h^* = f^* \circ \varphi_i \circ \varphi_j$. In other words, reparametrizations may violate the triangle inequality, so we obtain more information by taking into account all possible reparametrizations.

The same techniques can be applied once we have the (extended) set of reparametrizations φ. That is, we can take the mean of all the φ or perform a PCA on them. Empirically, it turns out there tends to be repeated information in the reparametrizations, which results in a certain amount of natural smoothing when taking the mean; this effect can be seen in Figure 3.

4. EXPERIMENTAL VALIDATION

In Section 3, we considered a collection of performances with tempo variations as compared to a reference performance. To perform the analyses described, we take the following steps. First, we map the audio into some suitable feature space; we take the chroma features implemented in the MIRtoolbox [7] to obtain sequences of chroma vectors. We then normalize these sequences to functions over the unit interval. Finally, we use DTW to compute the relative tempo variations φ that best align the performances.

Explicitly, let $f^*, g^* : [0,1] \to \mathbb{R}^d$ be sequences of chroma vectors (in our case, $d = 12$, as analysis at the semitone resolution suffices). Then DTW finds the function φ that satisfies Properties 1 and 2 and minimizes $\|f^* - (g^* \circ \varphi)\|_2$, i.e., the L^2 norm of the difference between f^* and the reparametrized g^*. We generate φ in this way for all performances in the collection.

Our goal is to analyze variations between performances. Local tempo variation should be reflected in φ, provided there is not too much noise and the same event sequence is followed (e.g. no inconsistent repeats). The way we bring out the local tempo variation is by taking the derivative $\dot{\varphi}$ (cf. Section 3.2). A derivative larger/smaller than 1 indicates that the tempo decreases/increases relative to the reference performance. Since the tempo variations are given as a discrete functions, we need to approximate the derivatives. We do this by fitting a spline to the discrete data and analytically computing the spline's derivative.

To avoid the ground truth issue mentioned in Section 2, we devise several classes of artificial data, representing different types of performance variations for which we want to verify the behavior of our analysis. We verify whether the analysis is robust to noise and uniform variation in the overall tempo (the scalar value mentioned in Section 3). Furthermore, we consider different types of local tempo variations, which, without loss of generalization, are inspired by variations typically expected in classical music performances.

In the previous section, we mentioned two possible analysis strategies: considering alignments to a reference performance or between all possible pairs of performances. Since the artificial data are generated not to have outliers, it is difficult to apply the analysis that uses all possible pairs to the artificial data. We therefore focus on the case of using a single reference performance, although we will briefly return to the possibility of using all pairs in Section 5.

4.1 Generating Data

The data were generated as follows. We start with a sequence $g \in \mathbb{R}^{12\times m}$ of m 12-dimensional Gaussian noise vectors. Specifically, for each vector g_i, each element $g_{i,j}$ is drawn from the standard normal distribution $N(0, 1)$. We then generate a collection $\mathcal{C}$ of 'performances' based on g, for seven different variation classes. We normalize the vectors in $\mathcal{C}$ such that each element is between 0 and 1, as it would be in natural chroma vectors. The classes are defined as follows:

Class 1: Simulate minor noise corruption. A new sequence c is generated by adding a sequence $h \in \mathbb{R}^{12\times m}$ of 12-dimensional vectors, where each element $h_{i,j} \sim N(0, \frac{1}{4})$, so $c = g + h$. We expect this does not lead to any significant alignment difficulty, so the derivative of the resulting $\bar{\varphi}$ (which we will call $\dot{\bar{\varphi}}$) will be mostly flat.

Class 2: Simulate linear scaling of the overall tempo by stretching the time. Use spline interpolation to increase the number of samples in g, to simulate playing identically, but with varying overall tempo. If there are n sequences generated, vary the number of samples from $m - \frac{n}{2}$ to $m + \frac{n}{2}$. Since this only changes 'performances' on a global scale, this should give no local irregularities in the resulting $\dot{\bar{\varphi}}$.

Class 3: Simulate playing slower for a specific section of the performance, with sudden tempo decreases towards a fixed lower tempo at the boundaries, mimicking common tempo changes in an A-B-A song structure. Interpolate the sequence to have 1.2 times as many samples between indices $l = \frac{1}{3}m - \frac{1}{2}X$ and $h = \frac{2}{3}m + \frac{1}{2}X$, where $X \sim U(0, \frac{m}{10})$ (the same randomly drawn X is used in both indices). We expect $\dot{\bar{\varphi}}$ to be larger in the B part than in A parts. Since in different samples, the tempo change will occur at different times, transitions are expected to be observed at the tempo change intervals.

Class 4: A variation on class 3. Simulate a disagreement about whether to play part of the middle section slower. Let $k = h - l$. With a probability of 0.5, do not interpolate the section from $l + \frac{k}{3}$ to $h - \frac{k}{3}$. We expect similar results as for class 3 with the difference that in the middle of the B part, we expect an additional jump in $\dot{\bar{\varphi}}$. In the B part, $\dot{\bar{\varphi}}$ will jump to a lower value, which should still be larger than the value in the A part since only half of the performances decrease the tempo.

Class 5: Simulate a similar A-B-A tempo structure as in class 3, but change the tempo gradually instead of instantly over intervals of size roughly $\frac{1}{6}m$. From index $l_1 = \frac{1}{4}m - \frac{1}{2}X$ to $l_2 = \frac{5}{12}m + \frac{1}{2}X$, gradually slow down to 120% of the original tempo by interpolating over a quadratic query interval [1] , then gradually speed up again the same way between indices $h_1 = \frac{7}{12}m - \frac{1}{2}X$ and $h_2 = \frac{3}{4}m + \frac{1}{2}X$. Here, $X \sim U(0, \frac{1}{18}m)$ and is drawn only once. Here again, we expect to see smaller values of $\dot{\bar{\varphi}}$ in the A parts and a higher value in the B part. Due to the gradual change in tempo, we expect a more gradual transition between A-B and B-A.

Class 6: A variation on class 5. Instead of varying the interval using X, vary the tempo. First speed up the tempo by a factor $1.3 + Y$ times the starting value (with $Y \sim U(-\frac{1}{10}, \frac{1}{10})$), then gradually slow down to a lower tempo and again speed up before the regular tempo of A is reached again. Here we expect to see a peak in $\dot{\bar{\varphi}}$ at the transition from A to B, before the lower value in the B part is reached and again a peak in the transition from B to A.

Class 7: Another variation on class 5: disagreement about speeding up or slowing down. Toss a fair coin ($p = 0.5$); on heads, gradually increase the tempo between l_1 and l_2 to $1.2+Y$ times the starting value and decrease it again between h_1 and h_2 as in class 5. On tails, decrease the tempo to $0.8+Y$ times the starting value between l_1 and l_2 and increase it again between h_1 and h_2, with $Y \sim U(-\frac{1}{10}, \frac{1}{10})$. We expect this to give much more noisy alignment, though there may be a more stable area in $\dot{\bar{\varphi}}$ where the tempos do not change, even though they are different.

[1] Normal linear interpolation corresponds to a constant tempo curve, but if the tempo curve changes linearly, the query interval for interpolation becomes quadratic.

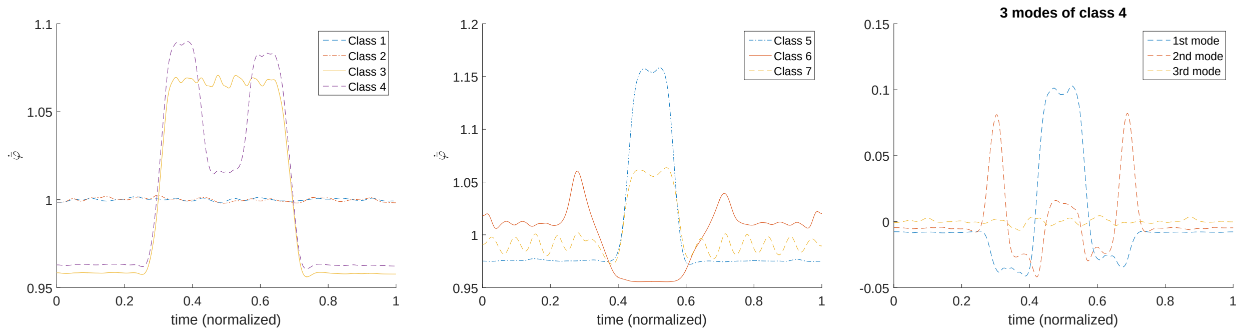


Figure 2. On the left: $\dot{\bar{\varphi}}$ for class 1–4. In the middle, $\dot{\bar{\varphi}}$ for class 5–7. On the right: the first three PCA modes for class 4.

When running our analysis on the classes of artificial data thus generated, we always took $m = 500$ and generated 100 sequences for each class. We used Matlab to generate the data, using 2017 as the seed for the (default) random number generator. A GitHub repository has been made containing the code for the analysis and for generating the test data [2]. The experiment was run 100 times, resulting in 100 $\bar{\varphi}$s and 100 sets of PCA modes; we took the mean for both and show the results in figures: Figure 2 (left and middle) show the derivatives when taking the mean (each time) as described in Section 3, while Figure 2 (right) shows what happens when taking the PCA, as also described in Section 3. We show the first three modes because these empirically turn out to cover most (around 90%) of the variance.

4.2 Discussion

We now briefly discuss what the analyses on artificial data tell us. First of all, the observed outcomes match our expectations outlined above. This demonstrates that our analysis can indeed detect the relative tempo variations that we know are present in performances of music.

We want to note that Figure 2 shows the derivatives of the relative tempo variation. For example, for class 3, all performances are shorter than the reference; therefore, they are stretched during the normalization. Consequently, the $\dot{\bar{\varphi}}$ in part A in the normalized performance is smaller than 1. This effect could be compensated by taking the length of the performances into account.

The PCA modes provide information about the variation in the set of performances. Figure 2 shows the first three modes found in Class 4. These three modes are the most dominant and explain more than 90% of the variation. The first mode has a large value in the middle part of the B section. This follows our expectation as only 50% of the performances slow down in this part, hence we expect much variation in this part. In addition, there are small values in the other parts of the B section. This is due to the fact that the performances do not speed up at the same time, so we expect some variation in these parts. Note that the principal modes are linear subspaces, hence sign and scale of the plotted function are arbitrary. An effect of this is that the modes cannot distinguish between speeding up the tempo or slowing it down. Since the first mode captures the main variation in the middle part of the B section, in the second mode the transitions between A and B are more emphasized. The third mode emphasizes the transitions too.

Finally, we note that it becomes possible to zoom in on a particular time window of a performance, in case one wants to do a detailed analysis. A hint of this is shown in Figure 1, left, where zoomed versions of φ are shown in insets. We have defaulted in our experiments to analyzing performances at the global level, and consider it future work to explore what information will be revealed when looking at the warping paths up close.

5. APPLICATIONS

Now that we have validated our approach, we describe several applications in which our method can be employed.

5.1 Analyzing Actual Performances

As mentioned in Section 3, we can analyze relative differences between a chosen reference performance and the other performances, or between all possible pairs of performances. We have access to the Mazurka dataset consisting of recordings of 49 of Chopin's mazurkas, partially annotated by Sapp [21]. Note that our analysis can handle any collection of performances and does not require annotations. Since we have no ground truth, it is difficult to make quantitative statements, but in this and the following subsection, we will discuss several illustrative qualitative examples.

In Figure 3, we show $\dot{\bar{\varphi}}$ for Mazurka op. 30 no. 2 for both approaches. Taking all pairs into consideration results in lower absolute values, as well as an apparent lag. For both approaches, it turns out the most important structural boundaries generally show up as the highest peaks. Another feature that stands out in both plots is the presence of peaks at the beginning and end. These can be interpreted as boundary effects, but we believe the final peak also is influenced by intentional slowing down by the musicians in a final retard [25].

Another example of applying the analysis on all pairs of performances is given in Figure 4. Here, we see two more

[2] `https://github.com/asharkinasuit/ismir2017paper`.

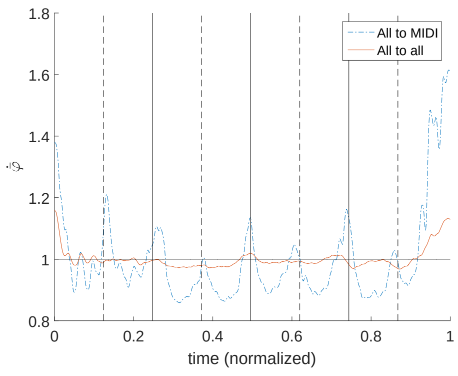


Figure 3. Sample showing $\dot{\bar{\varphi}}$ for Mazurka op. 30 no. 2, comparing warping to a deadpan MIDI and warping everything to everything. Note the smoothing effect in the latter case. Salient structural parts are indicated with vertical lines: repeats (dotted) and structural boundaries (solid).

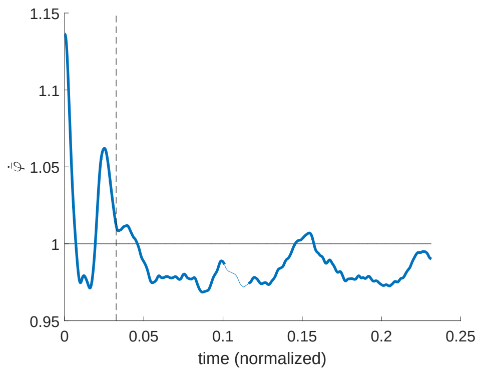


Figure 4. $\dot{\bar{\varphi}}$ of the start of mazurka op. 17 no. 4. The start of the melody is marked with a vertical dashed bar, while the *delicatissimo* section is drawn in a thinner line.

interesting features of the analysis. Firstly, it tends to hint at the musicians' interpretation of the structure of the piece (as also in Figure 3); the start of the melody is indicated with the vertical dashed line. Most performers emphasize this structural transition by slowing down slightly before it. However, the time at which they slow down varies slightly (compare this to e.g. class 3 and 5 of our artificial data). This will show in φ, and consequently in $\dot{\bar{\varphi}}$. Secondly, we note that ornaments tend not to vary tempo as much: the thin section in the figure is closer to 1 than the peak near the start of the melody. This helps corroborate Honing's results, e.g. [2, 5].

5.2 Guiding Synthesis

For the performances in question, we know the piece that is performed and we have a score available. A direct acoustic rending of the score (via MIDI) would sound unnatural. Now, reparametrizations and their means are just functions, which we can apply to any other suitably defined function. Following the suggestion in [20] that a generated 'average' performance may be more aesthetically pleasing, we can now use these functions for this: by applying the $\bar{\varphi}$ derived from a set of performances to a MIDI rendition, a more natural-sounding result will indeed be obtained. As an example, we ran our analysis on Chopin's mazurka op. 24 no. 2 with the MIDI rendition as reference performance and applied the resulting reparametrization to the MIDI[3]. Note that, as in Figure 3, the tempo naturally decreases towards the end.

Applying $\bar{\varphi}$ directly to audio is not the only thing that we can do. One possibility is exaggeration of tempo variation. To amplify sections that show major tempo variation, we can modify the φ by squaring it. Alternatively, to better display the tempo variations in an individual performance, we can rescale the function $\varphi - \bar{\varphi}$, capturing the difference of the actual performance to the mean in a performance collection. Such modifications offer useful analysis tools for bringing out more clearly the sometimes subtle effects employed by professional musicians.

Another possibility is to take φ from various sources, e.g., by generating φ for several different reference performances, and applying them to a MIDI rendition with various coefficients to achieve a kind of mixing effect. Finally, the principal modes of variation in the set can be used to modify the tempo in which the MIDI is rendered. Example audio files are available on request for any of these different ways of rendering musical scores using information from actual performances.

6. CONCLUSIONS AND FUTURE WORK

We have presented a formal framework for analyzing relative local tempo variations in collections of musical performances, which enables taking the mean and computing a PCA of these variations. This can be used to analyze a performed piece, or synthesize new versions of it.

Some challenges may be addressed in the future. One would be to give a more rigorous interpretation to the case of taking all pairwise comparisons into account. Furthermore, quantification of variation still presently is used in a relative fashion; our analysis indicates some amount of variation, but further interpretation of this amount would be useful. One might also substitute other DTW variants that can e.g. deal more intuitively with repeat sections [4].

Furthermore, while the studied variation classes were inspired by local tempo variations in classical music performances, it should be noted that our framework allows for generalization, being applicable to any collection of alignable time series data. Therefore, in future work, it will be interesting to investigate applications of our proposed method on other types of data, such as motion tracking data.

7. REFERENCES

[1] A. Benetti Jr. Expressivity and musical performance: practice strategies for pianists. In *2nd Performance Studies Network Int. Conf.*, 2013.

[3] See `https://github.com/asharkinasuit/ismir2017paper`, which includes the original for comparison.

[2] P. Desain and H. Honing. Does expressive timing in music performance scale proportionally with tempo? *Psychological Research*, 56(4):285–292, 1994.

[3] S. Flossmann, M. Grachten, and G. Widmer. Expressive Performance Rendering with Probabilistic Models. In *Guide to Computing for Expressive Music Performance*, pages 75–98. Springer, 2013.

[4] M. Grachten, M. Gasser, A. Arzt, and G. Widmer. Automatic alignment of music performances with structural differences. In *ISMIR*, 2013.

[5] H. Honing. Timing is Tempo-Specific. In *ICMC*, 2005.

[6] K. Kosta, O. F. Bandtlow, and E. Chew. Practical Implications of Dynamic Markings in the Score: Is Piano Always Piano? In *53rd AES Conf. on Semantic Audio*, 2014.

[7] O. Lartillot and P. Toiviainen. A matlab toolbox for musical feature extraction from audio. In *Int. Conf. Digital Audio Effects*, pages 237–244, 2007.

[8] E. Liebman, E. Ornoy, and B. Chor. A phylogenetic approach to music performance analysis. *Journal of New Music Research*, 41(2):195–222, 2012.

[9] C. C. S. Liem and A. Hanjalic. Expressive Timing from Cross-Performance and Audio-based Alignment Patterns: An Extended Case Study. In *ISMIR*, pages 519–524, 2011.

[10] C. C. S. Liem and A. Hanjalic. Comparative analysis of orchestral performance recordings: an image-based approach. In *ISMIR*, 2015.

[11] C. C. S. Liem, A. Hanjalic, and C. S. Sapp. Expressivity in musical timing in relation to musical structure and interpretation: a cross-performance, audio-based approach. In *42nd AES Conf. Semantic Audio*, 2011.

[12] L. Lu, M. Wang, and H. Zhang. Repeating pattern discovery and structure analysis from acoustic music data. In *6th ACM SIGMM Int. Workshop on Multimedia Information Retrieval*, pages 275–282. ACM, 2004.

[13] M. Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[14] M. Müller and S. Ewert. Joint Structure Analysis with Applications to Music Annotation and Synchronization. In *ISMIR*, pages 389–394, 2008.

[15] M. Müller and F. Kurth. Enhancing similarity matrices for music audio analysis. In *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, volume 5. IEEE, 2006.

[16] O. Nieto and T. Jehan. Convex non-negative matrix factorization for automatic music structure identification. In *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 236–240. IEEE, 2013.

[17] M. Ohriner. What can we learn from idiosyncratic performances? Exploring outliers in corpuses of Chopin renditions. In *Proc. of the Int. Symp. on Performance Science*, pages 635–640, 2011.

[18] Y. Panagakis, C. Kotropoulos, and G. R. Arce. ℓ_1-graph based music structure analysis. In *ISMIR*, 2011.

[19] J. Paulus and A. Klapuri. Music structure analysis by finding repeated parts. In *Proc. of the 1st ACM Audio and Music Computing Multimedia Workshop*, pages 59–68. ACM, 2006.

[20] B. H. Repp. The aesthetic quality of a quantitatively average music performance: Two preliminary experiments. *Music Perception: An Interdisciplinary Journal*, 14(4):419–444, 1997.

[21] C. S. Sapp. Comparative Analysis of Multiple Musical Performances. In *ISMIR*, pages 497–500, 2007.

[22] C. S. Sapp. Hybrid numeric/rank similarity metrics for musical performance analysis. In *ISMIR*, pages 501–506, 2008.

[23] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *IEEE Trans. Multimedia*, 16(5):1229–1240, 2014.

[24] J. B. L. Smith, I. Schankler, and E. Chew. Listening as a Creative Act: Meaningful Differences in Structural Annotations of Improvised Performances. *Music Theory Online*, 20(3), 2014.

[25] J. Sundberg and V. Verrillo. On the anatomy of the retard: A study of timing in music. *Journal of the Acoustical Society of America*, 68:772–779, 1980.

[26] G. Widmer and A. Tobudic. Playing Mozart by Analogy: Learning Multi-level Timing and Dynamics Strategies. *Journal of New Music Research*, 32(3):259–268, 2003.

Oral Session 6

Structure

SAMPLING VARIATIONS OF SEQUENCES FOR STRUCTURED MUSIC GENERATION

François Pachet
Sony CSL Paris
pachet@gmail.com

Alexandre Papadopoulos
UPMC Univ Paris 06, UMR 7606, LIP6
alexandre.papadopoulos@lip6.fr

Pierre Roy
Sony CSL Paris
roypie@gmail.com

ABSTRACT

Recently, machine-learning techniques have been successfully used for the generation of complex artifacts such as music or text. However, these techniques are still unable to capture and generate artifacts that are convincingly structured. In particular, musical sequences do not exhibit pattern structure, as typically found in human composed music. We present an approach to generate structured sequences, based on a mechanism for sampling efficiently variations of musical sequences. Given an input sequence and a statistical model, this mechanism uses belief propagation to sample a set of sequences whose distance to the input sequence is approximately within specified bounds. This mechanism uses local fields to bias the generation. We show experimentally that sampled sequences are indeed closely correlated to the standard musical similarity function defined by Mongeau and Sankoff. We then show how this mechanism can be used to implement composition strategies that enforce arbitrary structure on a musical lead sheet generation problem. We illustrate our approach with a convincingly structured generated lead sheet in the style of the Beatles.

1. INTRODUCTION [1]

Recent advances in machine learning, especially deep recurrent networks such as LSTMs, led to major improvements in the quality of music generation [7, 10]. They achieve spectacular performance for short musical fragments. However, musical structure typically exceeds the scope of statistical models. As Waite recently wrote, the music produced by recurrent models tend to lack a sense of direction and becomes boring after a short while [15]. Pionneering works on music composition with LSTMs already showed how some structure, such as chord structure [6] or metrical structure [5] can be *spontaneously* captured, but the general problem of generating music with repetitive long-term structure remains open. In this paper, we propose a method to *explicitly* enforce such structure in a controlled way, in a "templagiarism" fashion [2, p. 49].

Musical structure is the overall organisation of a composition into sections, phrases, and patterns, very much like the organisation of a text. The structure of musical pieces is scarcely, if ever, linear as it essentially relies on the repetition of these elements, possibly altered. For example, songs are decomposed into repeating sections, called verses and choruses, and each section is constructed with repeating patterns. It has been shown that the *listeners' emotional arousal responses to music* is correlated with the degree of similarity between musical fragments (high for repetitions, moderate for variations, and least for contrasting segments) [9]. In fact, the striking *speech to song* illusion discovered by [4] shows that repetition truly *creates* music, for instance by turning speech into music. This is further confirmed by [11] who observed that inserting arbitrary repetition in non-repetitive music improves listeners rating and confidence that the music was written by a human composer.

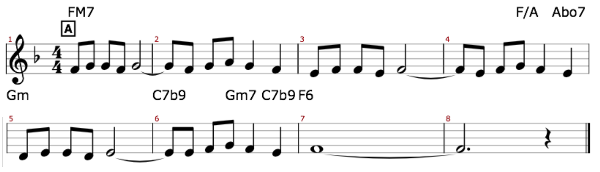


Figure 1. The last eight bars of *"Strangers in the Night"*.

Variations are a specific type of repetition, in which the original melody is altered in its rhythm, pitch sequence, and/or harmony. Variations are used to create diversity and surprise by subtle, unexpected changes in a repetition. The song *"Strangers in the Night"* is a typical 32-bar form with an AABA structure consisting of four 8-bar sections. The three **A** sections are variations of each other. The last **A** section, shown in Figure 1, consists of a two-bar cell which is repeated three times. Each occurrence is a subtle variation of the preceding one. The second occurrence (bars 3-4) is a mere transposition of the original pattern by one descending tone. The third instance (bars 5-6) is also transposed, but with a slight modification in the melody, which creates a surprise and concludes the song. Bars 5-6 are both a variation of the original pattern in bars 1-2. Current models for music generation fail to reproduce such

[1] Authors are listed alphabetically: Pachet originated the general problem and contributed musical examples; Papadopoulos developed and implemented the technical solution especially the integration with the regular belief propagation model, devised and performed the evaluation procedure; Roy brought the original idea and the technical solution, developed the first prototype and the structured lead sheet generation procedures.

long-range similarities between musical patterns. In this example, it is statistically unlikely that bars 5-6 be almost identical to bars 1-2.

Our goal is to generate such structured musical pieces from statistical models. Our approach is to impose a predefined musical structure that specifies explicitly repetitions and variations of patterns and sections, and use a statistical model to generate music that "instantiates" this structure. In this approach, musical structure is viewed as a procedural process, external to the statistical model.

Our approach subsumes previous attempts at generating music with an imposed long-term structure with Markov models such as [1]. Their approach lacks both a variation mechanism and a constrained Markov model. As a result, it is limited to strict repetitions of patterns. Furthermore, the use of ad hoc joining techniques to glue copied fragments, violates the Markov model, resulting in unnatural transitions.

An essential ingredient to implementing our approach is a mechanism to generate variations of a given musical pattern from a statistical model. Although it is impossible to characterise formally the notion of variation, it was shown that some measures of melodic similarity are efficient at detecting variations of a theme [12]. We propose to use such a similarity measure in a generative context to sample from a Markov model, patterns that are similar to a given pattern. This method is related to work on stochastic edit distances [3, 14], but is integrated as a constraint in a more general model for the generation of musical sequences [13]. Moreover, our approach relies on an existing similarity measure rather than on labeled data (pairs of themes and related variations), which is not available. Similar approaches exist in the context of text generation. For example, [8] propose a model using a technique based on skip vectors. They train a model that learns the similarity between sentences. Using this model, they can predict the semantic relatedness of two sentences, a standard similarity measure for text, but they can also generate sentences similar to an existing sentence.

We remind the Mongeau & Sankoff similarity measure [12] between melodies, and then, we describe our model for sampling melodic variations based on this similarity, which we validate experimentally, Finally, we show examples of variations of a melody, and a longer, structured musical piece generated with an imposed structure.

2. MELODIC SIMILARITY

The traditional string edit distance considers three editing operations: *substitution*, *deletion*, and *insertion* of a character. Mongeau and Sankoff [12] add two operations motivated by the specificities of musical sequences, and inspired by the time compression and expansion operations considered in *time warping*. The first operation, called *fragmentation*, involves the replacement of one note by several, shorter notes. Similarly, the *consolidation* operation, is the replacement of several notes by a single, longer note. Mongeau and Sankoff proposed an algorithm to compute the similarity between melodies in polynomial time. Considering melodies as sequences of notes, the algorithm, based on dynamic programming, computes MGD(A, B), the measure of similarity between the sequences of notes A and B. Note that this is not a distance, in particular MGD(A, B) is not necessarily equal to MGD(B, A).

The Mongeau & Sankoff measure is well-adapted to the detection of variations, but has a minor weakness: there is no penalty associated with fragmenting a long note into several shorter notes of same pitch and same total duration. The same applies to consolidation. This is not suited to a generative context, as fragmentation or consolidation change the resulting melody.

In the dynamic programming recurrence equation given in their paper [12], Mongeau and Sankoff introduce various weight functions, denoting predefined local weights associated with the basic editing operations (substitution, deletion, insertion, fragmentation and consolidation). We modify the original measure by adding a penalty p to the weights of the consolidation and fragmentation operations.

The weight associated with a fragmentation of a note a_i into a sequence of notes $b_{j-k+1}, \ldots, b_j$ is:

$$\begin{aligned} w_{frag}(a_i, b_{j-k+1}, \ldots, b_j) &= w_{pitch}(a_i, b_{j-k+1}, \ldots, b_j) \\ &\quad + k_1 n(a_i, b_{j-k+1}, \ldots, b_j) + p \end{aligned}$$

For consolidation, a similar extra-weight is added. The consolidation weight is defined by:

$$\begin{aligned} w_{cons}(a_i, b_{j-k+1}, \ldots, b_j) &= w_{pitch}(a_i, b_{j-k+1}, \ldots, b_j) \\ &\quad + k_1 n(a_i, b_{j-k+1}, \ldots, b_j) + p. \end{aligned}$$

3. A MODEL FOR THE GENERATION OF MELODIC VARIATIONS

Given an original *theme*, i.e. a melodic fragment, we generate variations of this theme by sampling a specific graphical model. This graphical model is a modified version of the general model of lead sheets introduced by [13]. We now briefly describe this general model and explain how we bias it to produce only melodies at a *controlled* Mongeau & Sankoff distance from the theme, the core technical contribution of this paper. For full explanations and implementation details, we refer the reader to [13].

3.1 The Model of Lead Sheets

The overall model comprises *two graphical models*, one for chord sequences, one for melodies. Both models are based on a factor graph that combines a Markov model with a finite state automaton. The Markov model, trained on a corpus of lead sheets, provides the stylistic model. The automaton represents hard temporal constraints that the generated sequences should satisfy, such as metrical properties (e.g., an imposed total duration) or user imposed temporal constraints.

Each factor graph is made of a sequence of variables, represented with circles, encoding the sequence of elements, related to unary and binary factors, represented by squares. In this model, a *variable* is not associated with

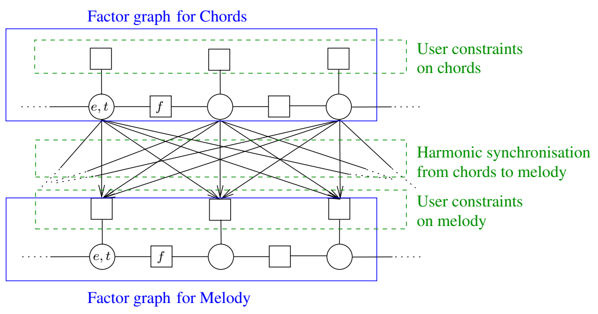

Factor graph for Chords
User constraints on chords
e,t
f
Harmonic synchronisation from chords to melody
User constraints on melody
e,t
f
Factor graph for Melody

Figure 2. The two-voice model for lead sheet generation

a specific temporal position in the sequence, but the *values* it takes specifies its temporal position. Each value is a chord or a note e, with a fixed duration $d(e)$ *along with* its temporal position t in the sequence. This is a very powerful property of this model. It allows us to specify unary temporal constraints, e.g., the second bar should start with a rest. It also allows us to specify harmonic relations between the chord sequence and the melody, e.g., the note at time t should be compatible with the chord at time t. Crucially, we will exploit this property to implement our variation mechanism.

A binary factor is a conditional probability $f\left((e,t)|(e',t')\right)$ on transitions between elements. In [13], the authors use binary factors to combine the Markov transition probabilities with the finite-state automaton transitions. Harmonic relationship between chords and notes are also specified by binary factors.

The graphical model defines a distribution $p(e_1,\ldots,e_n)$ over the sequence of variables defined by the product of all unary and binary factors. A belief propagation-based procedure samples successively the two models by taking into account partially filled fragments and propagating their effect to all empty sections.

3.2 Generating Variations of a Theme

We introduce an extra binary factor $\beta(e|t,e')$: the probability of placing element e at time t and preceded by element e'. We will use β to implement the variation mechanism. In practice, this additional binary factor is simply multiplied with the existing binary factors, without affecting the structure of the model on Figure 2. The probability p' of a sequence in the resulting model becomes:

$$p'(e_1,\ldots,e_n) = p(e_1,\ldots,e_n)\prod_{i=2}^{n}\beta(e_i|t,e_{i-1}).$$

We set the value of $\beta(e|t,e')$ according to a "localised" similarity measure between the sequence $[e',e]$ and the fragment of the theme between $t-d(e')$ and $t+d(e)$. Biases are set so that a bias of 1 does not modify the probability of putting element e at time t after e', and a bias less than 1 decreases this probability.

The lead sheet in Figure 3 shows the first four bars of *"Solar"* by Miles Davis. Suppose we train a lead sheet model on a corpus of all songs by Miles Davis. Sampling

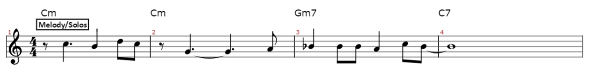


Figure 3. The first four bars of *"Solar"*, by Miles Davis.

this model produces new lead sheets in the style of Miles Davis, but not necessarily similar to Solar specifically. To favour sequences with the same notes as the theme is to set the β factors so that:

- $\beta(n|t,n')=1$ if the melodic fragment consisting of note n' followed by note n at position t appears in the theme, e.g., we set $\beta(\text{C}_5|t=1.5,rest)=1$ for note C$_5$ *dotted quarter note*;
- $\beta(n|t,n')<1$ otherwise, and the value of $\beta(n|t,n')$ will be set to very small values (close to zero), if the melodic fragment made by n' and n at time t is very different, musically, from the corresponding melodic fragment in the theme, e.g., $\beta(\text{F}\sharp_4|t=1.5,\text{G}\flat_5)\ll 1$. On the contrary, if the two fragments are very similar, musically, the value of $\beta(n|t,n')$ will be set to a value closer to 1, e.g., $\beta(\text{C}_5|t=1.5,rest)\gg 0$ for note C$_5$ *quarter note*.

More precisely, we evaluate the similarity between each possible note n at a given position t, preceded by note n' in the generated sequence, and the notes in the theme around position t. We then set each bias $\beta(n|t,n')$ based on this similarity measure.

Technically, for every candidate note n, we consider all potential temporal positions t and all potential predecessors n'. We compute MGD$([n',n],t)$, the Mongeau & Sankoff similarity between the two-note melody $[n',n]$ and the melodic fragment of the theme between time $t-d(n')$ and $t+d(n)$, where $d(n)$ is the duration of the note n, i.e. the melodic fragment that would be replaced by placing the melody $[n',n]$ at time $t-d(n')$. The notes of the theme that overlap the time interval $[t-d(n'),t+d(n)]$ are trimmed so that the extracted melody has the same duration as the candidate notes. Similarly MGD$([n'],t)$ denotes the similarity of the one-note sequence $[n']$ starting at $t-d(n')$. We call those similarities *localised* Mongeau & Sankoff similarity measures. The idea is that the similarity measure obtained by summing those localised measures over a complete sequence approximates the actual Mongeau & Sankoff similarity. This will be confirmed experimentally in the next section.

To convert the similarity measure into a weight between 0 and 1, we rescale those values to the $[0,1]$ interval, and then invert their order, so that a value of 1 is the closest to the theme, and 0 the furthest away. Finally, we exponentiate the result, so that the logarithm of the product of the biases achieved by the model is proportional to the approximated Mongeau & Sankoff similarity. Formally, we define $\beta(n|t,n')$ as follows, where MGD$_{max}$ is the maximal value of localised Mongeau & Sankoff similarities:

$$\beta(n|t,n') = \exp\left(1-\frac{\text{MGD}([n',n],t)-\text{MGD}([n'],t)}{\text{MGD}_{max}}\right)$$

3.3 Controlling the Similarity

We define an additional mechanism to control the *intensity* of the variation mechanism, i.e. the extent to which the generated melodies should be similar to the imposed theme. We introduce a parameter α, which is used to adjust the values of the biases β to new values β', defined as $\beta'(n|t,n') = \max(0,(1-\alpha).\beta(n|t,n')+\alpha)$. In theory, α ranges from $-\infty$ to 1: a very small value will cause almost all adjusted biases β' to be equal to 0, except when β was very close to 1, leading to melodies highly similar to the theme. Conversely, when α is 1, all adjusted biases β' are equal to 1, and have no effect. The interesting, non-trivial, behaviour is obtained with in-between values, which can be chosen by the user of the variation mechanism. However, the range of values where the non-trivial behaviour is observed depends on a particular corpus and a given theme. This means that a specific value of α has no general semantics, which hinders usability. As a result, we calibrate the range of α, by estimating the values for which the non-trivial behaviour occurs, given a specific corpus and theme. We estimate the values α^- and α^+ such that the average value of all adjusted biases β' is a given value close to 0 or close to 1, respectively. We estimate those values with a simple binary search. Given those two values, the user of the system then sets a parameter $\sigma \in [0,1]$, the *strictness* of the variation mechanism, and the actual value α is deduced by setting $\alpha = \sigma(\alpha^+ - \alpha^-) + \alpha^-$. We evaluate the effect of σ in practice in the next section.

4. EXPERIMENTAL RESULTS

Our approach relies on the intuition that *local* similarities, favoured by the biased model, will result in a *global* similarity between the generated melodies and the theme. In this section, we evaluate how the choice of the value for the parameter σ influences the Mongeau & Sankoff similarity between the generated melodies and the original theme. In particular, we show that the biased model favours sequences closer to the theme and penalises sequences less similar to the theme. We then explain the result more analytically, for $\sigma = 0$. We first show that applying the bias to the model approximates the localised Mongeau & Sankoff similarity, and then we show that this localised Mongeau & Sankoff similarity is a good approximation of the actual, global Mongeau & Sankoff similarity.

In the experiments below, the theme is the melody in the first four bars of *"Solar"* (Miles Davis, Figure 3). The training corpus contains 29 lead sheets by Miles Davis. In each experimental setup, we build a general model of 4-bar lead sheets in the style of Miles Davis, called the *unbiased model*, and then, we bias the model to favour the theme with some value for σ. Actual examples of variations at various distances are shown in Section 5.1.

4.1 Correlation between the Biases and the Mongeau & Sankoff Distance

For one value of σ, we generate 10 000 variations of the original theme (first four bars of *"Solar"*). For each sequence, we compute its probability p_o in the unbiased model and its probability p_b in the biased model, and then consider the ratio p_b/p_o. This probability ratio shows by how much the sequence has been favoured, for values greater than 1, or conversely penalised, for values less than 1, in the biased model. On Figure 4, points in blue are sequences generated with the most biased model, i.e.$\sigma = 0$. For each sequence, we plot its probability ratio, on a log scale, against its Mongeau & Sankoff similarity with the theme. We observe that the logarithm of the probability ratio tends to decrease linearly as the Mongeau & Sankoff distance with the theme increases. Sequences at a distance less than 75 from the theme are boosted while sequences at a distance more than 75 from the theme are hindered. Points in black are sequences generated with $\sigma = 0.95$, i.e. almost no bias at all. We observe that most sequences have a probability ratio of 1, i.e. that the biased model hardly affects the probability of sequences. Only sequences very far from the theme have their probability slightly decreased. Points in the red are generated with $\sigma = 0.5$. They display an intermediate behaviour as expected.

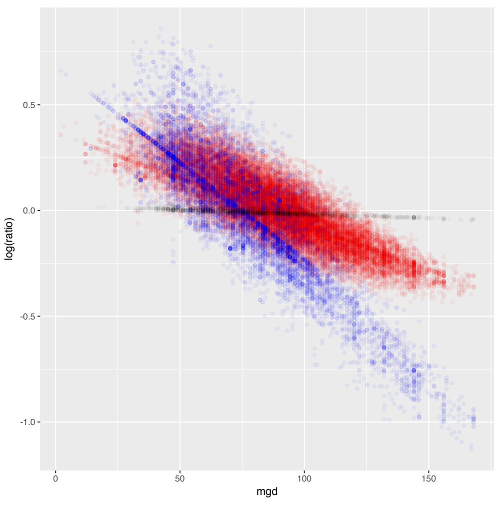


Figure 4. Sequence probability ratio (log) against Mongeau & Sankoff similarity to theme. Sequences in blue, red and black have been generated with $\sigma = 0, \sigma = 0.5, \sigma = 0.95$, respectively.

4.2 Explaining the Correlation

We explain the correlation observed by the application of two successive approximations. We concentrate on the case where $\sigma = 0$, but similar results are obtained with other values. We can break our analysis in three steps.

First, we note that for a given sequence, its probability ratio is equal, by definition of the biased model, to the product of all the local biases applied to each element of the sequence, up to a normalisation factor. We verified this experimentally too: for each generated sequence, we computed the local bias of each of the elements of the sequence, and computed the product of those local biases. We observed that this product is perfectly correlated with the ratio of probabilities of the sequence. Second, we show how the probability ratio compares with the approximated Mongeau & Sankoff similarity measure obtained by summing the localised Mongeau & Sankoff similarity measures. For each sequence, we sum, over all its elements, the localised Mongeau & Sankoff that was used when computing the biases, as explained in Section 3.2. Then, we compare this sum to the product of the local biases, equal to the probability ratio. We plot the result on Figure 5. We observe that the approximated Mongeau & Sankoff similarity measure is tightly correlated with the logarithm of the product of the local biases, i.e., the logarithm of the product of the local biases approximates closely enough the sum of the localised Mongeau & Sankoff distances.

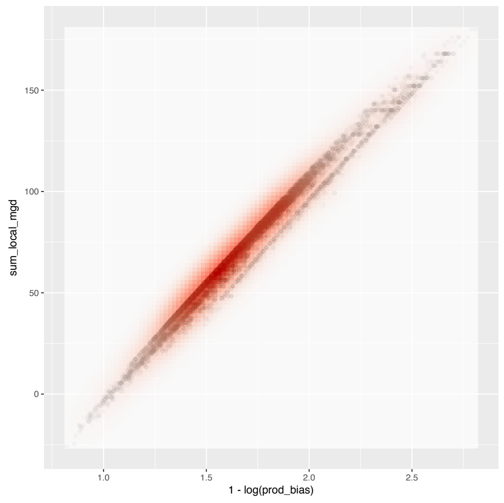


Figure 5. The sum of localised Mongeau & Sankoff similarity measures against the product of local biases (log), for $\sigma = 0$

Finally, we show that this approximated Mongeau & Sankoff similarity measure approximates the actual Mongeau & Sankoff similarity measure. On Figure 6, we plot for each sequence, the approximate versus the actual similarity measure. We observe that, although the actual measure is a global, dynamic programming-based measure, it is adequately approximated by summing the localised versions. This is probably because the localised measure captures sufficiently the effect of a note on the global similarity measure.

5. GENERATING STRUCTURED LEAD SHEETS

We show examples of melodic variations produced with our techniques, to give a concrete illustration of the variation mechanism. Then, we use the variation mechanism as the key building block to generate structured lead sheets [2].

5.1 Melodic Variations

Figure 7 shows six melodic variations of the first four bars of *"Solar"*, by Miles Davis. These variations were created

[2] All examples are available on `http://www.flow-machines.com/ismir-examples/`

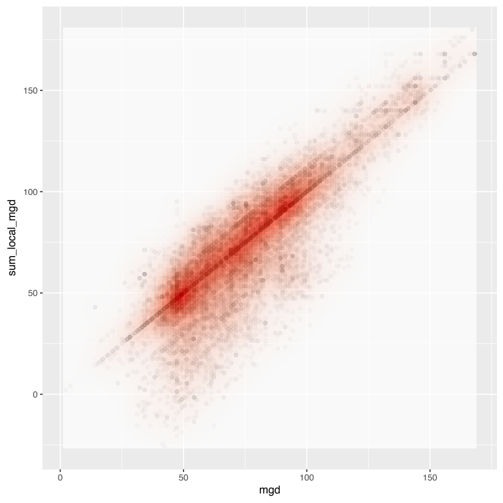
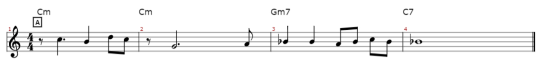
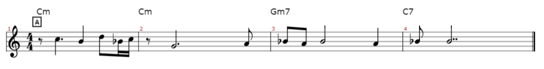


Figure 6. The sum of localised Mongeau & Sankoff similarity measures against the actual Mongeau & Sankoff measure, for $\sigma = 0$

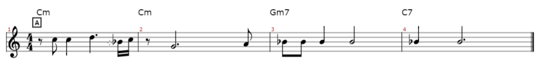
using a model trained on 29 songs by Miles Davis (Section 4). The variations are presented in increasing order of Mongeau & Sankoff distance with the original theme (Figure 3). Note that the variations are increasingly different from the theme, both rhythmically and melodically.



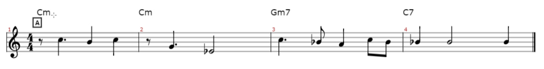
(a) Mongeau & Sankoff distance 12: highly similar to the theme



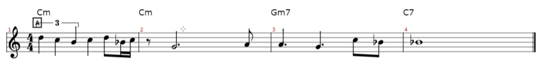
(b) Distance 86, minor enrichments in bars 1 and 3



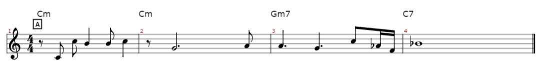
(c) Distance 87, minor enrichments in bars 1 and 3



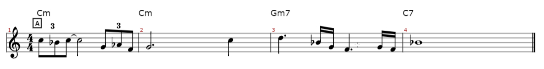
(d) Distance 224, with major differences in bars 2 and 3



(e) Distance 285, interesting triplet rhythm in bar 1



(f) Dist. 295, large initial interval (octave) and end of bar 3 differs from other variations



(g) Dist. 906, first bar uses a rhythm similar that of *"Miles Ahead"* (Miles Davis), and bar 3 is introduces a new rhythm, similar to that of the original theme, except with dotted quarter notes

Figure 7. Several variations of the first four bars of *"Solar"*, by increasing Mongeau & Sankoff distance.

5.2 Enforcing Structure

We describe our strategy for automatic composition of structured lead sheets. We use the structure of *"In a Sentimental Mood"* (Duke Ellington, Figure 8). This song has a classical AABA 32-bar structure preceded by a pickup bar:

- Sections: **Pickup**: bar 1; **A1**: bars 2 to 9, **A2**: bars 2 to 8 and bar 10, **B**: bars 11 to 18; **A3**: bars 19 to 26.
- Bar 12 is a *transposed variation* of bar 11;
- Bars 15-16 are *exact copies* of bars 11-12;
- The last bar 26 is a *variation* of bar 10, the ending of Section **A2**.

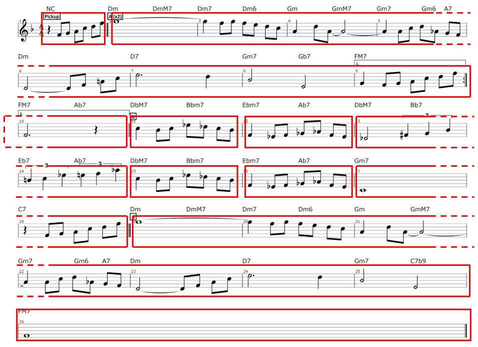


Figure 8. *"In a Sentimental Mood"* by Duke Ellington. Red boxes correspond to the basic blocks induced by the structure of the piece.

We illustrate our approach with an automatically generated lead sheet that conforms to this structure. This structure induces a segmentation of the lead sheet into contiguous blocks of music. We transform the description of the structure into a procedure that executes it. The first occurrence of each block is generated using the general model of lead sheets. Subsequent occurrences, if any, are copied from the first occurrence. If specified by the structure description, we use the variation mechanism to obtain a variation instead of an exact copy, with a strictness that may be specified by the structure description.

Each block may appear in several places, but has been generated only once, without taking into account all possible contexts. This may have the adverse effect of creating awkward transitions that the model would not have created. In these situations, we systematically apply the variation mechanism to ensure seamless transitions between blocks. Since these variations are not specified by the structure, we impose a very strict variation to ensure minimal differences with the structure description.

The chords are generated by the general model of lead sheets, either before the melody or after. In fact, there is often structure in the chord sequence too. For example, bars 4-5 of *"In a Sentimental Mood"* are a transposition of bars 2-3. We can apply the same approach, with a different notion of distance on chords.

Figure 9 shows a lead sheet with this structure and generated from a stylistic model of the Beatles (trained from a corpus with 201 lead sheets by the Beatles). The music does not sound similar to *"In a Sentimental Mood"* at all, but its structure, with multiple occurrences of similar patterns, make it feel like it was composed with some intentions. This is never the case of structureless 32-bar songs composed from the general model. Each part of the lead sheet has a strong internal coherence. The melody in the **A** parts use mostly small steps and fast sixteenth notes, many occurrence of a rhythmic pattern combining a sixteenth note with a dotted eighth note. The **B** part uses many leaps (thirds, fourth and fifth) and a regular eighth note rhythm. This internal coherence is a product of the imposed structure. For instance, in the **B** part, four out of eight bars come from a single original cell, consisting of bar 11. The fact that the **A** and **B** parts contrast with one another is also a nice feature of this lead sheet. This contrast simply results from the default behaviour of the general model of lead sheets.

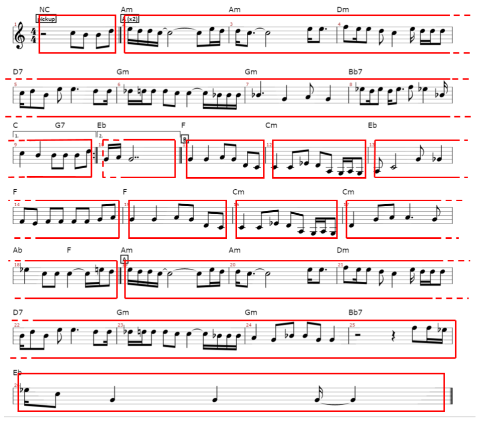


Figure 9. A lead sheet with the structure of *"In a Sentimental Mood"* but in the style of the Beatles. Note that bar 12 is a transposed variation of bar 11, as in the original song. The ending is also a variation of the ending of Section **A1**.

6. CONCLUSION

We have presented a model for sampling variations of melodies from a graphical model. This model is based on the melodic similarity measure proposed by [12]. Technically, we use an approximated version of the Mongeau & Sankoff similarity measure to bias a more general model for the generation of music. Experimental evaluation shows that this approximation allows us to bias the model towards the generation of melodies that are similar to the imposed theme. Moreover, the intensity of the bias may be adjusted to control the similarity between the theme and the variations. This makes this approach a powerful tool for the creation of pieces complying with an imposed musical structure. We have illustrated our method with the generation of a long structured lead sheet. A pop music album is currently being produced using this method.

Acknowledgments: This research is conducted within the Flow Machines project funded by the European Research Council under the EU's 7th Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 291156.

7. REFERENCES

[1] Tom Collins and Robin Laney. Computer–generated stylistic compositions with long–term repetitive and phrasal structure. *Journal of Creative Music Systems*, 1(2), 2017.

[2] David Cope. *Virtual music: computer synthesis of musical style*. MIT press, 2004.

[3] Ryan Cotterell, Nanyun Peng, and Jason Eisner. Stochastic Contextual Edit Distance and Probabilistic FSTs. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 625–630, 2014.

[4] Diana Deutsch, Trevor Henthorn, and Rachael Lapidis. Illusory Transformation from Speech to Song. *The Journal of the Acoustical Society of America*, 129(4):2245–2252, 2011.

[5] Douglas Eck and Jasmin Lapalme. Learning musical structure directly from sequences of music. *University of Montreal, Department of Computer Science, CP*, 6128, 2008.

[6] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 747–756. IEEE, 2002.

[7] G Hadjeres and F. Pachet. Deepbach: a steerable model for bach chorales generation. Technical report, arXiv:1612.01010, December 2016. https://arxiv.org/abs/1612.01010.

[8] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[9] Steven R Livingstone, Caroline Palmer, and Emery Schubert. Emotional Response to Musical Repetition. *Emotion*, 12(3):552–567, 2012.

[10] Qi Lyu, Zhiyong Wu, Jun Zhu, and Helen Meng. Modelling high-dimensional sequences with lstm-rtrbm: Application to polyphonic music generation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4138–4139. AAAI Press, 2015.

[11] Elizabeth Hellmuth Margulis. Aesthetic responses to repetition in unfamiliar music. *Empirical Studies of the Arts*, 31(1):45–57, 2013.

[12] Marcel Mongeau and David Sankoff. Comparison of Musical Sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[13] Alexandre Papadopoulos, Pierre Roy, and François Pachet. Assisted Lead Sheet Composition using Flow-Composer. In *Principles and Practice of Constraint Programming – CP 2016*. Springer, 2016.

[14] Eric Sven Ristad and Peter N Yianilos. Learning String-Edit Distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532, 1998.

[15] Elliot Waite. Generating Long-Term Structure in Songs and Stories. `https://magenta.tensorflow.org/blog/2016/07/15/lookback-rnn-attention-rnn/`, 2016.

QUANTIZED MELODIC CONTOURS IN INDIAN ART MUSIC PERCEPTION: APPLICATION TO TRANSCRIPTION

Ranjani, H. G.
Dept of ECE,
Indian Institute of Science,
Bangalore
ranjani@iisc.ac.in

Deepak Paramashivan
Dept of Music,
University of Alberta,
Canada
paramash@ualberta.ca

Thippur V. Sreenivas
Dept of ECE,
Indian Institute of Science,
Bangalore
tvsree@iisc.ac.in

ABSTRACT

Rāgas in Indian Art Music have a florid dynamism associated with them. Owing to their inherent structural intricacies, the endeavor of mapping melodic contours to musical notation becomes cumbersome. We explore the potential of mapping, through quantization of melodic contours and listening test of synthesized music, to capture the nuances of *rāgas*. We address both Hindustani and Carnatic music forms of Indian Art Music. Two quantization schemes are examined using stochastic models of melodic pitch. We attempt to quantify the salience of *rāga* perception from reconstructed melodic contours. Perception experiments verify that much of the *rāga* nuances inclusive of the *gamaka* (subtle ornamentation) structures can be retained by sampling and quantizing critical points of melodic contours. Further, we show application of this result to automatically transcribe melody of Indian Art Music.

1. INTRODUCTION

Melody contours are often perceived as continuous functions though generated from notes which assume discrete pitch values. The rendition of a *rāga*, the melodic framework of Indian Art Music (IAM), is a florid movement across notes, embellished with suitable ornamentations (*gamakas*). Several engineering approaches to analyse and/or model pitch contours rely on 'stable' notes [5, 12]; yet, it contradicts the perceptions and claims of musicians in both Carnatic and Hindustani forms of music and also that of detailed experiments which assert that it is actually the manner of approaching notes that characterizes a *rāga* [1, 6]. Algorithms to automatically align note transcription to melodic contours show promise more at a rhythm cycle level rather than at a note level [21], leading to a hypothesis that it is necessary to study the role of pitch in rendering notes rather than finding / transcribing notes from pitch contours [11]. The difficulty involved in identifying notes from a rendered melodic contour can be seen in Figure 1.

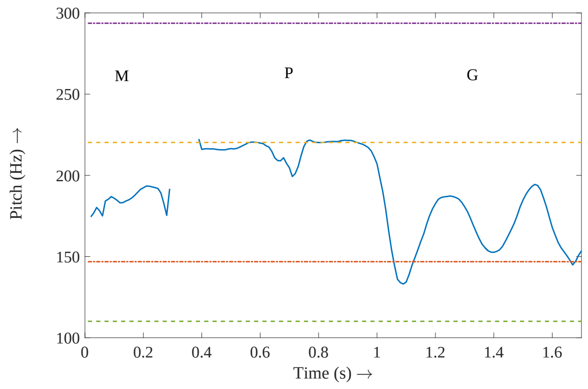


Figure 1. [Color Online] Contour of a vocal melodic clip (rendition by Vid. Deepak Paramashivan) in *Thodi rāga* of Carnatic music form (tonic frequency at 146.83 Hz). The transcribed notes correspond to '$MaPaGa$'. The presence of ornamentations pose difficulty for transcription.

In this work, we analyze pitch contours in an attempt to understand (i) how musicians possibly assess a correctly rendered note (ii) how they approach subsequent note(s) in a *rāga*. We explore the possibility to incorporate this understanding to engineer an automated framework to represent a *rāga* in terms of note sequences. A perceptual study of the effects of two quantization schemes on *rāga* characteristics (*rāga-bhava*) is explored. For a more detailed exposition of *rāgas* in Indian Art Music, interested readers can refer to [19, 20].

1.1 Complexity of Pitch Contours in Indian Art Music

A *rāga* contains 3 structures of information : (i) Pitch positions of notes (*swarasthāna*) (ii) Ornamentation of notes (*Gamaka*) (iii) Note movement (*swarasanchāra*). All the three structures are coupled in a *rāga* rendition. The note position is embellished with *gamakas*, and is also dependent on the note transitions themselves.

In [13], different notes and their transitions are studied and classified as 'inflected intervals', 'transient notes' and 'transient inflexions', while acknowledging that musical insight and knowledge is necessary to distinguish between different transitions.

From an engineering perspective, in a *rāga*, we observe that inspite of all such note transitions detected at both finer and coarser levels, the peaks, valleys and stable regions correspond to discrete pitch values with an error factor. It is also logical that a musician perceives these points to assess if the intended note has been reached [1]. The salience of peaks, valleys and stable regions is utilized for motif spotting in Carnatic music in [10]. However, not all peaks, valley regions can correspond to notes as per conventional transcriptions [15]. As an engineering approach, we propose to discretize or quantize the pitch contours at these critical points using a semi-continuous Gaussian mixture model (SC-GMM) proposed in [18] (Section 2) and thus map continuous pitch contours to discrete note sequences. We believe such a mapping brings us closer to understanding the structures of *rāgas* in accordance with the theory that discrete elements carry the structural information of musical forms while expressions are realized in continuous variations [14].

While analysing contours w.r.t. discrete pitch values, we often encounter scenarios in which pitch values can either overshoot or undershoot the intended values as can be seen in Figure 1; this is also reported in literature [13, 15]. The following reasons can be attributed to such detours w.r.t. discrete pitch values - (i) Performers' intent to generate certain perceived effect in the listener (ii) Possible deviations during learning/ fast renditions (iii) Creative freedom and margin of error allowed in rendering a *rāga* as an art form. Any deviation which does not bring about the required perceptual effect can cause a connoisseur/musician to not appreciate the rendition in its totality.

In this work, we assume the deviations to be due to any of above reasons and hence is part of errors in quantizing pitch contours. If the quantization process has disregarded the musically intended overshooting and undershooting of pitch values, it only implies that the effect of the *rāga* is not captured completely in the quantized sample. In order to analyze the importance of limits of quantization, we reconstruct the melody from quantized sequences and conduct perception experiments on these melodies (Section 3.3). Further, we propose a framework by using these quantized notes to transcribe a contour (Section 4.1).

2. QUANTIZATION MODEL

Given pitch contours, $y(t)$ estimated from audio recordings, it is possible to identify the tonic frequency f_T as shown in [7, 18]. The pitch contours are tonic normalized and mapped to a common tonic, f_U; let $y_n(t) = y(t) * f_U/f_T$ denote pitch contours mapped to common tonic frequency [2]. This helps to analyze different renditions of same *rāga*. Let $\tau = \{\ t \mid \nabla y_n(t) = 0\ \}$ be the set of critical points and $\boldsymbol{x} = \{\ y_n(\tau)\ \}$ be the corresponding critical pitch values. The tuple $\mathbf{X} = (\boldsymbol{x}, \tau)$ are the critical points of $y_n(t)$. Mathematically, critical points can be obtained only if a function is differentiable. We estimate $\mathbf{X}$ from the zero crossings of numerical gradient of $y_n(t)$.

2.1 Semi-Continuous Gaussian Mixture Model

Consider the Semi-Continuous Gaussian Mixture Model (SC-GMM) [18] with K number of components whose means, $\mu_k \forall k \in \{1, 2, \ldots, K\}$ within an octave are fixed in accordance to the note ratios used in IAM, as shown in Table 1. The distribution of pitch values in y_n and the critical pitch values $\boldsymbol{x}$ can be modeled using SC-GMM as:

$$p(y) = \sum_{k=1}^{K} \alpha_{k,y} \mathcal{N}(y_n; \mu_k, \sigma_{k,y}) \quad (1)$$

$$p(x) = \sum_{k=1}^{K} \alpha_{k,x} \mathcal{N}(x; \mu_k, \sigma_{k,x}) \quad (2)$$

For a fixed K components, the set of parameters estimated from distribution of pitch are $\{\boldsymbol{\alpha}_y, \boldsymbol{\sigma}_y\}$ and $\{\boldsymbol{\alpha}_x, \boldsymbol{\sigma}_x\}$. $\boldsymbol{\mu}$ parameters are not estimated since they are fixed and are same in both cases.

2.2 Quantization using SC-GMM

We use the above model to quantize pitch contours. Each pitch sample of $y_n(t)$ can be quantized to a nearest component of SC-GMM which maximizes its probability:

$$k_y^*(t) = \underset{k \in \{1,2,\ldots,K\}}{\arg\max}\ \alpha_{k,y} \mathcal{N}(y_n(t); \mu_k, \sigma_{k,y}) \quad (3)$$

Similarly, every critical pitch of $\boldsymbol{x}$ can be quantized as:

$$k_x^*(\tau) = \underset{k \in \{1,2,\ldots,K\}}{\arg\max}\ \alpha_{k,x} \mathcal{N}(x(\tau); \mu_k, \sigma_{k,x}) \quad (4)$$

Thus, both y_n and $\boldsymbol{x}$ are now quantized and correspond to a sequence of notes; their temporal information (corresponding to $\{t\}$ and $\{\tau\}$) are retained.

3. SYNTHESIS FROM QUANTIZED SEQUENCE OF NOTES

To check if this mapping process captures the essence of *rāgas* and to assess the effect of quantization on *rāga* perception, we conduct perception experiments. Audio clips are synthesized for perception. In order to synthesize audio from quantized note sequences, we first synthesize melody contours, and use the same to synthesize audio clips.

3.1 Quantized Pitch Contour

The un-quantized $y_n(t)$, the quantized $k_y^*(t)$ and $k_x^*(\tau)$ are interpolated to obtain a contour sampled at F_s, the sampling frequency of the discrete-time audio signal [3]. Piecewise cubic hermite interpolating polynomial is used with

[1] This also explains the fact that music listeners do not perceive intermediate notes during note transitions which are greater than a semitone; for example, when a musician glides from Sa (tonic) to Pa (fifth) in a *rāga*, we do not perceive all the intermediatary semi-tones which the glide passes through.

[2] In this work, f_U is chosen as 146.83 Hz corresponding to $D3$ note of Western scale.

[3] $y_n(t)$ also requires interpolation as it is estimated at frame rate coarser than F_s.

$(t, y_n(t))$, $(t, \mu_{k_y^*}(t))$ and $(\tau, \mu_{k_x^*}(\tau))$ being knots for each of the interpolation. These result in 3 different pitch contours for signal synthesis. The pitch contour obtained from interpolating $(t, y_n(t))$ can be considered (for all practical purposes) as a reference pitch contour. A comparison of pitch contours obtained by interpolating (τ, k_x^*) and $(t, k_y^*(t))$ are shown in Figure 2.

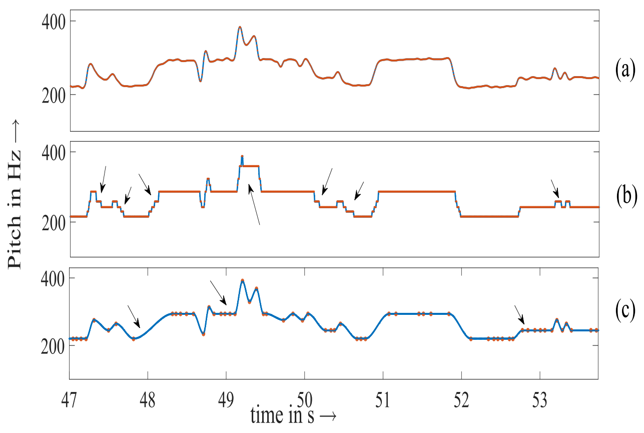


Figure 2. [Color Online] Pitch contours obtained by interpolation of (a) $y_n(t)$ (b) $k_y^*(t)$ and (c) $k_x^*(\tau)$ (*Sāveri rāga*). $f_U = 146.83\ Hz$ for all 3 contours. Red dots are knots of interpolation; solid blue lines are the interpolated contours. Arrows highlight local distortions in interpolated contours.

We see from Figure 2(b) that quantizing every pitch sample ($k_y^*(t)$) can potentially lead to introducing additional stable notes (which are not perceived in the original contour). This may result in an unfavorable shape of the contour and hence dynamics of notes may be altered during interpolation. The advantage of quantized critical pitch samples ($k_x^*(\tau)$) is that, on interpolation, the dynamics around a note are more likely to be retained as seen in Figure 2(c). Some subtle *gamakas*, in-spite of being captured, could be quantized onto single note due to statistical weight of an adjacent note, resulting in synthesizing a flat region. In this work, pitch values at critical points are perturbed while the critical points themselves are unaffected; hence slopes might be perturbed in a different manner in various sections of the pitch contours and hence perceptual effect of the same is not simple to predict.

3.2 Synthesizing Audio

The audio signal for perception can be synthesized from the interpolated pitch contours (sampled at F_s) using the time-varying sinusoidal synthesis model. The model can be expressed as:

$$\hat{s}_f(t) = a(t) * \left(\sum_{i=1}^{H} sin\left(\frac{2\pi h}{F_s} \int_0^t f(t)dt \right) \right) \quad (5)$$

where $a(t)$ represents the vocal-tract shaping filter, $*$ is the convolution operation, H denotes the number of harmonics, F_s is the sampling frequency, $f(t)$ represents the pitch contour which is to be synthesized (explained in Section 3.1) and $\int_0^t$ is approximated as cumulative sum for discrete implementation. The vocal-tract shaping filter is chosen to be time-invariant and is that of vowel /ā/. An all-pole model is used to synthesize the transfer function using formant frequencies and bandwidth of /ā/ as $(730, 1090, 2440, 3781, 4200)\ Hz$ and $(60, 50, 102, 309.34, 368)\ Hz$ respectively [17]. A drone signal is added to the synthesized audio so that reference tonic is present in it.

3.3 Perception Test Experiments

Let $\hat{s}_{ref}(t)$ be the audio signal synthesized from interpolated $(t, y_n(t))$, $\hat{s}_y(t)$ be the audio synthesized from interpolated $(t, k_y^*(t))$ and $\hat{s}_x(t)$ synthesized from interpolated $(\tau, k_x^*(t))$. We quantize using both 22-note and 12-note intervals to study the effect of number of quantization levels on *rāga* perception i.e., $\hat{s}_{y_{22}}(t)$ and $\hat{s}_{x_{22}}(t)$ are the audio signals synthesized using (3) and (4) with $K = 22 * 3$ (covering 3 octaves), while $\hat{s}_{y_{12}}(t)$ and $\hat{s}_{x_{12}}(t)$ correspond to $K = 12 * 3$ levels. The means within an octave of the SC-GMM are as chosen according to Table 1.

We choose certain *rāgas* along with the corresponding pitch features from the publicly available Carnatic and Hindustani music database used in [8, 9]; in this database, pitch has been estimated every $4.44\ ms$ using Essentia [3].

3.3.1 *Comparison of* $\hat{s}_y(t)$ *and* $\hat{s}_x(t)$

As argued earlier, we hypothesize $\hat{s}_x(t)$ to be a closer representative of $\hat{s}_{ref}(t)$ than $\hat{s}_y(t)$. To verify which among $\hat{s}_x(t)$ is indeed perceptually closer to $\hat{s}_{ref}(t)$, a MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor) kind of experiment is performed. We select $K = 12 * 3$ for quantization levels. 6 musically trained listeners are tasked with three experiments - different reference clips (average 7 s duration) from *Nattai rāga* of Carnatic music are presented in each experiment. Within each experiment - (i) $\hat{s}_{y_{12}}(t)$ (ii) $\hat{s}_{x_{12}}(t)$ (iii) hidden $\hat{s}_{ref}(t)$ - form 3 audio stimuli presented to listeners in randomized order along with an explicit reference clip $\hat{s}_{ref}(t)$. Listeners are asked to rate the closeness of each to the reference signal on a scale of 1-100 (100 implies the stimuli is indistinguishable from the reference). We refer to this as perception test 1 (PT-1).

From the results, $\hat{s}_{y_{12}}(t)$ is consistently rated least by all the listeners. These audio clips are perceived to be 'electronic', with temporal distortion clearly heard. Listeners have rated $\hat{s}_{x_{12}}(t)$ at an average of 88.88% close to explicit reference, while the hidden reference $\hat{s}_{ref}(t)$ is rated at an average of 96.5% closeness to explicit $\hat{s}_{ref}(t)$; this is because $\hat{s}_{x_{12}}(t)$ is confused with the hidden $\hat{s}_{ref}(t)$ in 38.8% cases by the subjects. $\hat{s}_{y_{12}}(t)$ is rated at an average 56.27% close to explicit reference. The bane of synthesizing melody with $k_y^*(t)$ sequence is easily perceivable by all trained listeners. This validates our hypothesis that $\hat{s}_x(t)$ is closer to $\hat{s}_{ref}(t)$ than $\hat{s}_y(t)$.

3.4 *Rāga* Perception Experiment

With $\hat{s}_x(t)$ being a close model of $\hat{s}_{ref}(t)$, we hypothesize much of *gamaka* structures in a *rāga* rendition is retained; also, micro-tonal dynamics of *rāga* might be better captured with $K = 22 * 3$ levels than $K = 12 * 3$.

Note name	22 Notes (Position)	Pitch Ratio	12 Notes(Position)
Sa	S (1)	**1**	S(1)
Ri	R_{11} (2)	256/243	
	R_{12} (3)	**16/15**	R_1(2)
	R_{21} (4)	10/9	
	R_{22} (5)	**9/8**	R_2/G_1(3)
Ga	G_{11} (6)	32/27	
	G_{12} (7)	**6/5**	R_3/G_2(4)
	G_{21} (8)	**5/4**	G_3(5)
	G_{22} (9)	81/64	
Ma	M_{11} (10)	**4/3**	M_1(6)
	M_{12} (11)	27/20	
	M_{21} (12)	**45/32**	M_2(7)
	M_{22} (13)	729/512	
Pa	P (14)	**3/2**	P(8)
Da	D_{11} (15)	128/81	
	D_{12} (16)	**8/5**	D_1(9)
	D_{21} (17)	5/3	
	D_{22} (18)	**27/16**	D_2/N_1(10)
Ni	N_{11} (19)	16/9	
	N_{12} (20)	**9/5**	D_3/N_2(11)
	N_{21} (21)	**15/8**	N_3(12)
	N_{22} (22)	243/128	
Sa'	S(next octave)(23)	**2/1**	S (upper octave)(13)

Table 1. Pitch ratios in an octave for 22-note system of Indian Art Music. The ratios used in 12-note system are in bold face.

3.4.1 Experimental Setup

We choose some *rāgas* (shown in Table 2) which are considered by experts to be musically challenging to render as they contain lot of *gamakas* and micro-tonal structures. For each listener, two different renditions (by different singers) are presented for every *rāga*. The singer identity is masked as a result of time-invariant /ā/, the shaping filter for the pitch contour; hence any bias factor due to singer in the listening experiments is reduced.

	Carnatic music	Hindustani music
1.	Begada	Bhairav
2.	Bhairavi	Darbari
3.	Saveri	Marwa
4.	Sahana	Puriya Dhanashree
5.	Sindhu Bhairavi	Yaman
6.	Thodi	

Table 2. *Rāgas* chosen for perception experiment.

To verify if $\hat{s}_x(t)$ captures the *rāga* nuances along with the *gamakas* in its entirety as represented in $\hat{s}_{ref}(t)$, in each experiment, we present a 1 min duration clip of $\hat{s}_{ref}(t)$ and its corresponding (i) $\hat{s}_{x_{22}}(t)$ and (ii) $\hat{s}_{x_{12}}(t)$ (synthesized) audio clips [4] .

We first present to music experts, $\hat{s}_{ref}(t)$ as the reference and ask them to rate on a scale of 1-10 for *rāga* characteristics present in $\hat{s}_{ref}(t)$. The same listener is now presented with $\hat{s}_{x_{22}}(t)$ and $\hat{s}_{x_{12}}(t)$ (not necessarily in that order) and asked to rate closeness of each with respect to *rāga* nuances of $\hat{s}_{ref}(t)$ on the scale of 1-10. Lower rating implies *rāga* nuances are compromised due to quantization. Thus, each listener for Hindustani music form participates in 10 (5 *rāgas* with 2 different renditions) such experiments; and, 12 experiments are presented for each Carnatic expert listener. This is perception test 2 (PT-2).

3.4.2 PT-2 Results and Analysis

5 performing Carnatic musicians were selected for the perception test in Carnatic music; similarly, 5 musicians trained in Hindustani music were considered for the Hindustani music perception tests.

The average ratings of $\hat{s}_{x_{12}}(t)$ and $\hat{s}_{x_{22}}(t)$ w.r.t. reference $\hat{s}_y(t)$ for each *rāga* considered in Carnatic and Hindustani music is as shown in Figure 3 (a) and (b) respectively.

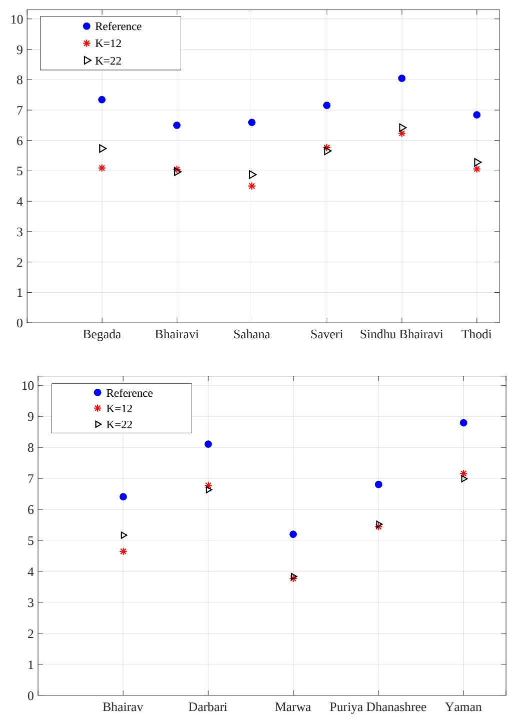


Figure 3. [Color Online] Perception rating for *rāga* characteristics for $\hat{s}_y(t)$, $\hat{s}_{x_{12}}(t)$ and $\hat{s}_{x_{22}}(t)$ for (a) Carnatic music averaged over 5 listeners (b) Hindustani music averaged over 5 listeners.

The $\hat{s}_{ref}(t)$ ratings absorbs anomalies such as sudden breaks and octave errors which commonly occur in pitch

[4] The clip is a part of the starting portion of the original rendition but between the region 30 s to 90 s. While the *rāga* characterizing phrases will be brought about initially, we hypothesize that *rāga* nuances must be showcased at any chunk of time.

estimation algorithms, as well as errors committed by the artist in the rendition. In both Carnatic and Hindustani music clips, $\hat{s}_{x_{12}}(t)$ and $\hat{s}_{x_{22}}(t)$ are reported to be quite close to $\hat{s}_{ref}(t)$ and requires more careful inspection by repeated contrasts against $\hat{s}_{ref}(t)$ as described ahead.

In order to pin-point distortions perceived, expert listeners had to intently re-listen the $\hat{s}_{ref}(t)$ multiple times to confirm if perceived foreign/distorted notes are present in $\hat{s}_{x_{12}}(t)$ or $\hat{s}_{x_{22}}(t)$ clips and not in $\hat{s}_{ref}(t)$ clip itself. Some of the commonly observed distortions are: *gamakas* being flattened, a foreign note perceived in $\hat{s}_{ref}(t)$ clip being corrected and note shift at micro-tonal level. A Hindustani music performer after multiple listenings, could report $\sim$ 13 perceivable distortions (in each $\hat{s}_{x_{12}}(t)$ and $\hat{s}_{x_{22}}(t)$) with respect to total 10 reference clips.

In the perception experiment in Hindustani music, at $K = 12 * 3$ and $K = 22 * 3$ quantization levels, given $\hat{s}_{ref}(t)$, expert listeners have given equal rating to both $\hat{s}_{x_{12}}(t)$ and $\hat{s}_{x_{22}}(t)$ for 42% of the test cases; and, for $\sim 32\%$ of the cases, the listeners have rated $\hat{s}_{x_{22}}(t)$ to be closer to reference than $\hat{s}_{x_{12}}(t)$. Also, the overall average rating for $\hat{s}_{x_{12}}(t)$ is very close to $\hat{s}_{x_{22}}(t)$ for most of the listeners. This is perhaps due to the inherent predisposition of performers of Hindustani music to elaborate individual notes, thereby minimizing the transitions between the 22 note positions.

Similar analysis on perception of Carnatic music shows that in 36% cases, ratings for $\hat{s}_{x_{12}}(t)$ and $\hat{s}_{x_{22}}(t)$ were the same. In $\sim 40\%$ cases, $\hat{s}_{x_{22}}(t)$ was found to be closer to reference than $\hat{s}_{x_{12}}(t)$.

In order to obtain measure of overall inter-listener agreement, we first categorize the ratings in each experiment into 3 categories - (i)$\hat{s}_{x_{22}}(t)$ closer to $\hat{s}_{ref}(t)$ (ii) $\hat{s}_{x_{12}}(t)$ closer to $\hat{s}_{ref}(t)$ (iii) equal ratings to both $\hat{s}_{x_{22}}(t)$ and $\hat{s}_{x_{12}}(t)$. For an i^{th} experiment, the agreement among the L number of listeners can be defined as [4]:

$$P_i = \frac{1}{L(L-1)} \sum_{j=1}^{3} n_{ij}^2 - L \tag{6}$$

where, n_{ij} is the number of raters who have assigned j^{th} category in i^{th} experiment.

In PT-2 perception test of Hindustani music, the average inter-listener agreement per experiment is found to be 0.37 while for Carnatic music, the average is 0.34.

From PT-1, we could infer that quantization at every pitch sample results in perceivable loss of *rāga* structure. The results of PT-2 shows that it is possible to quantize at critical points while retaining the *rāga* structure. There could be a few note omissions and distortions at micro-tonal levels which are not perceivable in one listening, implying *rāga* structure is well retained. This also implies that **quantizing critical pitch values keeps much of the *gamaka* structure** (which has been indefinable till now) **intact**. Expert musicians show sensitivity to 22-note positions; in some clips, musicians appreciate the approach to a note as interpolated by $\hat{s}_{x_{22}}(t)$ more than $\hat{s}_{ref}(t)$ [5]. We infer that both $K = 12 * 3$ and $K = 22 * 3$, depict close scores and retain *rāga* structure well.

[5] Sometimes, $\hat{s}_{x_{12}}(t)$ is also reported to interpolate transitions better than $\hat{s}_{ref}(t)$

3.5 Relation to Waveform Quantizers

The model corresponding to Equation (3) is a waveform quantizer. While an uniform quantizer assumes $y_n(t)$ to have uniform distribution, the model corresponding to Equation (1) and (3) is a non-uniform, parameterized, stochastic waveform quantizer. The stochastic SC-GMM incorporates shape of the pdf through its parameters to derive rendition-specific and/or *rāga*-specific quantization thresholds. While a well-designed optimum waveform quantizer with sufficient bit-depth can result in hi-fidelity audio, we have shown, from results of perception experiment PT-1, that non-uniform, parameterized pitch 'waveform' quantization unsettles the *rāga-bhava* even within a small 7 s melodic phrase. Increasing bit-depth without correlating to essential pitch-ratios (within an octave) will be of limited utility.

From model defined by Equation (2) and (4), we have seen from results of PT-2 that sub-sampling (at critical points) and then using a non-uniform, parameterized, stochastic quantizer results in melodic contours which can reconstruct *rāga-bhava* with less distortions. Increasing bit-depth (from $K = 12 * 3$ to $K = 22 * 3$) need not always result in lesser 'perceptual' distortions in melody signals which are inherently structured.

4. APPLICATIONS OF QUANTIZED PITCH CONTOUR

4.1 Note Transcription

A direct application of discretizing melody contours is in note transcription. While attempts have been made to capture regions corresponding to discrete notes, we now theorize that discrete notes can occur as points and/or regions in the melodic-temporal domain; elongated notes result in regions, while other-wise they can be essentially considered as points.

4.1.1 Experimental Setup

We have recorded a total of close to 50 phrases each in Hindustani and Carnatic music forms; the phrases are spread across 5 *rāgas* (as listed in Table 3) and is a modest database to quantify accuracy of note transcription. These phrases contain *rāga* specific *gamakas* such that their conventional transcription differs from their renditions. The Hindustani database is rendered with *Sārangi* instrument, while the Carnatic database contains vocally rendered phrases. Each phrase is associated with 2 note sequences - (i) note sequences as transcribed by musicians (referred as TA transcription) (ii) note sequences as musicians render it with the associated *gamakas*, but now explicitly notated (referred to as TB transcription). Figure 4 is a sample depicting the differences between TA and TB. The transcription notation used here consider only the note sequences and do not include duration information.

Pitch is extracted using Praat [2] every 8 ms. A SC-GMM model is built for each *rāga* by combining all phrases; note sequences are obtained as per Equation 4.

4.1.2 Results and Analysis of transcribed sequences

The performance of automatic note transcription task is measured using TA and TB transcriptions as ground truths. The automatically obtained note sequence is aligned with TA (or TB) using Needleman Wunsch global alignment algorithm [16] with gap penalty set to zero. Performance is reported in terms of recall accuracy and insertion rate. Table 3 summarizes the performance of automatic transcription using $k^*_{y_{12}}$ and $k^*_{x_{12}}$ note sequences in both Carnatic and Hindustani music.

Transcription using $k^*_{y_{12}}$ sequences always shows high recall results (as expected) and also results in high insertion rate; as every pitch sample is quantized, there is less likelihood of missing any note but more chances of false alarms.

With TA transcription as ground truth, recall rates using $k^*_{x_{12}}$ sequences is comparable to that using $k^*_{y_{12}}$ in Hindustani music; for Carnatic music, recall rate performance of $k^*_{x_{12}}$ sequence is seen to have decreased. Due to dynamic nature of Carnatic music, some notes in TA are not representative of the rendition. For example in *rāga Thodi*, though TA contains note Ga, it is rendered as $Ma - Ri$ (cf. Figure 1). Also, frequent notes or stable notes have dominant presence as Gaussian component (reflected as $\boldsymbol{\alpha}$); any critical pitch value in the vicinity of a dominant note can be assigned a higher probability in-spite of its distance to another adjacent but not-frequent note. This can cause incorrect pitch-to-note mapping.

Drastic reduction in transcription insertion rate can be attributed to $k^*_{x_{12}}$ sequences being estimated from sub-sampled version of $y_n(t)$. Thus, not all points are transcribed.

With TB transcription as ground truth, insertion rate is reduced for both $k^*_{y_{12}}$ and $k^*_{x_{12}}$ sequences. This is attributed to TB version of ground truth being a more elaborate explanation of a rendition. A sample depiction of the same can be seen in Figure 5. The number of false note assignment is reduced with transcription using $k^*_{x_{12}}$ as against $k^*_{y_{12}}$.

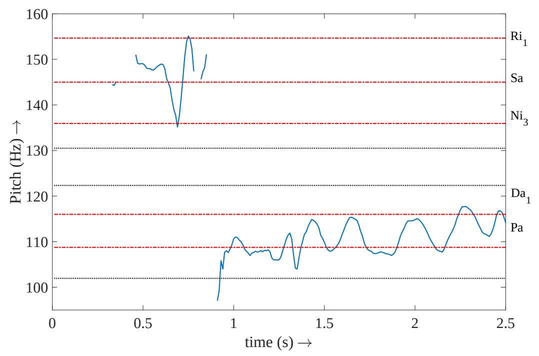


Figure 4. [Color Online] The blue contour corresponds to a phrase of *rāga Keervani* (Carnatic). Red lines indicate the pitch values of notes used in the *rāga*, while black lines denote pitch of notes that are not used. This phrase is transcribed as '$SaNiPaDa$' (TA). Considering the *gamakas* involved, it is rendered as '$SaNiRiSaPaDaPaDaPaDaPaDa$' (TB).

```
a)   ------------SN-----RS----------PDPDPD----PD
                 | |      | |             | | | | | |      | |
b)   SRSNSGRSNSRSRSMPDPDPDPDPDPMPDS


c)   ---SNR---SPDPDPDPD--
        | | |    | | | | | | | |
d)   SNSNRPDPDPDPDPDM
```

Figure 5. Melodic notes for pitch contour of Figure 4, corresponding to (a) Ground truth, TB (b) Transcription obtained using $k^*_{y_{12}}$ (c) Ground truth, TB (d) Transcription obtained using $k^*_{x_{12}}$

	(a) Hindustani							
	TA				TB			
	$k^*_{y_{12}}(t)$		$k^*_{x_{12}}(t)$		$k^*_{y_{12}}(t)$		$k^*_{x_{12}}(t)$	
Rāga	Rec	Ins	Rec	Ins	Rec	Ins	Rec	Ins
Bihag	1	3.65	1	1.55	0.93	1.39	0.88	0.39
Goud Sarang	1	4	1	2	0.88	1.97	0.83	0.88
Keervani	0.97	4.59	0.95	2.06	0.88	1.69	0.8	0.6
Madhuvanti	1	3.93	0.96	1.77	0.98	1.97	0.96	0.67
Marwa	1	7.18	0.97	3.36	0.96	2.4	0.88	0.90

	(b) Carnatic							
	TA				TB			
	$k^*_{y_{12}}(t)$		$k^*_{x_{12}}(\tau)$		$k^*_{y_{12}}(t)$		$k^*_{x_{12}}(\tau)$	
Rāga	Rec	Ins	Rec	Ins	Rec	Ins	Rec	Ins
Begada	1	8.58	0.85	1.91	0.98	2.93	0.80	0.32
Bhairavi	1	7.75	0.92	2.17	0.81	2.32	0.54	0.57
Hamsadvani	1	7.15	0.87	2	0.97	2.61	0.82	0.45
Hindola	1	10.8	1	3	0.9	3.03	0.86	0.46
Keervani	0.98	7.27	0.90	2.45	0.81	2.25	0.70	0.54
Thodi	1	10	0.88	2.55	0.92	3.04	0.88	0.36

Table 3. Performance of $k^*_{y_{12}}$ and $k^*_{x_{12}}$ sequences for automatic transcription in terms of average recall rate (Rec) and insertion rate (Ins) w.r.t. TA and TB ground truth transcription of phrases in (a) Hindustani (b) Carnatic music.

5. CONCLUSIONS

We have explored two different quantization techniques using stochastic models for mapping continuous melody contours to discrete pitch values; perception experiments show that *rāga-bhava* can be preserved by quantizing the pitch contour at critical points instead a waveform-quantization type of approach. The stochastic, parameterized SC-GMM assimilates information in pitch pdf to derive quantization thresholds. Applying results of perception experiments to automatic transcription task results in a detailed description of a melodic piece; such a detailed transcription can inherently aid in mapping *rāga* dynamics and *gamakas* into musical notation.

6. REFERENCES

[1] A. Bellur, V. Ishwar, and H. A Murthy. Motivic analysis and its relevance to raga identification in carnatic music. In *Proc. 2nd CompMusic Workshop*, pages 153–157, Istanbul, Turkey, 2012.

[2] P. Boersma and D. Weenink. Praat: doing phonetics by computer (version 6.0.14), 2016.

[3] 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 *Proc. Int. Soc. on Music Info. Retr. Conf (ISMIR)*, pages 493–498, 2013.

[4] J.L. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[5] K. K. Ganguli, S. Gulati, P. Rao, and X. Serra. Data-driven Exploration of Melodic Structures in Hindustani Music. *Proc. Int. Soc. on Music Info. Retr. Conf (ISMIR)*, pages 605–611, 2016.

[6] K. K. Ganguli and P. Rao. Perceptual Anchor or attractor: How do Musicians perceive Raga Phrases. *Proc. Frontiers in Research in Speech and Music (FRSM)*, pages 174–178, 2016.

[7] S. Gulati, A. Bellur, J. Salamon, H. G. Ranjani, V. Ishwar, H. Murthy, and X. Serra. Automatic tonic identification in Indian art music: approaches and evaluation. *Journal of New Music Research*, 43(1):53–71, 2014.

[8] S. Gulati, J. Serrà, K. K. Ganguli, S. Şentürk, and X. Serra. Time-Delayed Melody Surfaces for rāga Recognition. In *Proc. Int. Soc. on Music Info. Retr. Conf (ISMIR)*, pages 751–757, New York (USA), 2016.

[9] S. Gulati, J. Serra, V. Ishwar, S. Sentürk, and X. Serra. Phrase-based rāga recognition using vector space modeling. In *Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pages 66–70. IEEE, 2016.

[10] V. Ishwar, S. Dutta, A. Bellur, and H. A. Murthy. Motif spotting in an alapana in carnatic music. In *Proc. Int. Soc. on Music Info. Retr. Conf (ISMIR)*, pages 499–504, 2013.

[11] G. K. Koduri. *Towards a multimodal knowledge base for Indian art music: A case study with melodic intonation*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2016.

[12] G. K. Koduri, V. Ishwar, J. Serrà, and X. Serra. Intonation analysis of rāgas in carnatic music. *Journal of New Music Research*, 43:72–93, 2014.

[13] A. Krishnaswamy. Pitch measurements versus perception of south indian classical music. In *Proc. of the Stockholm Music Acoustics Conference (SMAC)*, pages 627–630, 2003.

[14] S. McAdams. Psychological constraints on form-bearing dimensions in music. *Contemporary Music Review*, 4(1):181–198, 1989.

[15] Wim van der Meer and S. Rao. What you hear isn't what you see: The representation and cognition of fast movements in Hindustani music. In *Frontiers in Research in Speech and Music (FRSM)*, pages 1–8, Lucknow, India, 2006.

[16] 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*, 48(3):443–453, 1970.

[17] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. PTR Prentice Hall, 1993.

[18] H. G. Ranjani, S. Arthi, and T. V. Sreenivas. Carnatic music analysis: Shadja, Swara identification and Raga verification in Alapana using stochastic models. In *Proc. Workshop on Applicat. of Signal Process. to Audio and Acoust.*, pages 29–32, Oct 2011.

[19] G. Ruckert, A.A. Khan, and U.A. Khan. *The Classical Music of North India: The first years study*. Munshiram Manoharlal Publishers, 1998.

[20] P. Sambamoorthy. *South Indian Music*. Number v. 1-2 in South Indian Music. Indian Music Publishing House, 1963.

[21] S. Şentürk, G. K. Koduri, and X. Serra. A Score-Informed Computational Description of Svaras Using a Statistical Model. In *Proc. Conf. Sound and Music Computing (SMC)*, pages 427–433, Hamburg, Germany, 2016.

MODELING THE MULTISCALE STRUCTURE OF CHORD SEQUENCES USING POLYTOPIC GRAPHS

Corentin Louboutin
Université Rennes 1 / IRISA, France
corentin.louboutin@irisa.fr

Frédéric Bimbot
CNRS - UMR 6074 / IRISA, France
frederic.bimbot@irisa.fr

ABSTRACT

Chord sequences are an essential source of information in a number of MIR tasks. However, beyond the sequential nature of musical content, relations and dependencies within a music segment can be more efficiently modeled as a graph.

Polytopic Graphs have been recently introduced to model music structure so as to account for multiscale relationships between events located at metrically homologous instants.

In this paper, we focus on the description of chord sequences and we study a specific set of graph configurations, called Primer Preserving Permutations (PPP). For sequences of 16 chords, PPPs account for 6 different latent systems of relations, corresponding to 6 main structural patterns (Prototypical Carrier Sequences or PCS). Observed chord sequences can be viewed as distorted versions of these PCS and the corresponding optimal PPP is estimated by minimizing a description cost over the latent relations.

After presenting the main concepts of this approach, the article provides a detailed study of PPPs across a corpus of 727 chord sequences annotated from the RWC POP database (100 pop songs). Our results illustrate both qualitatively and quantitatively the potential of the proposed model for capturing long-term multiscale structure in musical data, which remains a challenge in computational music modeling and in Music Information Retrieval.

1. INTRODUCTION

One of the essential properties of music structure is the multiscale nature of the inner organization of musical segments, i.e. the existence of relationships between musical elements at different time-scales simultaneously.

Given its important role in supporting the local harmonic ground-plan of the music in a significant number of music genres, chord sequences are commonly considered as an essential source of information in a variety of MIR tasks (see for instance [13, 17, 22]).

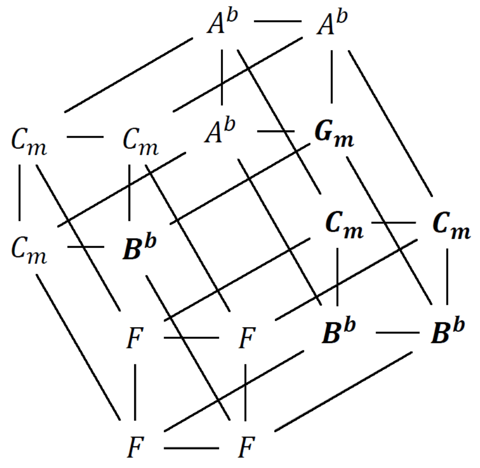


Figure 1. A chord sequence represented on a tesseract.

However, beyond the sequential order of chords along the timeline, relations and dependencies between chords within a music segment tend to be more efficiently modeled as a graph.

Polytopic Graphs of Latent Relations (PGLR) [11] have been recently introduced to model music structure, so as to account for multiscale relationships between events located at metrically homologous instants, by means of an oriented graph supported by a n-dimensional polytope.

This class of models is assumed to be particularly well suited for strongly "patterned" music, such as pop music, where recurrence and regularity tend to play a central part in the structure of the musical content.

PGLR also relax the adjacency hypothesis of the GTTM model [10], by which the grouping of elements into higher level objects is strictly limited to neighbouring units. This is particularly useful to account for period-like $abac$ patterns, where the similarity relationship between the two a segments spans above (and irrespective of) the b segment.

In this paper, we focus on the description of metric-synchronous chord sequences of 16 elements, resting on regular phrasal structures or *carrures*. In that case, the supporting polytope is a tesseract (i.e. a 4-cube) as illustrated by Fig. 1, and the graph description lives on this tesseract (as represented on Fig. 4).

After providing, in Section 2, the main concepts and formalisms related to the approach, we study in detail a particular variant of the model, where the graph structure is restricted to a set of 6 configurations, called Primer Preserving Permutations (PPP). We show in Section 3.1 that PPPs relate to prototypical multi-scale structural patterns which we call Prototypical Carrier Sequences (PCS) and we explain how observed chord sequences can be viewed as distorted versions of these prototypical patterns. In the last part of the article (Section 4), we provide an experimental study of PPPs across a corpus of 727 chord sequences annotated from the RWC POP database (100 pop songs) with qualitative and quantitative results illustrating the potential of the model. We conclude with perspectives outlined by the proposed approached.

2. CONCEPTS AND FORMALISM

2.1 The PGLR Framework

As mentioned in the introduction, the PGLR approach views a sequence of musical elements within a structural segment as exhibiting privileged relationships with other elements located at similar metrical positions across different timescales.

For metrically regular segments of 2^n events, the corresponding PGLR conveniently lives on an n-dimensional cube (square, cube, tesseract, etc...) [1], n being the number of scales considered simultaneously in the multiscale model. Each vertex in the polytope corresponds to a musical element of the lowest scale, each edge represents a latent relationship between two vertices and each face forms an elementary system of relationships between (typically) 4 elements. In addition, the proposed model views the last vertex in each elementary system as the denied realization of a (virtual) expected element, itself resulting from the implication triggered by the combination of former relationships within the system (see Section 2.3).

For a given support polytope, the estimated PGLR structure results from the joint estimation of (i) the configuration of an oriented graph resting on the polytope, with the constraint that it reflects causal time-order preserving dependencies and interactions between the elements within the musical segment, and (ii) the inference of the corresponding relations between the nodes of the graph, these relations being termed as latent, as they are not explicitly observed (and may even not be uniquely defined).

2.2 Chord Representation and Relations

Strictly speaking, a chord is defined as any harmonic set of pitches that are heard as if sounding simultaneously. However, in tonal western music, chords are more specifically understood as sets of *pitch classes* which play a strong role in the accompaniment of the melody (in particular, in pop songs).

A number of formalisms exist for describing chord relations, either in the context of classical musicology or in the framework of more recent theories, for instance, the neo-Riemannian theory and voice-leading models [5,6,20], or computational criteria such as Minimal Transport [10].

[1] and more generally speaking, on an n-polytope

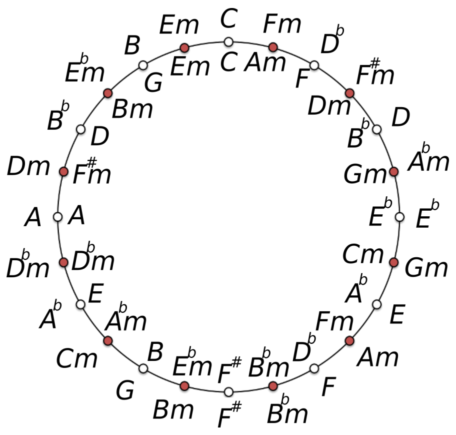


Figure 2. Circles of thirds (inner) and phase-shifts (outer).

While chords may contain combinations of four pitch classes or even more, they are frequently reduced to triads (i.e. sets of three pitch classes), with a predominance of major and minor triads. A minimal representation of triads boils down to 24 distinct triads (12 major and 12 minor). In the rest of this article, we restrict ourselves to this case, in spite of its simplified nature.

In order to model relations between triads, we consider triadic circles, i.e. circular arrangements of chords aimed at reflecting some proximity relationship between triads along their circumference.

The circle of thirds is formed by alternating major and minor triads with neighbouring triads sharing two common pitch classes, which is a way to model some kind of proximity between chords. In particular, chords belonging to a given key lie in a same sector of the circle of thirds. As an alternative, we also consider the circle of phase-shifts, which consists of a chord progression resulting from a minimal displacement on the 3-5 phase torus of triads as defined in [1]. Both circles are shown together on Fig. 2.

Each circle provides a way to express (in a unique way), the relationship between two triads, as the angular displacement along the circle. Note that a "chromatic" circle (... B_m B C_m C D^b_m D^b ...) could also be considered, but it is not represented on Fig. 2, for reasons explained later.

2.3 Systemic Organization

Based on the hypothesis that the relations between musical elements form a system of projective implications, the System & Contrast (S&C) model [2] has been recently formalized [3] as a generalization and an extension of Narmour's Implication-Realization model [16]. Its applicability to various music genres for multidimensional and multiscale music analysis has been explored in [7] and algorithmically implemented in an early version as "Minimal Transport Graphs" [10].

The S&C model primarily assumes that relations between 4 elements in a musical segment x_0 x_1 x_2 x_3 can be viewed as based on a two-scale *system* of relations rooted on the first element x_0 (the *primer*), which thus plays the role of a common *antecedent* to all other elements in the

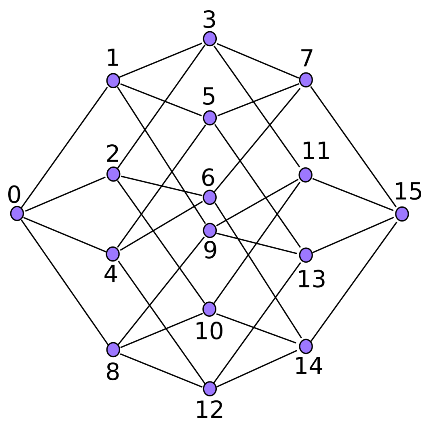


Figure 3. Tesseract where vertices at a same depth (or geodesic distance) from vertex #0 are aligned vertically. The resulting partial order between vertices is causal.

system. This is the basic principle that enables the joint modeling of several timescales simultaneously.

In the S&C approach, it is further assumed that latent systemic relations $x_1 = f(x_0)$ and $x_2 = g(x_0)$ trigger a process of projective implication:

$$x_0\ f(x_0)\ g(x_0) \overset{implies}{\Longrightarrow} g(f(x_0)) = \hat{x}_3 \quad (1)$$

The *virtual* element $\hat{x}_3$ may then be more or less strongly denied by a *contrast*: $x_3 = \gamma(\hat{x}_3) \neq \hat{x}_3$, which creates some sense of closure to the segment.

In this work, the S&C model is used as the basic scheme to describe systems of music elements forming the faces of the tesseract.

3. GRAPH-BASED DESCRIPTION

3.1 Nested Systems

Elementary systems of four elements, as introduced in Section 2.3, can be used to describe longer sequences of musical events. In particular, sequences of 2^n elements arranged on an n-cube, provide a layout of the data where each face potentially forms a S&C, involving time instants that share specific relationships in the metrical grid.

As opposed to the sequential viewpoint which assumes a total order of elements along the timeline, the systemic organization on the tesseract leads to a partial order (illustrated on Fig. 3), where elements of the same depth are aligned vertically and where, in the framework of the S&C, the fourth element of each face can be defined in reference to the virtual element resulting from the projective implication of the three others. In the most general case, valid systemic organizations can be characterized by a graph of nested systems, the flow of which respects the partial ordering of Fig. 3. Note however that there is a possible conflict between three implications systems for elements 7, 11, 13 and 14 (each possible implication corresponding to a face of the tesseract[2]), and six for element 15.

[2] for instance, node 7 can be viewed as resulting from 3 implication systems: $[1, 3, 5, 7]$, $[2, 3, 6, 7]$ and $[4, 5, 6, 7]$.

$P0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	A	A	A	A	B	B	B	B	C	C	C	C	D	D	D	D
$P1$	0	1	4	5	2	3	6	7	8	9	12	13	10	11	14	15
	A	A	B	B	A	A	B	B	C	C	D	D	C	C	D	D
$P2$	0	2	4	6	1	3	5	7	8	10	12	14	9	11	13	15
	A	B	A	B	A	B	A	B	C	D	C	D	C	D	C	D
$P3$	0	1	8	9	2	3	10	11	4	5	12	13	6	7	14	15
	A	A	B	B	C	C	D	D	A	A	B	B	C	C	D	D
$P4$	0	2	8	10	1	3	9	11	4	6	12	14	5	7	13	15
	A	B	A	B	C	D	C	D	A	B	A	B	C	D	C	D
$P5$	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D

Table 1. List of the 6 PPPs, together with their corresponding Prototypical Carrier Sequences (PCS).

3.2 Primer Preserving Permutations (PPP)

One way to handle these conflicts is to constrain the graph to preserve systemic properties at higher scales. This can be achieved by forcing lower-scale systems to be supported by parallel faces on the tesseract, while the first elements of each of the 4 lower-scale systems are used to form an upper-scale system. This approach drastically brings down the number of possible graphs to 6, which corresponds to specific permutations of the initial index sequence (see Table 1), termed here as PPP (Primer Preserving Permutations).

To illustrate a PPP, let's consider the subdivision of a sequence of 16 chords into four sub-sequences of four successive chords. Each sub-sequence can be described as a separate Lower-Scale S&C (LS): $[0, 1, 2, 3]$, $[4, 5, 6, 7]$, $[8, 9, 10, 11]$ and $[12, 13, 14, 15]$. Then, these four S&Cs can be related to one another by forming the Upper-Scale S&C (US) $[0, 4, 8, 12]$, linking the four primers of the 4 LS. This configuration ($P0$) turns out to be particularly economical for describing chord sequences such as SEQ_1:

$Cm\ Cm\ Cm\ Bb \quad Ab\ Ab\ Ab\ Gm \quad F\ F\ F\ F \quad Cm\ Cm\ Bb\ Bb$

as most similarities develop between neighbouring elements.

If we now consider the following example (SEQ_2):

$Bm\ Bm\ A\ A \quad G\ Em\ Bm\ Bm \quad Bm\ Bm\ A\ A \quad G\ Em\ Bm\ Bm$

a different configuration appears to be more efficient to explain this sequence. In fact, grouping chords into the following 4 LS: $[0, 1, 8, 9]$, $[2, 3, 10, 11]$, $[4, 5, 12, 13]$ and $[6, 7, 14, 15]$, and then relating these four faces of the tesseract by a US $[0, 2, 4, 6]$ (configuration $P3$) leads to a less complex (and therefore more economical) description of the relations between the data within the segment. Fig. 4 illustrates these two configurations.

3.3 Prototypical Carrier Sequences (PCS)

Each of the 6 PPPs can be related to a prototypical structural pattern which turns out to be the one that is the most concisely described in the framework of this particular configuration. These 6 patterns, identified in Table 1, can be interpreted as "Prototypical Carrier Sequences" (PCS) over which the actual chord sequence appears as partially "modulated" information.

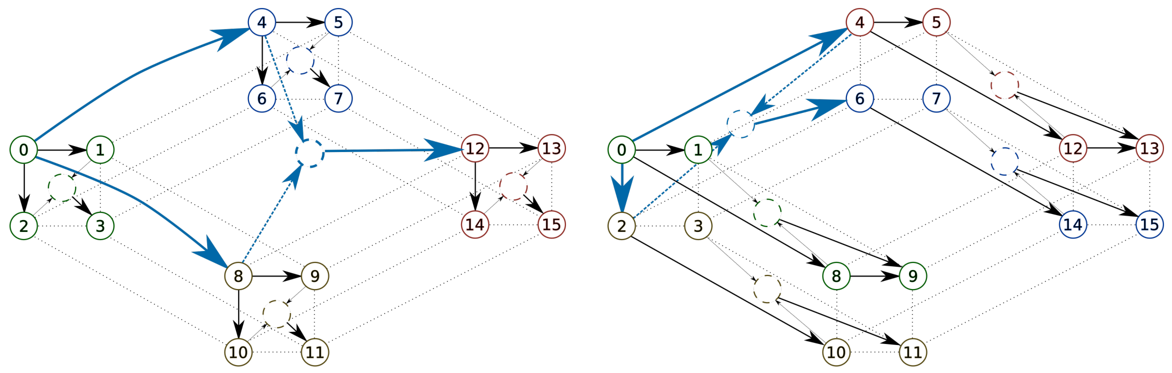


Figure 4. Representations of two PPP-based PGLRs on a tesseract: $P0$ (left), $P3$ (right). In blue, the Upper-Scale S&C – in black, the 4 Lower-Scale (LS) S&Cs. Dotted nodes indicate the virtual elements ($\hat{x}$) in the implication scheme (Section 2.3).

For instance, SEQ_1 appears merely as a sequence of type $P0$, which has been altered in positions 3, 7, 14 and 15 from the following carrier system:

$Cm\,Cm\,Cm\,Cm \quad Ab\,Ab\,Ab\,Ab \quad F\,F\,F\,F \quad Cm\,Cm\,Cm\,Cm$

Conversely, SEQ_2 exhibits a pattern that strongly relates to $P3$, with scattered deviations from:

$Bm\,Bm\,A\,A \quad G\,G\,Bm\,Bm \quad Bm\,Bm\,A\,A \quad G\,G\,Bm\,Bm$

located in positions 5 and 13.

Inferring the PCS shows interesting analogies with a demodulation operation and/or an error correcting code process, by concentrating the redundancy on the carrier sequence. It can of course happen that a sequence has several possible descriptions of equivalent plausibility, i.e. multiple coexisting interpretations w.r.t. its prototypical PPP structure.

In summary, PPP provide a limited set of baseline multiscale structural patterns which can be used to characterize actual chord sequences, via a minimum deviation criterion.

3.4 Algorithmical Considerations

In practice, given a (chord) sequence, $X = x_0 \ldots x_{l-1}$, its optimal description ($\mathcal{D}^X$) within a class of PGLRs, can be obtained by minimizing a criterion $\mathcal{F}$ written as:

$$\mathcal{D}^X = [\Psi^X, R^X] = argmin_{[\Psi,R]}\ \mathcal{F}(\Psi, R|X) \qquad (2)$$

where Ψ is a PGLR and R is a set of latent relations compatible with the connections of Ψ.

In the general case, both quantities are optimized jointly, considering all possible relations between each set of elements associated to each possible Ψ, and minimizing the cost over the whole sequence X.

Assuming that $\mathcal{F}$ is measuring the complexity of the sequence structure, $\mathcal{D}^X$ can be defined as the shortest description of the sequence. Therefore, searching for $\mathcal{D}^X$ can be seen as a Minimum Description Length (MDL) problem [21] and $\mathcal{F}$ can be understood as a function that evaluates the size of the "shortest" program needed to reconstruct the data [9]. This is strongly related to the concept of Kolmogorov complexity, which has received increasing interest in the music computing community over the past years [12, 14, 15, 19].

In the general case, the above optimization problem may turn out to be of a relatively high combinatorial complexity (see [10, 11]). But when considering triads over a circular arrangement, and limiting the set of possible Ψ to 6 PPPs, the optimization of $\mathcal{D}$ becomes easily tractable: all six PPPs can be tested exhaustively and for each of them, the set R comprises 16 relations (15 displacements over the triadic circle + the initialization of x_0) which are uniquely defined. Therefore, each cost can be readily computed as the sum of 16 terms, and the minimal PPP is easily found.

4. EXPERIMENTS

In order to study the ability of the PGLR model to capture structural information in chord sequences, we have carried out a set of experiments on the RWC POP dataset [8] on a corpus of 727 $\times$ 16 beat-synchronous chords sequences annotated manually as triads [3].

As there exists no ground truth as of the actual structure of a chord sequence, we compare different models as regards their ability to predict and compress chord sequences: in other words, how much side information is brought by the structure model, that saves information needed to describe of the content.

4.1 Distribution of PPPs

For each chord sequence X, the polytopic S&C graph P_X, corresponding to the PPP with minimal cost can be estimated by the optimization algorithm of Section 3.4. This yields the histogram depicted on Figure 5 [4].

Permutation $P3$ appears as the dominant one ($\approx 33\%$) and this may be related to the fact that its prototypical carrier sequence corresponds to a rather common "antecedent-consequent" form in music (especially, in pop music). Conversely, the least frequent PPP ($P2$), displays a frequency of occurrence below 5%. Somewhere in between, the 4 other permutations see their frequencies ranging loosely within 10% to 20%).

[3] Data are available on *musicdata.gforge.inria.fr/RWCAnnotations.html*

[4] About $2/3$ of test sequences correspond to a unique optimal PPP but when $k > 1$ permutations provide equally optimal solutions, each of them is counted as $1/k$.

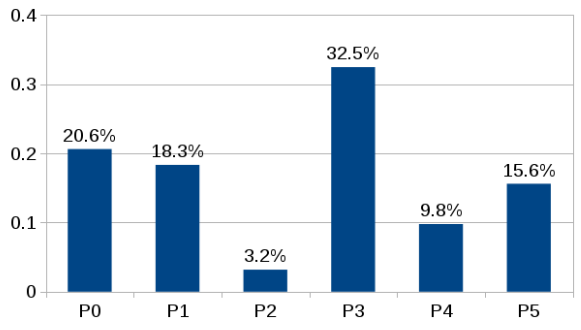


Figure 5. Histogram of PPPs across the test data.

4.2 Prediction and Compression

In order to compare the prediction and compression capabilities provided by multiscale polytopic graphs, we consider 4 structure models:

- S, a purely sequential graph where each element is related to its immediate predecessor [5] ,
- P_0, the polytopic S&C graph corresponding to PPP P_0 for all sequences,
- P_3, the polytopic S&C graph corresponding to PPP P_3 for all sequences,
- P_X, the polytopic S&C graph corresponding to the PPP with minimal cost optimised a posteriori for each 16-chord sequence, as described in section 3.4.

All models are first-order models, in the sense that any given element within the sequence is related to a single antecedent (its time predecessor for the sequential graph, a primer or a virtual element, in the case of the S&C model).

Performance for each model is obtained by calculating a *perplexity* [4] B^*, derived from the *entropy* H^*.

Given a model M, the computation of perplexity requires the definition of a probability density function (pdf) for all observable events which underlie the model. In our case, this means assigning a probability value $P_M(x_i|\Phi(x_i))$ to any pair $(x_i, \Phi(x_i))$, where $\Phi(x_i)$ is the antecedent of x_i in the graph. This is equivalent to defining $P_M(r(\Phi(x_i), x_i))$, where $r(\Phi(x_i), x_i)$ is the relation which turns $\Phi(x_i)$ into x_i. Considering a simple rotation angle $\theta(x_2|x_1) = \theta_2 - \theta_1$ on the triadic circle, $P_M(r(x_1, x_2))$ is a pdf that takes $z = 24$ distinct values.

The entropy of model M can be computed as:

$$H^*(M) = -\sum_{k=0}^{z} P_M(r_k) \log_2 P_M(r_k) \qquad (3)$$

$B^* = 2^{H^*}$ can be interpreted as a branching factor, that is the equivalent number of distinct relations between two chords, if these relations were equiprobable. It measures the compression capacity of the model and is smaller for models which capture more information in the data.

[5] This corresponds to a sequential bi-gram model, a very common approach in MIR [18].

	Triad Circle		
	Third	Phase	Random
$B(S)$	8.00	7.67	9.32
$B(P_0)$	6.68	6.77	7.84
$B(P_3)$	5.35	5.35	6.02
$B(P_X)$	**4.63**	**4.63**	**5.21**
$B_{tot}(P_X)$	5.18	5.18	5.83

Table 2. Average cross-perplexity obtained for the various models on RWC-Pop data with 2-fold cross-validation (training on even songs + testing on odd songs and vice-versa).

In this work, we consider specifically the *cross-perplexity* B derived from the *negative log likelihood* (NLL) $\hat{H}$, computed on a test-set (of L observations). In that case, the capacity of the model to catch relevant information from an unseen musical segment is measured by means of a cross-entropy score, which quantifies the ability of the model to predict unknown sequences from a similar (yet different) population.

For a given model M, $\hat{H}$ is defined as:

$$\hat{H}(M) = -\frac{1}{L}\sum_{i=0}^{L-1} \log_2 P_M(x_i|\Phi(x_i)) \qquad (4)$$

with the convention $P(x_0|\Phi(x_0)) = 1/24$.

In that context, the cross-perplexity $B = 2^H$ can be understood as an estimation of the (per symbol) average branching factor in predicting the sequence knowing its structure, on the basis of probabilities learnt on *other* sequences, assumed to be of the same sort.

Additionnally, for model P_X, we also compute the total entropy $\hat{H}_{tot}(P_X) = \hat{H}(P_X) + Q$, which includes the number of bits needed to encode the optimal configuration of the PPP (1 among 6) for each sequence of 16 chords, namely:

$$Q = \log_2(6)/16 \approx 0.16 \text{ bits/symbol}, \qquad (5)$$

this term being equal to 0 for the other models.

The first column in Table 2 compares cross-perplexity figures obtained with the 4 structure models and considering the circle of thirds for modeling relations between chords. These results show that all tested polytopic models outperform the sequential model, with an additional advantage for the P_X approach, even when taking into account the overhead cost required for PPP configuration encoding.

4.3 Impact of the Triadic Circle

In the rest of Table 2, cross-perplexity values are provided for two other circles of triads: the circle of phase-shifts as defined on Fig. 2 and a randomized circle, where triads are positioned at random. Results show that the phase circle performs quite the same as the circle of thirds, whereas the randomized circle clearly performs less well. All outperform their counterpart in the sequential model, as for all polytopic models, the identity relation is of zero cost and higher probability. We do not report perplexities on the chromatic circle, given that it is congruent to the circle of thirds, thus yielding strictly identical results.

US	LS_1	LS_2	LS_3	LS_4
44.4 %	8.0 %	15.7 %	19.7 %	22.4 %

Table 3. Proportion of sequences with contrastive US (Upper-Scale system) and LS_k (k^{th} Lower-Scale system).

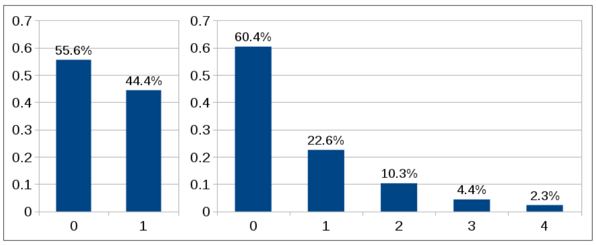


Figure 6. Proportion of contrastive systems within US systems (left) and the 4 LS systems (right)

4.4 Distribution and Density of Contrasts

To study the specific relationship between the virtual and the contrastive element in the P_X scheme, we investigated on the location and the number of contrastive vs. non-contrastive elements in potentially contrastive positions defined by the PPP framework.

Table 3 presents the distribution of actual contrasts for the Upper-Scale (US) and the 4 Lower-Scale (LS) in contrastive positions. While 44.4% of Upper-Scale Systems are contrastive, it can also be noted that the frequency of LS contrasts (or, so to speak, the occurrence of surprises at the lower-scale span) increases with the index of the LS system (i.e., its depth in the tesseract).

Figure 6 depicts the proportion of sequences as a function of the number of actual contrasts observed in different contrastive positions. It can be observed that the number of contrastive Lower-Scale systems decays (roughly exponentially) from over 60.4% of sequences with no contrastive Lower-Scale system down to only 2.3% with all 4 LS systems being contrastive.

It would surely be interesting to compare these profiles across different music genres and a variety of musical dimensions, in order to study possible correlations.

4.5 Contrast Intensity

Table 4 reports the perplexity obtained when considering separately the systemic positions and the contrastive positions. Keeping in mind that they may be specific to the corpus, results show nevertheless two very interesting trends.

Perplexity is higher in systemic positions (5.6) as opposed to constrastive positions (3.5), implying that the actual observations in contrastive positions often correspond (or are close) to the projective implication. This can be related to the results observed in the previous section, w.r.t. the relatively low density of actual contrasts.

However, when different from identity (column Diff), these relations show a *lower* perplexity for systemic relations (14.6 vs 18.7) indicating that, when a relation is not identity, the contrast is more unpredictable and/or more

	All	Diff
Systemic position	5.6	14.6
Contrastive position	3.5	18.7

Table 4. Perplexity of relations for systemic relations and contrastive relations, including (All) or excluding (Diff) the identity relation.

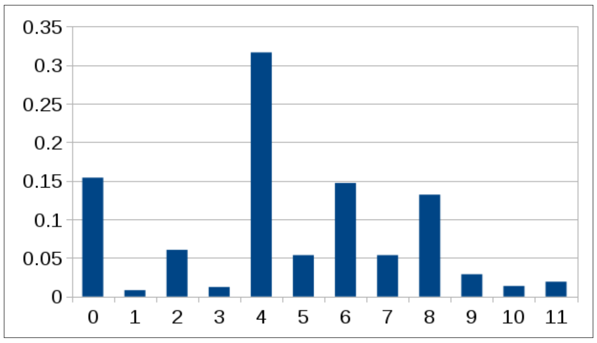


Figure 7. Proportion of chord sequences showing n distortions relative to their Prototypical Carrier Sequence (PCS).

distant on the circle of thirds, than it is for systemic relations.

In summary, strictly contrastive relations tend to be less frequent but more intense than systemic relations. This certainly relates to the presumed role of contrasts as carrying a strong quantity of surprise. These observations may be a motivation for a different treatment of systemic relations vs. contrastive ones.

4.6 Distortion of Prototypical Carrier Sequences

Ultimately, we considered the distribution of the number of distortions between observed chord sequences and their PCS, as defined in section 3.3. Figure 7 shows a dominance of 4 deviations, with an overall prevalence of even values, suggesting that modelling systems of *relations* (i.e. edges) within the tesseract could be useful to further improve the compression capabilities of the PGLR model.

5. CONCLUSIONS

Both from the conceptual and experimental viewpoints, the polytopic approach presented in this article appears as an efficient way to model multiscale relations in chord sequences.

While still at an early stage of development, the PGLR model provides a potentially useful and powerful framework for a number of tasks in MIR, as well as interesting tracks for music analysis and generation. Indeed, the core principles of the PGLR scheme are not specific to chord sequences: its application to other types of musical objects, such as melodic motives and rhythmic patterns are currently being explored.

Ongoing work also includes the extension of the polytopic model to a wider range of timescales, and the handling of segments of irregular size.

6. REFERENCES

[1] Emmanuel Amiot. *The Torii of Phases*, pages 1–18. Springer, Mathematics and Computation in Music: 4th International Conference, MCM 2013, Montreal, QC, Canada, June 12-14, 2013. Proceedings, Berlin, Heidelberg, 2013.

[2] Frédéric Bimbot, Emmanuel Deruty, Gabriel Sargent, and Emmanuel Vincent. Semiotic structure labeling of music pieces: concepts, methods and annotation conventions. In *Proc. ISMIR*, 2012.

[3] Frédéric Bimbot, Emmanuel Deruty, Gabriel Sargent, and Emmanuel Vincent. System & Contrast : A Polymorphous Model of the Inner Organization of Structural Segments within Music Pieces. *Music Perception*, 33:631–661, June 2016. Former version published in 2012 as Research Report IRISA PI-1999, hal-01188244.

[4] Peter F Brown, Vincent J Della Pietra, Robert L Mercer, Stephen A Della Pietra, and Jennifer C Lai. An Estimate of an Upper Bound for the Entropy of English. *Computational Linguistics*, 18(1):31–40, 1992.

[5] Clifton Callender, Ian Quinn, and Dmitri Tymoczko. Generalized voice-leading spaces. *Science*, 320(5874):346–348, 2008.

[6] Richard Cohn. *Audacious Euphony: Chromatic Harmony and the Triad's Second Nature*. Oxford University Press, 2011.

[7] Emmanuel Deruty, Frédéric Bimbot, and Brigitte Van Wymeersch. Methodological and Musicological Investigation of the System & Contrast Model for Musical Form Description. Research Report RR-8510, INRIA, 2013. hal-00965914.

[8] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC Music Database: Popular, Classical and Jazz Music Databases. In *ISMIR*, volume 2, pages 287–288, 2002.

[9] Peter Grunwald and Paul Vitányi. Shannon Information and Kolmogorov Complexity. *arXiv preprint cs/0410002*, 2004.

[10] Corentin Louboutin and Frédéric Bimbot. Description of Chord Progressions by Minimal Transport Graphs Using the System & Contrast Model. In *ICMC 2016 - 42nd International Computer Music Conference*, Utrecht, Netherlands, September 2016.

[11] Corentin Louboutin and Frédéric Bimbot. Polytopic Graph of Latent Relations: A Multiscale Structure Model for Music Segments. In *MCM 2017 - 6th International Conference of the Society of Mathematics and Computation in Music*, Mexico City, Mexico, June 2017.

[12] Corentin Louboutin and David Meredith. Using General-Purpose Compression Algorithms for Music Analysis. *Journal of New Music Research*, 45(1):1–16, 2016.

[13] Matthias Mauch, Katy Noland, and Simon Dixon. Using Musical Structure to Enhance Automatic Chord Transcription. In *ISMIR*, pages 231–236, 2009.

[14] Panayotis Mavromatis. Minimum Description Length Modelling of Musical Structure. *Journal of Mathematics and Music*, 3(3):117–136, 2009.

[15] David Meredith. Music analysis and Kolmogorov complexity. *Proceedings of the 19th Colloquio d'Informatica Musicale (XIX CIM)*, 2012.

[16] Eugene Narmour. *The Analysis and Cognition of Melodic Complexity: The Implication-Realization Model*. University of Chicago Press, 1992.

[17] Hélène Papadopoulos and George Tzanetakis. Modeling chord and key structure with markov logic. In *ISMIR*, pages 127–132, 2012.

[18] Marcus Thomas Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. City University London, 2005.

[19] David Temperley. Probabilistic Models of Melodic Interval. *Music Perception*, 32(1):85–99, 2014.

[20] Dmitri Tymoczko. Scale Theory, Serial Theory and Voice Leading. *Music Analysis*, 27(1):1–49, 2008.

[21] Paul MB Vitányi and Ming Li. Minimum description length induction, Bayesianism, and Kolmogorov complexity. *IEEE Trans. Information Theory*, 46(2):446–464, 2000.

[22] Xinquan Zhou and Alexander Lerch. Chord Detection Using Deep Learning. In *Proceedings of the 16th ISMIR Conference*, volume 53, 2015.

STRUCTURED TRAINING FOR LARGE-VOCABULARY CHORD RECOGNITION

Brian McFee[1,2]
[1]Center for Data Science
New York University
`brian.mcfee@nyu.edu`

Juan Pablo Bello[2]
[2]Music and Audio Research Laboratory
New York University
`jpbello@nyu.edu`

ABSTRACT

Automatic chord recognition systems operating in the large-vocabulary regime must overcome data scarcity: certain classes occur much less frequently than others, and this presents a significant challenge when estimating model parameters. While most systems model the chord recognition task as a (multi-class) classification problem, few attempts have been made to directly exploit the intrinsic structural similarities between chord classes.

In this work, we develop a deep convolutional-recurrent model for automatic chord recognition over a vocabulary of 170 classes. To exploit structural relationships between chord classes, the model is trained to produce both the time-varying chord label sequence as well as binary encodings of chord roots and qualities. This binary encoding directly exposes similarities between related classes, allowing the model to learn a more coherent representation of simultaneous pitch content. Evaluations on a corpus of 1217 annotated recordings demonstrate substantial improvements compared to previous models.

1. INTRODUCTION

Automatic chord recognition has been an active area of research within music informatics for nearly two decades [8]. Chord recognition systems take as input an audio signal, and produce a time-varying symbolic representation of the signal in terms of *chord labels*, which encode simultaneous pitch class content, such as `C:maj` or `G:hdim7`. Many systems focus on simplified versions of this task, by predicting only the root note and *major* or *minor* qualities, or *no-chord* (`N`). Recently, interest has shifted toward the *large-vocabulary* regime, where a broader class of chord qualities must be estimated, such as triads, sixths, sevenths, and suspended chords.

Typical chord recognition systems model the task as a time-varying multi-class classification problem. This approach may be reasonable for the small-vocabulary regime, where the classes are sufficiently distinct to be modeled as unrelated, and each class may be observed with approximately uniform probability. However, in the large-vocabulary setting, the multi-class formulation ignores the structural similarity between related chords, such as the shared notes between `C:min` and `C:min7`. Moreover, the distribution of classes becomes highly skewed, thereby making it difficult to model these relationships from purely symbolic representations with no additional structure. We hypothesize that leveraging known relationships between chord classes in terms of common roots and shared pitch classes can help mitigate the problem of observation bias, resulting in more accurate models of rare classes.

1.1 Our Contributions

We address the problem of large-vocabulary chord recognition by introducing a structured representation of chord qualities, which decouples the problem of detecting roots and pitch classes from the problem of mapping these properties onto symbolic labels. We integrate this representation with deep, convolutional-recurrent neural networks, which are trained end-to-end to predict time-varying chord sequences from spectral audio representations. The proposed models achieve substantially higher accuracy than previous models based on convolutional networks and hidden Markov models, resulting in absolute gains of 4–5% in the most difficult categories (sevenths and tetrads).

2. RELATED WORK

Chord recognition has received a substantial amount of attention in the MIR literature, and a comprehensive survey of existing methods is beyond the scope of this paper. Here, we highlight the work that is most closely related to the proposed methods in this paper.

Hidden Markov models (HMMs) have been a popular method for designing chord recognition systems, and provide a flexible framework in which to integrate musical domain knowledge. The general HMM approach models chord identities as latent state variables to be inferred from observed time-series features (*e.g.*, chroma vectors). Systems like Chordino [17] and HPA [20] extend this idea by introducing additional latent variables to model key, bass, and metrical position. In these systems, bass is modeled by weighting or partitioning the frequency range to produce distinct *bass* and *treble chroma* observations. The K-stream HMM takes this idea a step further by modeling K

distinct frequency sub-bands, though it does not explicitly infer bass [5]. The structured representation we describe in Section 3 differs in that root, bass, and chord quality are jointly inferred from the entire spectrum, and it makes no assumptions about absolute height. Weller *et al.* [23] also adapted structured training techniques for chord recognition, but at the level of dynamics rather than the chord vocabulary.

In recent years, deep learning methods have been increasingly popular for chord recognition. The majority of existing systems are trained in two stages. First, a model is built first to encode short patches of audio, *e.g.*, as an idealized chroma vector [2, 16] or likelihood distribution over chord categories [7, 11, 22, 24]. Second, a dynamics model integrates the time-series of learned representations to produce a sequence of predicted labels, *e.g.*, using an HMM [11, 24], recurrent neural network (RNN) [2, 22], or simple point-wise prediction [16]. The models proposed here differ in that they are jointly trained end-to-end from spectral features, and learn the internal representation along with the dynamics using multiple recurrent layers.

Regardless of the model architecture, it is common to exploit some structural properties of chords, *e.g.*, by tying model parameters for the same chord quality across roots [11], or rotating chroma vectors through all possible root positions during training [5]. Although the methods we propose do not model quality independent of root, they do model active pitch class sets independently. Chroma rotation can be viewed as a form of data augmentation, and the models we develop benefit substantially from a slightly more general form of augmentation described in Section 3.4. To the best of our knowledge, the proposed method is the first to exploit similarities between chords by jointly modeling labels and structured encodings.

3. METHODS

This section outlines the data preparation, architectures, and training strategies for the models under comparison. We consider three independent design choices: convolutional or recurrent decoding, the inclusion of structured output training, and the use of data augmentation. This results in eight model configurations.

3.1 Encoder-decoder models

The models we investigate fall under the umbrella of *encoder-decoder* architectures [3]. The *encoder* component maps time-varying input (audio) into a latent feature space, while the *decoder* component maps from the latent feature space to the output space (chord labels).

3.1.1 The encoder architecture

The encoder, and depicted in Figure 1, is common to all models considered in this paper. Input audio is represented as a $T \times F$ time-series of log-power constant-Q transform (CQT) spectra (for T frames and F frequency bands). After batch normalization [13], the first convolutional layer consists of a single two-dimensional 5×5 filter, followed

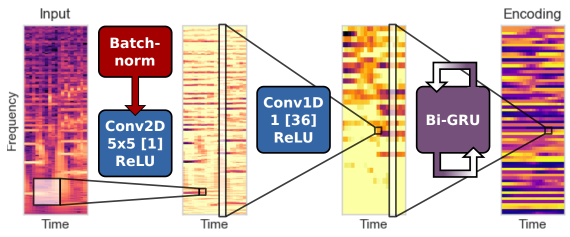


Figure 1. The encoder module uses a convolutional-recurrent network architecture to map the input (CQT frames) to a sequence of hidden state vectors $h(t) \in \mathbb{R}^D$.

by a bank of 36 single-frame, one-dimensional convolutional filters, resulting in a $T \times 36$ feature map. Both layers use rectified linear (ReLU) activations. The first layer can be interpreted as a harmonic saliency enhancer, as it tends to learn to suppress transients and vibrato while emphasizing sustained tones. The second layer summarizes the pitch content of each frame, and can be interpreted as a local feature extractor.

Finally, the local features are encoded by a bi-directional gated recurrent unit (GRU) model [4]. The GRU model is similar to the long-short-term memory (LSTM) model [10], but has fewer parameters and performs comparably in practice [14]. For a sequence of d-dimensional input vectors $x(t) \in \mathbb{R}^d$, a GRU layer produces a sequence of D-dimensional hidden state vectors $h(t) \in [-1, +1]^D$ as follows:

$$
\begin{aligned}
r(t) &= \sigma\left(W_r x(t) + T_r h(t-1) + b_r\right) && (1)\\
u(t) &= \sigma\left(W_u x(t) + T_u h(t-1) + b_u\right) && (2)\\
\hat{h}(t) &= \rho\left(W_h x(t) + T_h\left(r(t) \odot h(t-1)\right) + b_h\right) && (3)\\
h(t) &= u(t) \odot h(t-1) + (1 - u(t)) \odot \hat{h}(t), && (4)
\end{aligned}
$$

where $r(t), u(t) \in [0, 1]^D$ are the *reset* and *update* vectors, each of which are controlled by RNN dynamics depending on the input $x(t)$ and previous hidden state $h(t-1)$, $\sigma(x) = (1 + \mathrm{e}^{-x})^{-1}$ denotes the logistic function, and $\rho = \tanh$. The parameters are the input mappings $W_* \in \mathbb{R}^{D \times d}$, transition operators $T_* \in \mathbb{R}^{D \times D}$, and bias vectors $b_* \in \mathbb{R}^D$.

When an element j of the update vector $u_j(t) \approx 1$, the corresponding element of the previous hidden state is copied directly to the current state $h_j(t) \leftarrow h_j(t-1)$. Otherwise, if $r(t) \approx 1$, then $h(t)$ evolves according to standard RNN dynamics. However, when both $u(t), r(t) \approx 0$, the h term in (3) goes to 0 and the update *resets*, depending only on the input $x(t)$. This allows the GRU model to persist a hidden state across arbitrarily long spans of time, and capture variable-length temporal dependencies. These properties make the GRU model appealing for chord recognition, where dependencies may span long ranges (compared to frames), and are subject to sudden changes rather than gradual evolution.

The bi-directional variant consists of two independent GRUs, one running in each temporal direction, whose hidden state vectors are concatenated to produce the bi-directional hidden state vector $h(t)$. This layer integrates over the entire input signal, and provides temporal smoothing and context for the encoded feature representation.

3.1.2 Decoder architectures

We investigate two models, depicted in Figure 3, for decoding $h(t)$ to the sequence of chord labels $\hat{y}(t)$. The first model, denoted *CR1*, decodes each frame independently:

$$\hat{y}(t) := \text{softmax}\left(W_y h(t) + b_y\right), \tag{5}$$

where the soft-max operates over the chord vocabulary V, producing a likelihood vector $\hat{y}(t) \in [0,1]^{|V|}$. For the *CR1* architecture, we set the dimensionality of the hidden state vector to 512 (256 for each temporal direction).

The second model, denoted *CR2*, uses a bi-GRU layer to map $h(t)$ to an intermediate representation $h_2(t)$ prior to frame-wise decoding by eq. (5). To keep the number of parameters roughly comparable between *CR1* and *CR2*, we set the dimensionality *CR2*'s recurrent layers to 256.

For each configuration, all model parameters Θ are jointly trained to maximize the empirical log-likelihood:

$$\underset{\Theta}{\text{argmax}} \sum_t \sum_{c \in V} y_c(t) \log \hat{y}_c(t), \tag{6}$$

where the reference labels are one-hot encoded vectors $y(t) \in \{0,1\}^{|V|}$. While both architectures have access to the entire observation sequence, *CR2* may be better at capturing long-range interactions. This should allow the encoder to focus on short-term smoothing and local context, while the decoder can model chord progressions and global context. In the *CR1* model, the encoder is responsible for both short- and long-range interactions.

At test time, the maximum likelihood label is selected for each frame, and the series of chord labels is run-length encoded to form the estimated annotation for the track.

3.2 Chord vocabulary simplification

To formulate chord recognition as a classification task, we define a mapping of all valid chord descriptions to a finite vocabulary V.[1] First, inversions and suppressed or additional notes are discarded, *e.g.*:

$$\texttt{D}\flat\texttt{:maj(9)/3} \quad \mapsto \quad \texttt{D}\flat\texttt{:maj/3} \quad \mapsto \quad \texttt{D}\flat\texttt{:maj}.$$

Next, labels are decomposed into *root* and *pitch classes* (relative to the root) using `mir_eval` [21]:

$$\texttt{D}\flat\texttt{:maj} \mapsto \begin{cases} 1 & \text{root} \\ (0,4,7) & \text{pitch classes} \end{cases}.$$

The set of active pitch classes is matched against 14 templates: `min, maj, dim, aug, min6, maj6, min7, minmaj7, maj7, 7, dim7, hdim7, sus2, sus4`. The root and matched template are combined, and mapped to a canonical form to resolve enharmonic equivalences:

$$(1, (0,4,7)) \mapsto \texttt{C}\sharp\texttt{:maj}.$$

If the pitch class set does not match one of the templates, it is mapped to the unknown chord symbol `X`; the no-chord symbol is represented distinctly as `N`. The final vocabulary contains 170 classes: 2 special symbols (`N, X`), and $12 \times 14 = 168$ combinations of root and quality.

[1] A *valid chord* is any string belonging to the formal language of Harte *et al.* [9], or the extended grammar implemented by JAMS [12].

3.3 Structured training

The CR models described above map each hidden state vector $h(t)$ to a fixed vocabulary produced described in Section 3.2. They can be optimized in the usual way to maximize (6), but this approach has some clear drawbacks.

First, it does not leverage the inherent structure of the space of chords. If the model predicts `B:maj` instead of `B:7`, it is penalized just as badly as if it had predicted `C:maj`. This is at odds with evaluation, where predictions are evaluated along multiple dimensions, such as capturing the root, third, or fifth. More generally, some mistakes are simply more severe than others, and this is not reflected in a 1-of-K classification formulation.

Second, the chord simplification strategy is *lossy* in that it discards information such as suppressed or additional notes. This can render certain chords ambiguous, and can introduce discrepancies between the (simplified) annotation and the corresponding acoustic content. Continuing the D♭:maj(9)/3 example, the simplification C♯:maj implies the absence of D♯, although it was explicitly included in the original annotation and should be expected in the signal. This introduces label noise to the model, and may negatively impact accuracy.

Third, out-of-gamut chords all map to a common class `X`, despite having disparate roots and tonal content. This class provides little useful information to the model while training. At test time, it would be beneficial if the model could predict "nearby" chords, but multi-class training provides little incentive to learn this behavior.

To counteract these effects, we introduce a structured representation, depicted in Figure 2. This is inspired by the standard evaluation criteria for chord recognition, which operate over a decomposed representation of (*root*, *pitch classes*, *bass*) [21]. This representation can be computed for any valid chord label, and provided as supervision to the model, thereby helping it learn common features shared by similar chords. At prediction time, the structured representation is used as an intermediate representation which contributes to the chord label prediction, which can now be interpreted as a human-readable decoding of the structured representation.

The structured models (denoted as *CR1/2+S*), depicted in Figure 3, predict for each frame t the root pitch class (`C`–`B`, plus `N` for no-root), the bass pitch class, and the active pitch classes from the hidden state vector $h(t)$. Root and bass estimation are modeled as a multi-class prediction with a soft-max non-linearity. Pitch class prediction

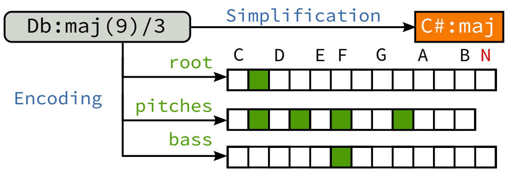


Figure 2. Target chords are represented in both simplified canonical form (Section 3.2), and as binary vectors encoding the root, bass, and pitch classes (Section 3.3). The special symbols `N`, `X` map to an extra root/bass class `N`, and the all-zeros pitch vector.

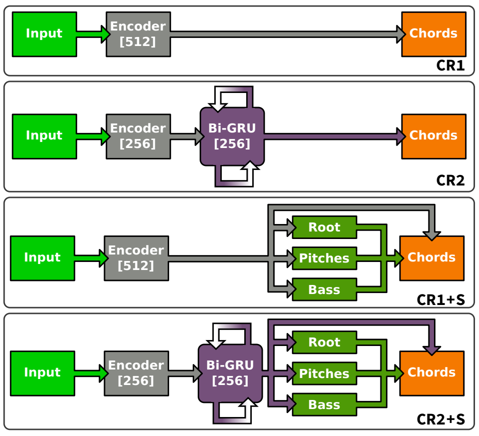


Figure 3. Block diagrams of all architectures described in Section 3. The *encoder* block is depicted in Figure 1.

is modeled as a multi-label prediction, and uses a logistic (sigmoid) non-linearity. This results in an idealized chroma representation similar to that of Korzeniowski and Widmer [16], but estimated from the full input observation rather than a fixed spectrogram patch. An illustrative example of this predicted encoding is provided in Figure 4.

It is generally non-trivial to invert the root-pitch-bass representation to a unique chord label. Therefore, these three layers are concatenated, along with the hidden state $h(t)$, to produce the structured representation from which the chord label is predicted. During training, the structured models learn to minimize the sum of losses across all outputs: root, pitches, bass, and label $\hat{y}(t)$. Minimizing the *root* and *pitches* losses corresponds to maximizing the *root* and *tetrads* recall scores during training, while eq. (6) learns the decoding into the human-readable chord vocabulary. This formulation effectively decouples the problems of root and pitch class identification from chord annotation, which is known to be subjective [11].

3.4 Data augmentation

To increase training set variability, we apply pitch-shifting data augmentation using MUDA [18]. For each training example, 12 deformations are generated by shifting up or down by between 1–6 semitones. Because each observation exists in all twelve root classes, this provides a brute-force, approximate root invariance to the model. Models trained with data augmentation are denoted by *+A*.

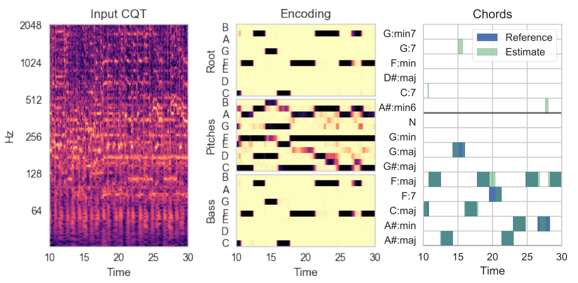


Figure 4. The predicted chord encodings and labels for *The Beatles — Hold Me Tight* by model *CR2+S+A*.

4. EVALUATION

For evaluation, we used the dataset provided by Humphrey and Bello [11], which includes 1217 tracks from the Isophonics, Billboard, RWC Pop, and MARL collections. To facilitate comparison with previous work, we retain the same 5-fold cross-validation splits, and randomly hold out 1/4 of each training set for validation. We compare to two strong baselines: a deep convolutional network [11] (denoted *DNN*), and the K-stream HMM [5] (*KHMM*). [2]

4.1 Pre-processing

Feature extraction was performed with librosa 0.5.0 [19]. Each track was represented as a log-power constant-Q spectrogram with 36 bins per octave, spanning 6 octaves starting at C1, and clipped at 80dB below the peak. Signals were analyzed at 44.1KHz with a hop length of 4096 samples, resulting in a frame rate of approximately 10.8Hz.

4.2 Training

All models are trained on 8-second patches (86 frames), though they readily support input of arbitrary length. For tracks with multiple reference annotations, the output is selected uniformly at random from all references for the patch, which reduces sampling bias toward specific annotators. Models are trained using mini-batches of 32 patches per batch, and 512 batches per epoch. We use the ADAM optimizer [15], and reduce the learning rate if there is no improvement in validation score after 10 epochs. Training is stopped early if there is no improvement in validation score after 20 epochs, and limited to a maximum of 100 epochs total. For all models, validation score is determined solely by label likelihood (eq. (6)). All models were implemented with Keras 2.0 and Tensorflow 1.0 [1, 6]. [3]

4.3 Results

The main results of the evaluation are listed in Figure 5, which illustrates the median weighted recall scores achieved by each model. [4] Each subplot reports the recall

[2] Comparisons were facilitated using the pre-computed outputs provided at `https://github.com/ejhumphrey/ace-lessons`.

[3] Our implementation is available at `https://github.com/bmcfee/ismir2017_chords`.

[4] The trends for the mean scores are qualitatively similar, but the scores are lower for all models. We report the median here to reduce the in-

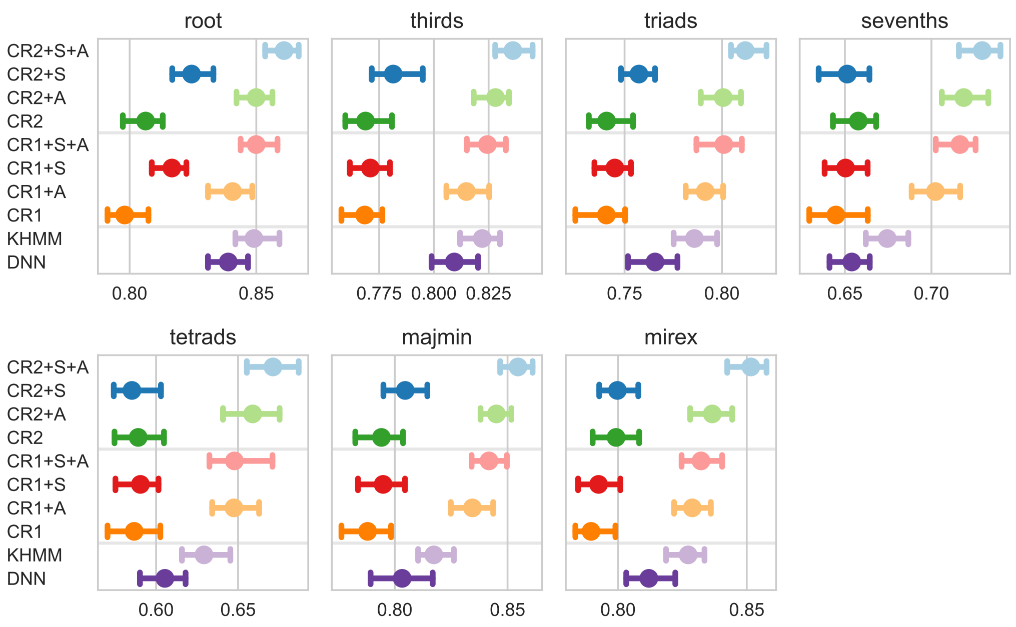


Figure 5. Weighted recall scores for all methods under comparison. Each dot represents the median score across all test points, with error bars covering the 95% confidence interval estimated by bootstrap sampling. *KHMM* denotes the K-stream HMM of Cho [5]; *DNN* denotes the convolutional network of Humphrey and Bello [11].

scores computed by `mir_eval`: 1. *root*; 2. *thirds*: root and third; 3. *triads*: root, third, and fifth; 4. *sevenths*: root, third, fifth, and seventh; 5. *tetrads*: all intervals; 6. *maj-min*: 12 major, 12 minor, and N class; and 7. *MIREX*: at least three correct notes.

From Figure 5, several trends can be observed. First, data augmentation (*+A* variants) provides a consistent and substantial improvement for all models. This is to be expected, since the *CR* models do not separate root from quality. Note that DNN models these independently, and KHMM was trained with chroma-rotation data augmentation, so it is unsurprising that augmentation is necessary to match performance of these methods.

Second, structured training (*+S* variants) provides a modest, but consistent improvement, for both the shallow *CR1* and deep *CR2* decoder models. The difference is most pronounced in the *root* evaluation, which is expected due to the explicit objective to correctly identify the root.

Third, the deep decoder models *CR2* provide another small, but consistent improvement over the shallow decoders *CR1*. The aggregate scores are reported in Table 1; for brevity, only the models with data augmentation are included. The combined effect of structured training, deep decoder, and data augmentation (*CR2+S+A*) results in the highest scoring model across all metrics.

4.4 Error analysis

To get some more insight about the mistakes made by the model at test time, we illustrate the frame-wise, within-

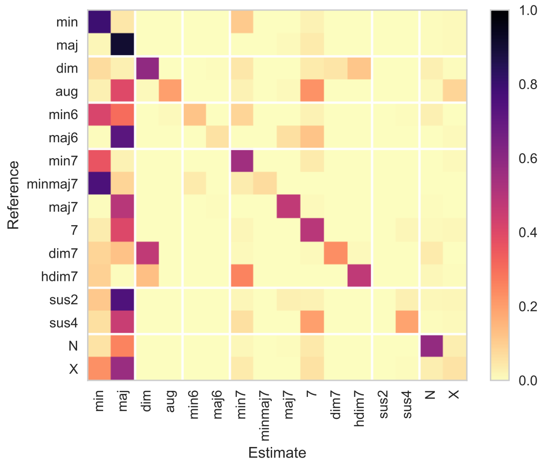


Figure 6. Within-root, frame-wise quality confusions for the best performing model *CR2+S+A*. The value at row i, column j corresponds to the fraction of frames labeled as class i but predicted as class j.

root quality confusion matrix for the *CR2+S+A* model in Figure 6. For each frame of a test track, its (simplified) reference label is compared to the label estimated by the model if they match at the root. Results are then aggregated across all test tracks, and normalized by (reference quality) frequency to produce the confusion matrix. Under this evaluation, the *CR2+S+A* achieves 63.6% accuracy of correctly identifying the simplified chord label (root and quality) at the frame level.

fluence of the erroneous or otherwise spurious reference annotations reported by Humphrey and Bello [11].

Method	Root	Thirds	Triads	Sevenths	Tetrads	Maj-Min	MIREX
CR2+S+A	0.861	0.836	0.812	0.729	0.671	0.855	0.852
CR2+A	0.850	0.828	0.801	0.719	0.659	0.845	0.837
CR1+S+A	0.850	0.824	0.801	0.716	0.648	0.842	0.832
CR1+A	0.841	0.815	0.791	0.702	0.647	0.834	0.829
KHMM [5]	0.849	0.822	0.785	0.674	0.629	0.817	0.827
DNN [11]	0.838	0.809	0.766	0.654	0.605	0.803	0.812

Table 1. Median weighted recall scores for methods under comparison.

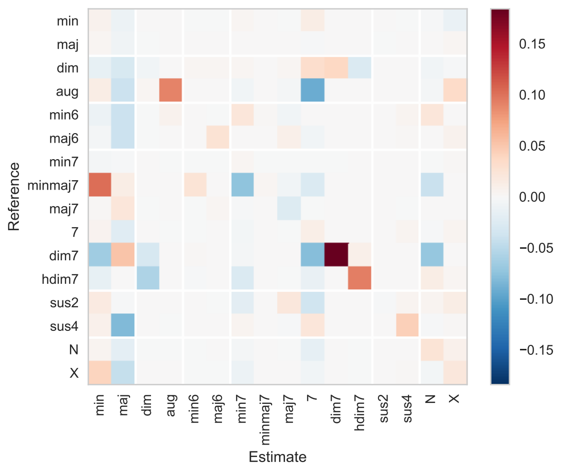


Figure 7. The difference between confusion matrices for *CR2+S+A* and the unstructured *CR2+A* (best viewed in color). Positive values along the diagonal indicate increased accuracy for *CR2+S+A*.

In Figure 6, the first obvious trend is a bias toward `min` and `maj`, in accordance with the natural bias in the training set (13.6% and 52.5% of the data, respectively, by duration). Note, however, that the confusions are generally understandable as simplifications: *e.g.*, (`min7`,`minmaj7`)→`min` and (`maj7`,`7`)→`maj`. The model still appears to struggle with 6th and suspended chords, which account for 1.5% and 2.5% of the data, respectively. The bottom row corresponds to out-of-gamut `X`-chords, which map overwhelmingly to `maj` and `min`. This can be explained by examining which labels map to `X` during simplification. There are 4557 instances of such chords in the corpus (2.2% of the data), and of these, 2091 are 1-chords (only the root) and 2365 are power chords (root+fifth), neither of which map unambiguously onto the simplified vocabulary. The model appears to resolve these toward the more commonly used `min` and `maj` qualities.

To understand the influence of structured training, Figure 7 illustrates the difference between the confusion matrices of the structured model *CR2+S+A* and the unstructured model *CR2+A*. Positive values (red) along the diagonal indicate increased accuracy for the structured model, while negative values along the diagonal (blue) indicate decreased accuracy. The net effect is positive, increasing accuracy by +0.8% over *CR2+A* (62.8%).

Despite a slight degradation for `maj7`, there are substantial improvements for `aug`, `dim7`, `hdim7`, and modest improvement for `sus4`. Moreover, the negative values in the second column reveal a consistent reduction of confusions to *maj*. This indicates that the structured model is more robust to quality bias in the training set. Compared to the unstructured model, the structured model reduces confusions from `aug` to (`maj`, `7`), and `dim7` to (`min`, `7`, `N`). The *CR2+S+A* still performs poorly on the rarest class `minmaj7` (0.03% of the data), but compared to *CR2+A*, it resolves toward `min` more often and `min7` less often. The structured model appears to be better at abstaining from predicting a seventh if it appears unlikely, rather than predict the wrong seventh.

5. CONCLUSION

This work developed deep architectures and a structured training framework for chord recognition in large vocabularies. Although the proposed models improve over the baseline methods, there are clear directions forward in extending the ideas presented here. First, although the proposed model predicts the bass note, this feature is only used for establishing context in decoding, and the model does not predict inversions. Supporting inversion prediction would be a simple extension of the method described here, and would not require creating special vocabulary entries for each potential inversion. Second, the structured representation facilitates modeling infrequently observed, complex chords, and could readily be extended to support extended chords by using a multi-octave pitch class representation. However, doing so effectively—and evaluating the resulting predictions—would require larger annotated corpora for these classes than are presently available.

6. ACKNOWLEDGMENTS

BM acknowledges support from the Moore-Sloan data science environment at NYU. We thank the NVIDIA Corporation for the donation of a Tesla K40 GPU.

7. REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek

Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340, 2013.

[3] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17(11):1875–1886, 2015.

[4] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua 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*, pages 1724–1734, 2014.

[5] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2014.

[6] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[7] Junqi Deng and Yu-Kwong Kwok. A hybrid gaussian-hmm-deep-learning approach for automatic chord estimation with very large vocabulary. ISMIR, 2016.

[8] Takuya Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *ICMC*, pages 464–467, 1999.

[9] Christopher Harte, Mark B Sandler, Samer A Abdallah, and Emilia Gómez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *ISMIR*, pages 66–71, 2005.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Eric J Humphrey and Juan Pablo Bello. Four timely insights on automatic chord estimation. In *ISMIR*, pages 673–679, 2015.

[12] Eric J Humphrey, Justin Salamon, Oriol Nieto, Jon Forsyth, Rachel M Bittner, and Juan Pablo Bello. JAMS: A JSON annotated music specification for reproducible MIR research. In *ISMIR*, pages 591–596, 2014.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456, 2015.

[14] Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2342–2350, 2015.

[15] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[16] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: the deep chroma extractor. In *ISMIR*, 2016.

[17] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *ISMIR*, 2010.

[18] Brian McFee, Eric J Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *ISMIR*, pages 248–254, 2015.

[19] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee. librosa 0.5.0, February 2017.

[20] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[21] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel PW Ellis. mir_eval: A transparent implementation of common mir metrics. In *ISMIR*, 2014.

[22] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *ISMIR*, pages 127–133, 2015.

[23] Adrian Weller, Daniel Ellis, and Tony Jebara. Structured prediction models for chord transcription of music audio. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 590–595. IEEE, 2009.

[24] Xinquan Zhou and Alexander Lerch. Chord detection using deep learning. In *ISMIR*, 2015.

Oral Session 7

Symbolic

MODELING AND DIGITIZING REPRODUCING PIANO ROLLS

Zhengshan Shi
CCRMA
Stanford University
kittyshi@ccrma.stanford.edu

Kumaran Arul
Department of Music
Stanford University
karul2@stanford.edu

Julius O. Smith
CCRMA
Stanford University
jos@ccrma.stanford.edu

ABSTRACT

Reproducing piano rolls are among the early music storage mediums, preserving fine details of a piano or organ performance on a continuous roll of paper with holes punched onto them. While early acoustic recordings suffer from poor quality sound, reproducing piano rolls benefit from the fidelity of a live piano for playback, and capture all features of a performance in what amounts to an early digital data format. However, due to limited availability of well maintained playback instruments and the condition of fragile paper, rolls have remained elusive and generally inaccessible for study. This paper proposes methods for modeling and digitizing reproducing piano rolls. Starting with an optical scan, we convert the raw image data into the MIDI file format by applying histogram-based image processing and building computational models of the musical expressions encoded on the rolls. Our evaluations show that MIDI emulations from our computational models are accurate on note level and proximate the musical expressions when compared with original playback recordings.

1. INTRODUCTION AND MOTIVATION

The invention of acoustic recordings in the late nineteenth century is widely accepted as a watershed moment in the history of music. However, piano rolls, which were in widespread use from approximately 1905 to 1940, are mistakenly treated as a footnote. Researchers studying early acoustic recordings have significant challenges with transcribing the nuances of a performance due to the poor sound fidelity, high noise artifacts, and limited recording length. Piano rolls did not share these shortcomings and were praised for their faithfulness and accuracy, providing a virtual transcription of a performance by punching holes on a paper scroll. Many important musicians who recorded on piano rolls never made acoustic recordings and were among the oldest generation to be recorded. These include composers like Claude Debussy, Scott Joplin, and Carl Reinecke, among others [13]. Modern digital audio workstations were inspired by this old music storage format and provide graphical display of MIDI (Music Instrument Digital Interface) files in what usually refers to a piano-roll editor. Thus, it can be claimed that piano rolls are among the most significant early commercial recordings and data storage media.

Playing and recording original rolls has been difficult due to the fragile paper and cumbersome demands of maintaining original instruments. Some recent efforts have pursued an alternative approach to accessing rolls by designing dedicated roll scanners to create image files of the rolls. This serves the purpose of archival preservation but does not allow piano rolls to be heard as the musical documents they were originally intended to be.

In this paper, we present a method for faithful automatic playback of reproducing piano rolls from image scans. Our research aims to digitize the data captured through analysis of notes, timings, and dynamics. The latter involves decoding proprietary expression mechanisms controlled through holes on the sides of the rolls. We evaluate our accuracy by comparing the digital file with acoustic recordings of rolls played on original player instruments.

2. RELATED WORK

There has been limited work on digitizing piano rolls by institutional researchers and independent hobbyists. Wayne Stahnke [14] is one of the first to scan piano rolls and convert them into a digital format. Stahnke transferred image information derived from a self-built roll scanner into a proprietary data format preserving the details of the punched holes. Colmenares et al [2] converted Stahnke's pre-processed data for piano rolls into MIDI information by using sparse matrices. A few researchers such as Trimpin [7], Trachtman [15], and Malosio et al [9] have worked on the emulation of piano rolls. Other individuals such as Anthony Robinson [12] developed scanning machines and software that allows manual adjustment of the punch holes for refinement and error correction in the process of MIDI generation. These pioneering efforts remain however largely inaccessible, as key algorithms are not revealed or evaluated and are unavailable for consideration and review. They also typically require intensive manual labor in the transfer process and are usually limited to one format of roll.

Our work aims to fill this gap by proposing techniques to digitize and model reproducing piano rolls including novel methods of transcribing the musical expres-

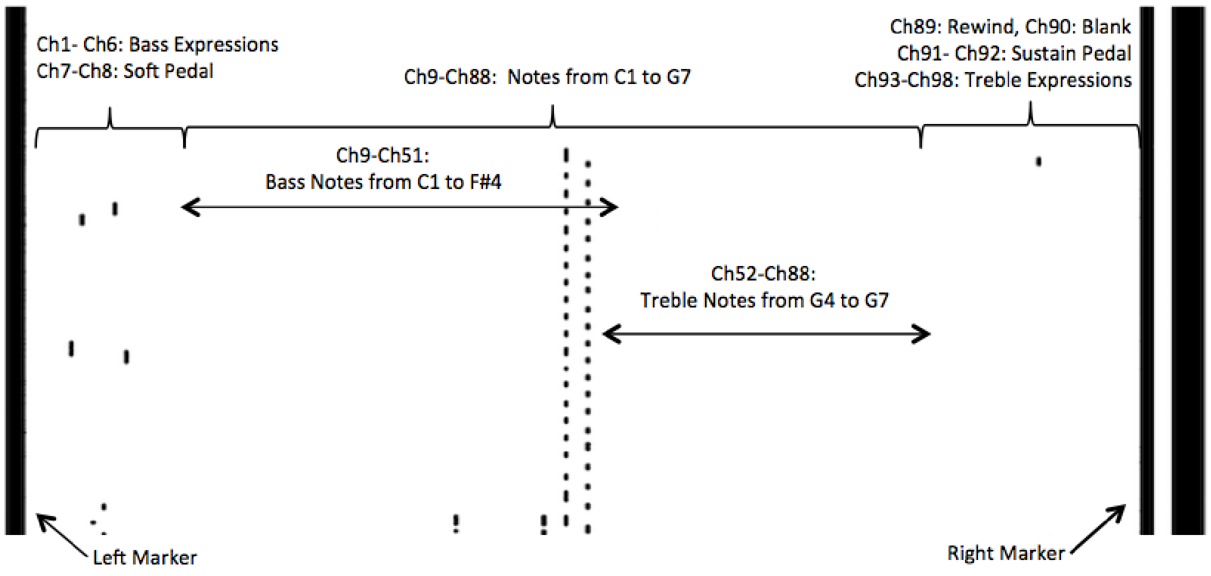

Ch1- Ch6: Bass Expressions
Ch7-Ch8: Soft Pedal
Ch9-Ch88: Notes from C1 to G7
Ch89: Rewind, Ch90: Blank
Ch91- Ch92: Sustain Pedal
Ch93-Ch98: Treble Expressions
Ch9-Ch51: Bass Notes from C1 to F#4
Ch52-Ch88: Treble Notes from G4 to G7
Left Marker
Right Marker

Figure 1. Excerpt from a Welte-Mignon piano roll image scan, consisting of the left marker, channels 1 to 98, and the right marker.

sion markings on those piano rolls. We evaluate our system by comparing the resultant MIDI emulation to the actual acoustic playback of the reproducing piano roll on a player piano.

3. REPRODUCING PIANO ROLLS FORMATS

A piano roll is a continuous roll of paper with perforations that store musical note data. It captured in real time the notes and rhythms of a pianist playing a special recording instrument. Music recorded on rolls are often performed by a player piano, a pneumatic machine that can decode the music data on the perforations and operate the piano action. Reproducing piano rolls are standard piano rolls with expression (dynamic and pedal) capabilities that can also be automatically executed by player piano. Expressions were captured in variable ways but were usually transcribed in shorthand and then coded onto the roll for bass and treble respectively. The complex process allowed for detailed editing of notes, rhythms and expressions.

The competitive environment of the early roll business created multiple different reproducing-piano systems, with variations in the size of rolls, configuration of holes, and pneumatic player designs. Although industry efforts at standardization eventually created some overlap, a thorough inventory of historical roll systems would number over two dozen. Most reproducing rolls of value to scholars are found in the catalogs of a handful of primary players, including Welte-Mignon (the first reproducing roll maker), Ampico, Duo-Art, and Hupfeld. Playing any reproducing roll requires a suitable player piano manufactured specifically for that format of roll. In some cases, as formats evolved over time, multiple players are needed to play back a manufacturer's rolls (early Red, T-100 Welte rolls do not play on later Green, T-98 Welte players, for example).

In this paper, we focus on the Welte-Mignon Licensee format (the third Welte format, a derivation of the T-98 rolls). A typical Welte-Mignon Licensee roll is 11 1/4 inches in width and holes spaced 9 per inch across [5], consisting of 98 *channels* [1] of punched holes [11]. As shown in Figure 1, channels 9 through 88 represent notes spanning from C1 to G7, with note F#4 as the splitting note for bass and treble. Channels 1 through 6 control bass (left-hand) expression, and channels 93 through 98 indicate treble (right-hand) expression. The pedal information is included in channel 7 and 8 (soft pedal) as well as channel 91 and 92 (sustain pedal). The expression channels combine to determine the dynamics of the piano performance.

4. ALGORITHM FOR DIGITIZING REPRODUCING PIANO ROLLS

In this section, we describe our method for digitizing and modeling the reproducing rolls. We first obtain piano roll images through a scanner, then construct a template that matches the layout geometry of the piano roll. Based on the template and locations of the perforations, we recover the note matrix that contain onset times for each note and musical expression. Finally we model musical expressions into dynamics and pedal information, that can be preserved in MIDI files.

4.1 Scanning

The first step in digitizing the reproducing-piano rolls is to generate the image file by scanning. The scanner used for this project was purposely built for rolls with a transport

[1] For the purpose of this paper, we refer to a "channel" as a column of the punched holes.

mechanism that allows continuous image capture by a contact image sensor (CIS) module which produces a graphic image file in either CIS or bitmap graphic format. The scan produces a directory of bitmap image files, initially preserving the 8-bit grayscale information. Each scanned image is 7296 pixels wide, juxtaposing front and back of the roll onto one side, with a pixel resolution of 324 dots-per-inch (dpi). We convert the grayscale raw image data to binary and invert the digit, so that 0 indicates no hole (white), and 1 indicates a hole coverage (black) at each pixel. On average, the size of a punched hole is 17 by 28 pixels.

4.2 Channel Grid Construction

Given a fixed piano-roll format, the edges of the zones stay sufficiently constant due to the general uniformity of the piano rolls. Thus, a quantitative template specifying piano-roll layout geometry such as the exact location for each channel and the spacing between different zones is necessary. We refer to it as a *Channel Grid*.

However, to the best of our knowledge, there is no such precise grid of hole placement available. Only general historical and empirical evidence from playback are available. Thus, using documentation of roll formats [11], and the notation of the musical works recorded, we derived reference points which were used to create a quantitative template locating the notes and expression holes on the roll.

Histogram-based image processing [6] has been used previously in optical music recognition to determine grid layouts of musical scores. Fujinaga [4] applied the histogram method to successfully detect staff lines on sheet music. We applied a similar method for the scanned piano roll images by projecting the piano-roll image onto its x axis to form a histogram of the holes in each channel. From the x-projection, we process each note-channel histogram to find the center point of each histogram peak relative to the left edge of the piano roll, as illustrated in Figure 2. We take that as defining center lines for each channel. Because the hole-channels are adequately straight and nearly parallel to the edge of the paper, the x projection histogram produces well separated "channel piles". The set of all such channel lines forms a grid on the piano roll in which each channel line is a fixed measured distance from the left edge of the paper. To map these note-center x-coordinates to MIDI note numbers, we use the first note of Waltz in E flat Major, Op.42 No.2 by Frédéric Chopin as an anchor note, namely E♭4 (MIDI note number 63). Under current image resolution, the median gap between each note channel was found to be approximately 33 pixels, which matches the roll specification of 9 holes per inch as mentioned in section 2.

Based on the distribution of the punched holes as well as our manual inspection of the roll image, we further partition the piano roll into three zones: zone 1 on the left includes the bass dynamics, zone 2 in the middle contains all the note information for the 80 keys, and zone 3 spans the treble dynamics, as shown in Figure 1. Our entire collection of Welte-Mignon Licensee rolls appears to be compatible with this template. Thus the template formation based on one example is found to be sufficient. We are able to prescribe the edges of the zones according to the plot of channel histograms generated by the system, as described above and further below.

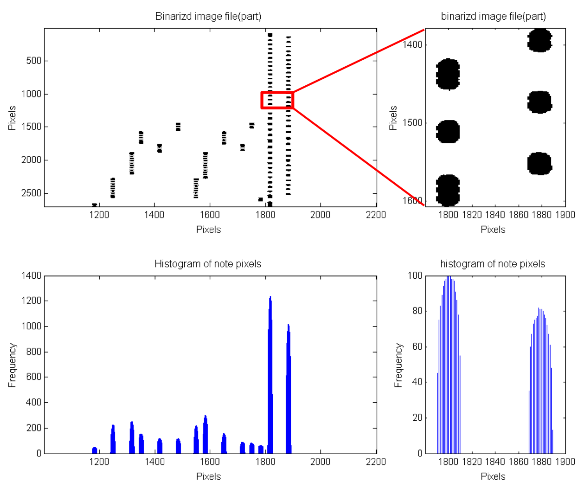


Figure 2. (Top left) Scanned image; (Top right) Zoomed-in view; (Bottom left) Histogram for top left; (Bottom right) Histogram for top right. Note: aspect ratio has been modified for top left and top right images in order to fit into plots.

4.3 Note Identification

We create a *note matrix* based on the channel grid we generated. We first match each hole with a note on the channel grid. We consider a match to occur when a hole intersects with any note on the channel grid. Note that the size of a hole is wider than one pixel. Then we perform a y-projection of the hole to determine the note onset time. The note matrix M is of size 80-by-N, where N is the length of the scanned roll in pixels, and $M_{i,j} = 1$ if a punched hole covers the particular pixel. We then convert the sparse matrix into readable format, row by row, consisting of MIDI note number and note onset times. We scale the note onset time from pixels to milliseconds. We determine the scale factor according to the time information indicated at the beginning of each roll. For example, for a piece with tempo mark 70, the roll should be played at a speed of 7 feet per minute [5]. We thus define the scale factor F to be the length pixels divided by the time it takes to finish the piano roll (in seconds). In the case of the Welte-Mignon Format, $F = 455$ pixels per second.

For long notes, the punching system needs to punch multiple holes closely spaced along the channel grid. Thus to obtain the actual note durations, we define a minimum threshold between two holes as 11 pixels. This threshold was determined empirically based on observations of the reference piano roll. If the gap between holes is smaller than this minimum threshold, the holes are considered to belong to the same note.

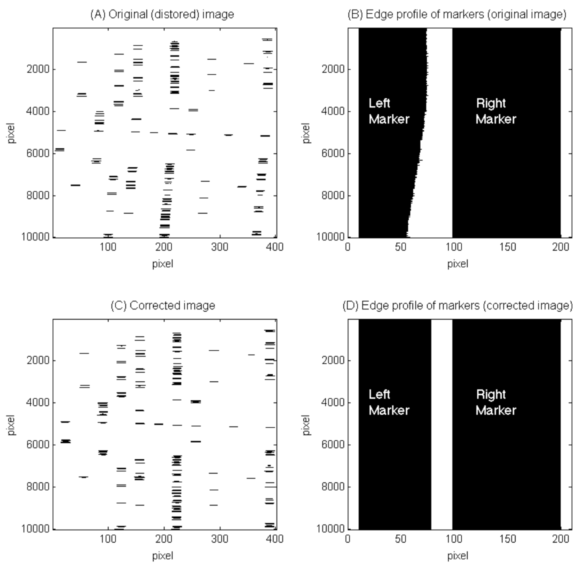


Figure 3. (A) Excerpt from the original (distorted) image (B) Edge profile of the left and right markers for the original image. (C) Excerpt from the corrected image (D) Edge profile of the left and right markers for the corrected image.

4.4 Distortion Correction

Due to instability in the scanner system, some scanned image were warped, as shown in Figure 3A. We fix this distortion by locating reference lines. On each roll, there are two straight bold lines marking the region of the punch holes. We refer to them as *markers*, as shown in Figure 1B. We first locate the two markers in the roll, and then use them as reference point to evaluate if the paper is warped or distorted. Our system iterates through each row and scale the pixels between the two markers such that the markers are enforced to be vertically straight and have a constant distance in between. Thus the imperfection in scanning can be detected and corrected, as in Figure 3C. Figure 3D shows straight left and right marker after the distortion correction.

4.5 Modeling Musical Expressions

Next we decode and model the expression markings on the left and right sides of the piano roll next to the markers. We obtain the location of each expression channel by constructing an expression channel grid similar to the note channel grid, as illustrated in Figure 4.

There are three types of musical expressions in Welte-Mignon piano rolls: constant velocity, changing velocity, and pedal information. The left zone of expression channels correspond to bass notes (notes below F♯4), and the zone on the right specifies the expression for treble notes (notes from G4) with the pedal information controlling the whole register.

For the constant-velocity controls, we simply map each indicated piano-key velocity to a chosen constant MIDI velocity, a seven-bit value between 0 and 127. Based on listening tests and calibration experiments described in the next section, we chose to map the *normal* (default) velocity to MIDI velocity 72, *mezzoforte* to 80, and *forzando* to 88. The piano samples used in this study is the Steinway Grand Piano in Logic Pro X [2].

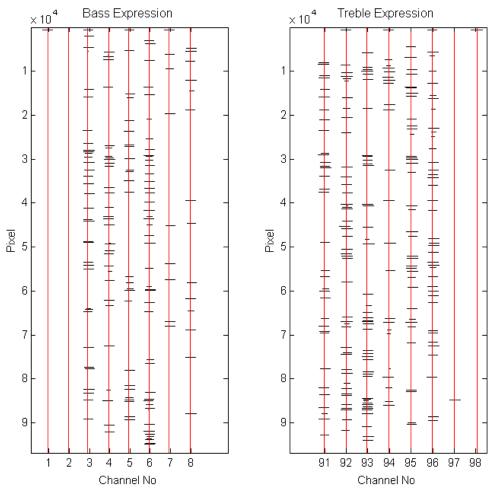


Figure 4. Musical expression holes in black with the expression channel grid in red vertical lines

To model the changing-velocity controls *crescendo* and *decrescendo*, we need some understanding of the expression pneumatics system itself.

In most Welte-Mignon systems, the expression is implemented using *pallet valves* [11]. Specifically, when the system reads a hole on the *crescendo* channel, the *crescendo valve* will be opened, introducing a suction to the expression pneumatic that pulls the pneumatic closed, producing a crescendo in the music. It takes significant time for the air to come into the pneumatic system to take effect. We model this process as an exponential approach to a target value, using one-pole unity-gain lowpass filter having impulse-response time-constant τ that is set to match the observed dynamics.

There are two types of crescendo:

1. a "very slow" crescendo produced by turning on the *crescendo* channel. This control is latching, so that one hole can turn it on.

2. a "fast" crescendo that is *not* latching.

A string of fast crescendo holes can used to speed up the slow crescendo, thereby providing many ultimate crescendo rates, as well as nonuniform crescendos. They are like little bursts of additional suction along the way as the slow crescendo develops.

Let the observed time-constant of the slow crescendo be denoted by τ_s (s for "slow"). Then the slow-crescendo one-pole filter has digital transfer function

$$H_s(z) = \frac{1 - p_s}{1 - p_s z^{-1}} \tag{1}$$

[2] https://www.apple.com/logic-pro/

where its pole p_s is defined as

$$p_s = e^{-T/\tau_s}$$

with T denoting the digital sampling interval in seconds (typically $T = 1/f_s$, where f_s denotes the sampling rate, and $f_s = 44100$ Hz or greater). The time-domain difference-equation used to implement the slow crescendo is given by

$$y_n = (1 - p_s)\, x_n + p_s\, y_{n-1}, \quad n = 0, 1, 2, \dots ,$$

where x_n is set to the target velocity at sample n, which is *forzando* for a crescendo, and either *normal* or *mezzoforte* for a decrescendo, depending on the last constant-velocity setting.

The fast crescendo is modeled exactly like the slow crescendo, but using a smaller time-constant $\tau_f \ll \tau_s$.

5. EVALUATION

We used MIDI file tools for Matlab [3] to convert our piano-roll matrix into MIDI format, and we synthesized the MIDI file in Logic Pro X using the Steinway Piano software-instrument that comes with Logic. We evaluated the success of our algorithm by comparing the synthesized audio to a recording of the player-piano, both driven from the same piano roll. Due to limited access to the machine and rolls, we only include one roll in our discussion here. However, given that each roll format possesses a fix template, we can assume that it will work for many additional rolls.

5.1 Recording Setup

We set up a recording environment for the player piano in a concert hall. The player we used is called a "push-up" because one pushes it up to a real piano where it plays the piano using padded wooden mechanical "fingers" (see Figure 5), as described further below. We recorded the push-up's performance on a 9' Steinway grand piano. The piano-roll chosen was the Chopin Op.42 Waltz in A♭, performed by Katherine Bacon, and published by Welte in 1924.

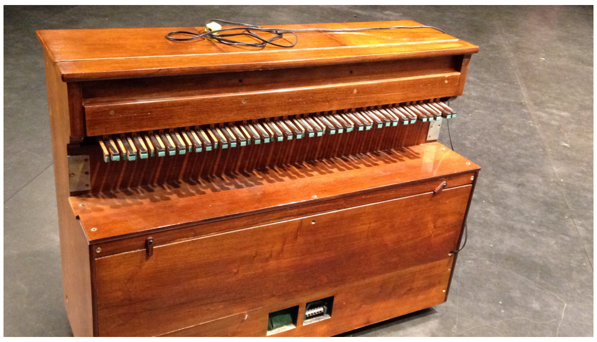


Figure 5. The Push-up Player

For the acoustic recording, we set up two cardioid microphones above the Steinway grand piano, one on the left,

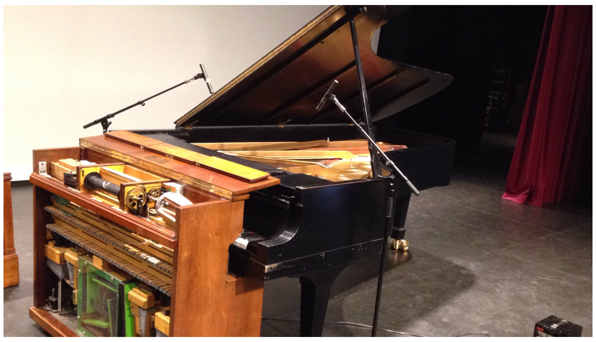


Figure 6. Acoustic Recording Setup of the player piano

capturing most of the energy from the bass, and one on the right, for the treble, as shown in Figure 6. The push-up player is aligned at the piano and the roll is set in the player, attaching the lead to the take up spool. The playback speed is set manually on the player. There can be some variation in playback on different players due to subtle differences in condition, however, most features of rolls are reproduced consistently on well functioning instruments. The instrument used in this project has been evaluated by player piano technicians to be in good working condition.

5.2 Note Similarity Measurement

An example MIDI file [4] is shown in Figure 7, with the raw image file on the top, and the MIDI file displayed in "piano roll" editor window in the program Logic Pro X for Mac OS X. We can see that the overall shape and trend of the notes are visually identical.

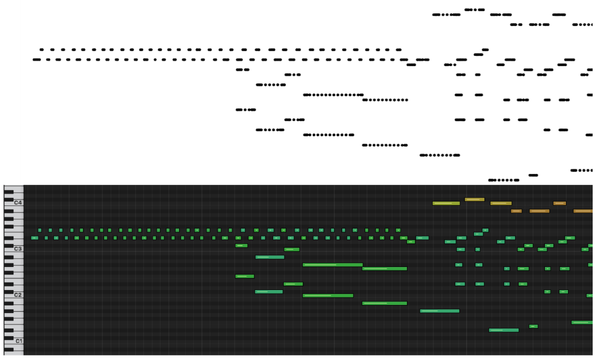


Figure 7. Visual inspection of the scanned image (top) and the "piano roll" image of the synthesized MIDI in Logic Pro X (bottom).

We measure the similarity of the audio content by calculating a similarity matrix [3] of the spectrogram between the MIDI-synthesized audio file and the audio recording. We then apply dynamic time warping [10] to align the MIDI file with audio and retrieve a path between the two. Figure 8 shows the similarity matrix comparing the spectrogram of the audio recording and MIDI-synthesized au-

[3] https://github.com/kts/matlab-midi

[4] MIDI synthesized audio file on Chopin Waltz Op.42 No.2 can be heard at https://tinyurl.com/kvyxc3s

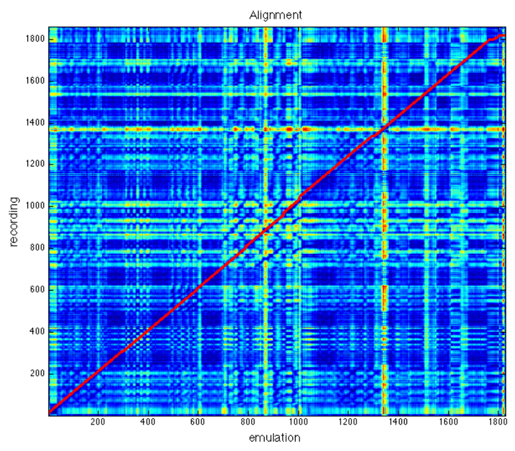


Figure 8. The spectrogram similarity matrix comparing the live audio recording and the synthesized MIDI, with an overlay of the time alignment line (red). Note: a straight diagonal line means perfect alignment.

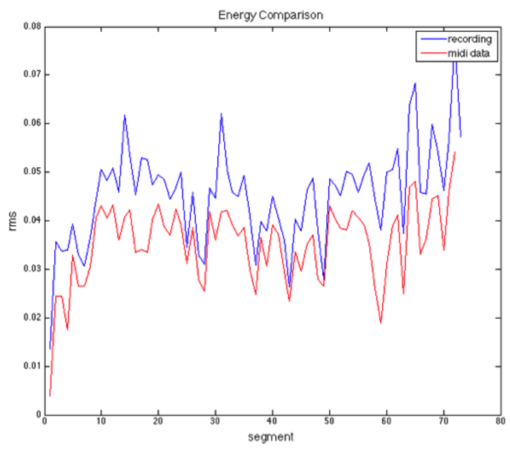


Figure 9. Comparison of the root mean square energy for live audio recording (blue) and synthesized MIDI (red).

dio, with the red diagonal line representing the time alignment between the recording and the MIDI. We see that the diagonal line is near straight and slope 1, meaning that we get almost perfect alignment between the audio and the MIDI. We found that there is some variation and curvature towards the end of the diagonal line and that the overall length of the MIDI emulation is 8-second longer than recording of the player piano.

5.3 Dynamics Measurement

To measure the success of modeling the dynamic expressions, we calculated the root mean square energy of both audio file. As in Figure 9, we see that their dynamic levels are similar, but we see that the original playback has a wider dynamic range than our synthesized MIDI file. We plan to optimize our setting of the dynamic variability to match the acoustic recording, but that will be the center (default) setting of a knob that can be varied from “flat” (no dynamics at all) to “exaggerated” (expanded dynamics). With this control users can adjust to taste. For example, it is nice to be able to make flatter renderings for noisy listening environments such as cars.

Note that every player piano will give a slightly different result due to variabilities in manufacturing and settings. This is another reason to make the end-result easily adjustable.

We do not yet include pedal information because the pedal on the player piano was not working perfectly at the time of our recording. We presently do not have control over how much pedal and pedal delay.

5.4 Discussion

We found that the MIDI file matched the original recording quite well, both visually in the graphs and audibly in recordings. As pointed out in the similarity measurement section, we observe that the MIDI roll is not quite at the same speed as the instrument playback of the roll. The MIDI roll appears to be “slowing down” towards the end compared to the original playback. Our research suggests that this is likely a deliberate design from the factory to compensate for increasing tempo change due to the changing diameter of the spool as it unwinds the paper upon playback. Wider spacing of the holes towards the end of the roll would keep the speed of the playback constant [1]. This would be consistent with our observation which finds the original roll playback faster than the MIDI of the paper roll itself. This observation will be explored with further evidence as more rolls are scanned and digitized.

6. CONCLUSION AND FUTURE WORK

We proposed methods for decoding reproducing piano roll images into MIDI files. We also proposed an apparently novel method for interpreting the expression markings on a piano roll. Though the system is designed to recognize the Welte-Mignon piano-roll format, our note matrix template is adaptable to all other systems with small modifications. However, the expression template is not adaptable from system to system. We also developed models for the Duo-art format. We have not yet evaluated this model with a playback comparison due to limited availability of an appropriate instrument. However, future work is planned for interpreting all piano-roll types, and including comparing different player instruments of the same type in order to measure variabilities. We further plan to create a master punch-matrix for the system types for correcting errors and repunching the piano rolls, thereby “restoring” them. We plan to develop a batch processing system for all the roll images created by the new scanning device that is presently being built for the Stanford Music Library [8]. Finally, we plan to release these digitized piano rolls on the Web as a free online resource.

7. ACKNOWLEDGEMENT

We'd like to thank Anthony Robinson and Jerry McBride for providing valuable information about reproducing piano rolls. We thank the Stanford Archive of Recorded Sound for providing the player piano as well as piano rolls.

8. REFERENCES

[1] Mechanical music digest archives. `http://www.mmdigest.com/Archives`. Accessed: 2017-04-18.

[2] Gustavo Colmenares, René Escalante, Juan F Sans, and Rina Surós. Computational modeling of reproducing-piano rolls. *Computer Music Journal*, 35(1):58–75, 2011.

[3] Jonathan Foote. Visualizing music and audio using self-similarity. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 77–80. ACM, 1999.

[4] Ichiro Fujinaga. *Optical music recognition using projections*. McGill University, 1990.

[5] The Pianola Institute. The reproducing piano - welte-mignon. `http://www.pianola.org/reproducing/reproducing_welte.cfm`. Accessed: 2017-04-15.

[6] Jagat Narain Kapur, Prasanna K Sahoo, and Andrew KC Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[7] Sasha Leitman. Trimpin: An interview. *Computer Music Journal*, 35(4):12–27, 2011.

[8] Stanford University Libraries. About the player piano project. `https://library.stanford.edu/projects/player-piano-project/about-project`. Accessed: 2017-05-16.

[9] Matteo Malosio, Flavio Pedrazzini, and Perego Niccol. The sisar project. *The Music Box*, 26(6):221–223, 2014.

[10] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[11] Arthur A Reblitz. *Player Piano: Servicing and Rebuilding*. Vestal Press, 1997.

[12] Anthony Robinson. Anthony's roll scanning. `http://semitone440.co.uk/scanner/`. Accessed: 2017-02-17.

[13] Larry Sitsky. *The Classical Reproducing Piano Roll: A Catelogue*. New York: Greenwood Press, 1990.

[14] Wayne Stahnke. Stahnke's roll archival methodology. `http://mmd.foxtail.com/Archives/Authors/Aut318.html`. Accessed: 2017-04-15.

[15] Warren Trachtman. Recovering inherent roll punch matrix spacing information and using it to dynamically self-correct roll scans. `http://www.trachtman.org/rollscans`. Accessed: 2017-03-15.

COMPARING OFFERTORY MELODIES OF FIVE MEDIEVAL CHRISTIAN CHANT TRADITIONS

Peter van Kranenburg
Utrecht University, Meertens Institute
peter.van.kranenburg@meertens.knaw.nl

Geert Maessen
independent scholar
gmaessen@xs4all.nl

ABSTRACT

In this study, we compare the melodies of five medieval chant traditions: Gregorian, Old Roman, Milanese, Beneventan, and Mozarabic. We present a newly created dataset containing several hundreds of offertory melodies, which are the longest and most complex within the total body of chant melodies. For each tradition, we train n-gram language models on a representation of the chants as sequence of chromatic intervals. By computing perplexities of the melodies, we get an indication of the relations between the traditions, revealing the melodies of the Gregorian tradition as most diverse. Next, we perform a classification experiment using global features of the melodies. The choice of features is informed by expert knowledge. We use properties of the intervallic content of the melodies, and properties of the melismas, revealing that significant differences exist between the traditions. For example, the Gregorian melodies contain less step-wise intervals compared to the other repertoires. Finally, we train a classifier on the perplexities as computed with the n-gram models, resulting in a very reliable classifier.

1. INTRODUCTION

In 789 Charlemagne ordained the Roman rite normative for Christian worship throughout his Empire. The chant of this rite became widely known as Gregorian chant (GRE). The earliest manuscripts with pitch-readable notation date from the beginning of the eleventh-century, increasing in number until the Renaissance. Manuscripts with neumatic contour notation go back to the end of the ninth century, and manuscripts with only the texts of the chants to almost 800. Basically all these manuscripts exhibit the same chants for specific liturgical occasions [13].

Since the invention of book printing and the Reformation, this uninterrupted and almost omnipresent European chant tradition came to an end. The Council of Trent (1545–1563) seems the beginning of many emended and sometimes drastically refashioned traditions of Gregorian chant. Since the restoration of Gregorian chant in the late nineteenth century, remnants of non-Gregorian chant traditions have continued to intrigue scholars. By the thirteenth century most of these traditions had already been abolished and replaced by Gregorian chant.

To this day the only surviving non-Gregorian tradition is the Milanese chant (MIL) of the Ambrosian rite in Northern Italy. The earliest notated manuscripts date from the twelfth century. Several hundreds of MIL chants are melodically related to GRE chants [2]. The Old Roman chant (ROM) that once existed in Rome itself is preserved in three graduals, several antiphoners and fragments from the eleventh till thirteenth centuries. Nearly all ROM chants are melodically related to GRE chants, with similar liturgical assignments. ROM was abolished in the thirteenth century [14]. Nearly 200 chants of the Beneventan rite of Southern Italy survive in eleventh and twelfth-century manuscripts among the regular GRE chants. Old Beneventan chant (BEN) was abolished in 1058 [15].

On the Iberian Peninsula and Southern France the Mozarabic rite was dominant from the sixth till the eleventh century. Its chant is called Old Hispanic chant. It was abolished in 1085 and replaced by the Roman rite with its GRE. Six parishes in Toledo were allowed to continue the tradition. The oral Mozarabic tradition was notated in early sixteenth century musical notation (MOZ). However, we also have over 5,000 Old Hispanic chants preserved in neumatic contour notation from the tenth till thirteenth centuries. Unfortunately, the vast majority of these chants do not correspond with MOZ and remain pitch unreadable [19, 27].

Since the 1950s the central question in chant scholarship concerned the relationship between GRE and ROM. Which of these traditions was the earliest? Was there perhaps another tradition preceding both? Many hypotheses have been put forward, but hardly any conclusive positions have been reached. Most scholars, however, believe that both GRE and ROM are later developments of the Roman tradition that was known to the Carolingians in the second half of the eight century. So the question became: Which was closer to eight century Rome, GRE or ROM? Some scholars believe the formulaic character of ROM to hold the earliest evidence, although the surviving manuscripts are of later date than the earliest GRE sources [7]. Some believe GRE reflects the earlier tradition, having adjusted the Roman chant only slightly to the specific needs of the Carolingian world [22]. Some still believe a third, Gallican or Hispanic, tradition played a major role in the creation of

tradition	century	chants	offertories	parts	avg. notes/part	std. notes/part
GRE: Gregorian Chant	XI–XII	1,000	115	344	162.54	62.65
ROM: Old Roman Chant	XI–XIII	700	94	285	170.14	69.04
MIL: Milanese Chant	XII–XIII	800	104	147	177.63	98.50
BEN: Beneventan chant	XI–XII	100	39	41	152.98	64.00
MOZ: Mozarabic chant	XI–XVI	400	71	139	127.94	52.74

Table 1. Estimation of date and number of mass proper chants in the main sources of five traditions, number of offertory-chants in our data set, number of offertory-parts, and average and standard deviation of the lengths of the parts.

the differences between GRE and ROM [17, 18]. A matter of debate also is the date when the chants were created. McKinnon [22] argues, primarily based on the liturgical assignment of the chant texts, that the Roman repertoire was composed according to a plan in the last decades of seventh-century Rome. Pfisterer [26] on the other hand argues, primarily based on the comparison of Latin Bible translations, that the repertoire has grown in accordance with the solemnity of the feasts between the fifth and early seventh centuries.

An important contribution to the discussion has been made by Rebecca Maloy in her 2010 monograph on the most complex of all chant genres in both traditions: the offertory [20]. Her book (including a digital edition of 94 cognate pairs of GRE and ROM offertories) provides a fascinating insight in modern scholarship and a highly sophisticated analysis of the offertory genre in both traditions. Basically, however, she does not reach conclusive arguments for a best hypothesis. In this paper, also, we do not pretend to present a conclusive position. Instead, we present the first results of a computational analysis of melodic similarities and differences between chant traditions, illustrating directions of research that may give new input to the longstanding questions. To this end we improved the musicological approach of traditional styles in terms of melismas, intervallic steps and leaps [13] to perplexities. We also verified and transformed Maloy's edition into a data set, and expanded the set with all offertories from the main sources of GRE, ROM, MIL, BEN and MOZ. Table 1 provides an overview with estimated numbers of mass proper chants in each tradition and the number of offertory chants included in our data set.

Important chant studies using computational techniques were published by several authors. However, most of the data are no longer available [10], represent only part of a single tradition [9], or a genre not easily available in five traditions [6, 12], or were not meant as exact data sets [30]. Some of the procedures used, however, need further investigation. Hansen's [10] distinction of different tonalities for different layers in GRE is one of these, as is the segmentation procedure used by Halperin [9] and Haas [8]. In fact this last approach can be seen as a precursor of the n-gram method we use in the current paper.

Maloy [20] does not use computational techniques, but she does with the offertory present a genre that is clearly available in five different traditions.

In this paper we demonstrate the importance of a computational approach for two longstanding and complementary questions in chant research. Based on local melodic structure, our n-gram method presents relations between different traditions (Section 3). Given a set of traditions, it shows which tradition has most characteristics in common with all (or most) traditions. This clearly relates to the musicological question of "origin". Based on global features of the chants, our decision-tree based classification method shows differences between the traditions, and is able to identify with high reliability the traditional "home" of single chants (Section 4). This can be helpful in identifying chants not corresponding to the catalogues in use.

2. DATA SET

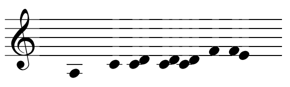
The contents of our data set is summarized in Table 1, showing the number of offertory chants included in our set. The first column lists codes and names of the separate traditions. The second and third columns give an estimate of period and number of the total preserved mass proper chants to which our offertories belong. In most cases, one offertory is divided in parts, the first part being the antiphon, and the subsequent parts the verses. Throughout this paper, we take the parts as basic units for analysis and classification. We include the number of parts per tradition in the table. We also include basic statistics on the length of chant-parts in number of notes.

For the GRE and ROM offertories we could have used the data set of Haas [8]. However, we preferred the critical edition of Maloy, because the Offertoriale Triplex [23] used by Haas is notably unreliable. One of the problems with the offertory concerns the many transpositions to avoid non-diatonical pitches. In selecting the best single manuscript for each separate chant Maloy chose, in our view, the best option. We converted Maloy's Finale scores to Volpiano strings and again carefully checked all details. We manually encoded the remaining offertories from the facsimile of Maloy's most important manuscript, Ben 34 [1] and one, GRE-115, *Audi Israhel*, from her book.

The Volpiano truetype font was developed by David Hiley and Fabian Weber at Regensburg University.[1] It is a typeface for note heads on the five line staff for monophonic music. It is perfectly suitable for our data set. It affords an encoding of each score as a string of characters. Characters `a` to `p` represent the notes A till a", while the

[1] Downloadable from: `http://www.uni-regensburg.de/Fakultaeten/phil_Fak_I/Musikwissenschaft/cantus/`

`i` represents the flat sign for b'. Small and capital `w`, `x`, `y` and `z` representing other alterations; some as defined in the font, some by new convention. Three dashes `---` separate the notes for the different words, two dashes `--`, for different syllables, and one dash marks a new neume within a syllable. Numerals indicate clefs and breaks. For clarification, Figure 1 shows an example of a Volpiano string and the rendering of it with the Volpiano font.



`1---a---c-cd-cdcd-f-fe---`

Figure 1. Example of a string in volpiano encoding and its rendering in the font.

We manually encoded the MIL, BEN and MOZ offertories from the best available sources; the Milanese mass book [29], the recent critical edition of Beneventan chant [16], and the facsimile of Mozarabic chant books [5].

3. COMPARING TRADITIONS USING N-GRAM MODELS

To examine the interconnections between the chant traditions concerning small-scale melodic fragments, we take an n-gram approach. n-gram modeling has been developed in computational linguistics [21]. It employs repetitive structures of a language to construct a probabilistic model allowing to compute the probability of occurrence of a word in its local context within a sentence. Concretely, let w be a word in vocabulary V belonging to a language L, and let $s = w_1, w_2, \ldots, w_l$ be a sentence of l words, also belonging to language L. Then, for a word w_i in s, an n-gram model allows to compute the conditional probability of w_i given the preceding context of $n-1$ words: $p(w_i|w_{i-(n-1)}, \ldots, w_{i-1})$. Previous application of n-gram modeling of music, notably include the IDyOM model [24], which combines long and short term models. For our purpose the basic n-gram approach suffices.

Typically, an n-gram model is derived from a large collection of training sentences belonging to the language of interest. In the most basic approach, for each unique context $w_{i-(n-1)}, \ldots, w_{i-1}$ in these training sentences, an inventory is made of all possible continuations w_i. This results in a distribution over the vocabulary, indicating the probability of each possible continuation of the context. The full model consists of the collection of distributions for the continuations of all unique contexts.

One of the uses of an n-gram model is to evaluate to what extent a given sentence fits in a given language. This is the way in which we employ n-gram models of the chant traditions. Since sentences are of variable length, it is not possible to simply compute the probability of a sentence as product of the probabilities of each word. Therefore, we will use the measure of *perplexity*, which indicates the degree to which the sentence 'fits' in the language:

$$PP = p(w_1, w_2, \ldots, w_l)^{-\frac{1}{l}}. \quad (1)$$

One particular problem in computing $p(w_i|w_{i-(n-1)}, \ldots, w_{i-1})$ occurs if the n-gram $w_{i-(n-1)}, \ldots, w_i$ has no occurrences in the training data. In that case, the probability of w_i is evaluated as zero, rendering the probability of the entire sentence zero. To circumvent this problem, several approaches exist. We use modified Kneser-Ney smoothing [4] as implemented in the KenLM Language Model Toolkit [11]. This method is widely accepted as the preferred method to deal with zero-counts.

3.1 Application on Chant Data

To derive an n-gram model from our chant data, we need to redefine some linguistic terms. We consider the traditions as languages. We consider each part of a chant as sentence, and we consider the intervals between the notes as words, where an interval is represented by a signed integer number indicating the direction (pos/neg) and the size of the interval in semi-tones. Because each of the chant traditions uses the same melodic intervals, the traditions have the same vocabulary, which allows us to compute the perplexity of a given chant for all five traditions. Also, since the vocabulary is very small compared to the vocabulary of any natural language, we need much less training data than typically is needed for natural language modeling.

3.2 Choosing *n*

An important question is which value to choose for n. For each $n \in \{2, 3, \ldots, 10\}$, and for each tradition we compute for each chant in that tradition the perplexity given its own tradition. To avoid overfitting, we follow a 10-fold evaluation, successively taking one subset of the data to compute the perplexities and taking the other 9 subsets for training, making sure that all parts of the same chant always are in the same subset. By visual inspection of the distributions of perplexities, we observe that for GRE, MIL, and MOZ, no further decrease of average perplexity is noticeable for $n > 5$, while for BEN and ROM slight improvement is achieved for respectively 7-gram and 8-gram models. Based on these findings, we choose $n = 5$ throughout this paper.

3.3 Comparing Chant Traditions using *n*-gram models

3.3.1 Method

As we are interested in the differences and commonalities of the five chant traditions, we perform an exhaustive evaluation in which we compute for each chant-part five perplexity values, one for each of the five traditions. In the case of the perplexity of a chant-part given its own tradition, we need to derive an n-gram model from all other chants of that tradition, excluding the chant that contains the chant-part. To include this chant in deriving the n-gram model would result in a too optimistic value for the perplexity. The chant must be 'unseen' by the model. The resulting perplexity reflects the extent to which the chant-part fits in its own tradition. For the perplexities given the other four traditions, we take n-gram models that have been derived from the entire sets of chants from those traditions. These resulting four perplexity values reflect to what extent the chant-part fits in the respective other four traditions.

After obtaining all perplexity values, we visualize the distributions for the various conditions as box-and-whisker plots [31]. The median is indicated with the red horizontal line. The box extends from the first to the third quartile, which is the interquartile range (IQR). The lower vertical whisker extends to the lowest data point still within 1.5 IQR from the first quartile and the upper whisker extends to the highest data point still within 1.5 IQR from the third quartile. The data points past the whiskers are considered outliers and are individually plotted as circles.

We evaluate whether two distributions differ significantly by performing a Kolmogorov-Smirnov test [28], and we evaluate the magnitude of the difference by computing the effect-size according to

$$e = \frac{\bar{x}_1 - \bar{x}_2}{\max(s_1, s_2)} \qquad (2)$$

in which x_1 and x_2 are the averages of the perplexities, and s_1 and s_2 are the standard deviations. By taking the max of s_1 and s_2, the resulting value for the effect size is a pessimistic estimation.

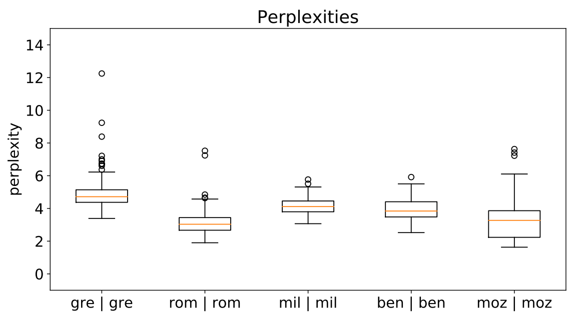


Figure 2. The distributions of perplexities of the chant-parts given their own respective tradition, represented as box-plots.

3.3.2 Results and Interpretation

The differences between the traditions are noticeable in Figure 2. The higher the perplexities, the higher the internal diversity of the repertoire. GRE is most divers. The outliers show specific chants of a single tradition most alien to this tradition. In GRE the two verses of GRE-63, *Oravi Deum meum,* are most extreme. This conforms to the fact that this is the most "chromatic" GRE chant. Its problematic pitches were already discussed by John of Afflighem (early twelfth century; [20]). The next GRE outlier is the antiphon of GRE-95, *Elegerunt apostoli*. *Oravi* and *Elegerunt* are considered two of only five offertories with possible Gallican origin, since they have cognate pairs in Old Hispanic chant. In ROM both antiphon and verse of ROM-92, *Domine Jesu Christe,* are most extreme. This conforms to the fact that this chant is almost identical to its GRE counterpart, GRE-92, *Domine Jesu Christe.* As Maloy demonstrates on textual evidence this chant in fact should be considered a GRE chant. As she puts it, "it is one of the few demonstrable instances of reverse, Frankish-to-Roman transmission in the offertory repertory" [20]. MIL and BEN hardly show outliers. However, in BEN the most extreme outlier, BEN-70, *Tunc imperator,* is the most syllabic chant of BEN. In MOZ, finally, the most extreme outlier is MOZ-13, *Offerte Domino,* the only MOZ chant in the fourth church mode.

Figure 3 shows the interrelations between the traditions. Comparing the five traditions to the five models we have 25 comparisons, resulting in 25 distributions of perplexity values. The Kolmogorov-Smirnov test for independence shows that only six out of the 300 possible pairs of distributions do not differ significantly ($p > 0.028$). Only 15 pairs of distributions have an effect size less than medium ($e < 0.5$). This indicates that the vast majority of the differences we see in the diagrams, are of significance.

Most striking is the top figure, showing the perplexities given the GRE model. The five box-plots there show that all five traditions are pretty close to the GRE model. BEN being most alien. However, compared to the four other figures, we see BEN being even more alien to ROM and MOZ. As is apparent from the diagrams, GRE gives the best overall model for all traditions. Second best is MIL. The worst model for all is ROM, followed by MOZ.

These findings can be related to the longstanding question about origin. Assuming that the process of oral transmission generally results in decreasing complexity, it is well conceivable that all traditions stem from GRE, while it seems inconceivable that ROM was the root of all.

4. CLASSIFICATION WITH GLOBAL FEATURES

4.1 Feature Set

We also examine the differences between the traditions in terms of a set of global features. A global feature summarizes the entire melody in one value. The feature set we use relates to earlier musicological approaches to characterize the traditions. There are two groups of features: features that describe the intervallic contents of a melody, and features that are related to the length of melismas. We measure the following features: the frequencies of occurrence of each of the melodic intervals from -12 to 12 semitones, where the sign indicates the direction; `aleaps`, `asteps`, `dleaps`, and `dsteps`, which measure the fraction of intervals that respectively are ascending leaps, ascending steps, descending leaps, and descending steps; `leaps` and `steps` are the fractions of leaps and steps disregarding direction; `unison` is the fraction of note repetitions; `melis1-1`, `melis2-2`, `melis3-5`, `melis6-10`, `melis11-20`,

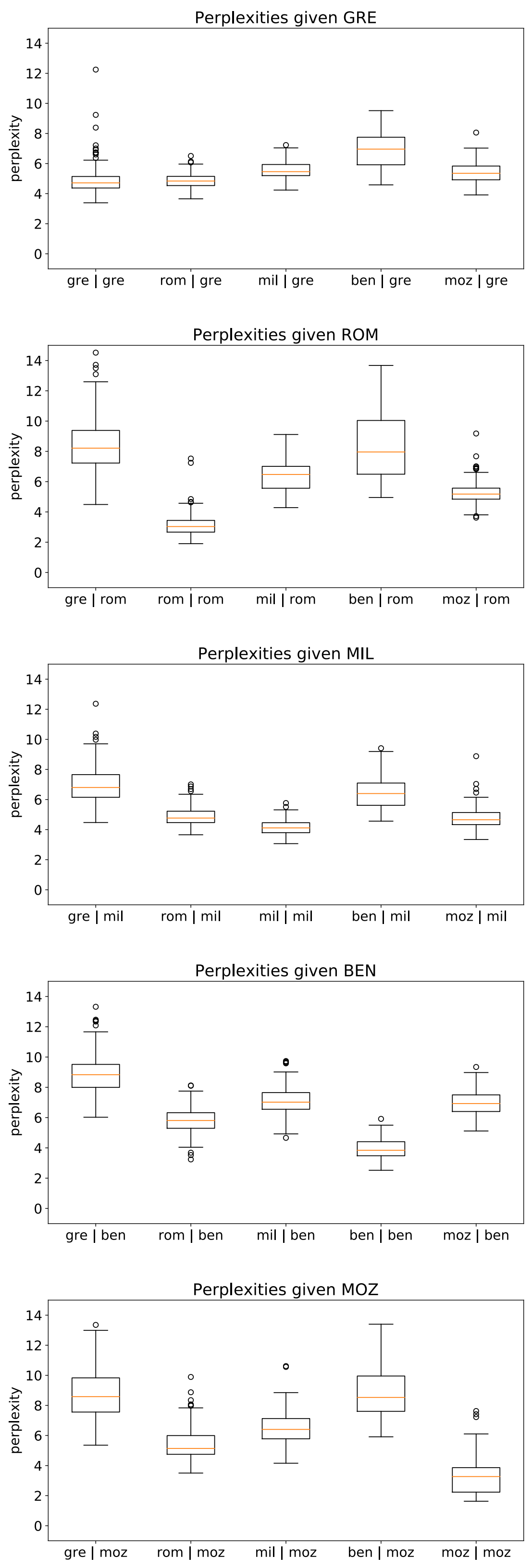


Figure 3. Distributions of the perplexities given the various traditions.

tradition	precision	recall	F1-score	support
BEN	0.38	0.22	0.28	41
GRE	0.90	0.89	0.89	344
MIL	0.64	0.65	0.65	147
MOZ	0.77	0.74	0.76	139
ROM	0.80	0.86	0.83	285
avg/total	0.79	0.79	0.79	956

Table 2. Classification results using the Decision Tree learner on the data set with global features.

`melis21-50`, `melis51-100`, `melis100-inf`, (`melisx-y` in general), represent the fraction of lyric syllables that have between x and y (y included) notes in the melody. `melis_mode` is the most common number of notes per syllable. `melis_longest`, `melis_secondlongest`, `melis_thirdlongest`, and `melis_fourthlongest` are the lengths of the four longest melismas. Finally, `melis_skewness`, and `melis_kurtosis` are the skewness and kurtosis of the distribution of melisma lengths.

We measure the values of these features in each of the 956 chant-parts. With the resulting dataset we perform a classification experiment to examine whether these features contain information for distinguishing between the five traditions.

4.2 Decision Tree Classification

Since we are not solely interested in classification accuracy, but we also want to understand the differences between the traditions, we prefer a learning algorithm that results in an interpretable model. Therefore, we learn a decision tree from our dataset with global features. We use the implementation of the tree learning algorithm as provided by the Python Scikit-learn library [25]. To prevent overfitting, and to obtain a relatively small tree, we set the minimum number of chant-parts per leaf to 10 and the maximum depth of the tree to 3.

To estimate the generalization of the learned tree, we perform 10-fold cross-validation, successively using one subset for testing and the other 9 subsets for learning a tree. For each chant in the current test-set, we record whether the classification was right. Again, we make sure to keep all parts from the same chant in either the test or the train set. After this procedure, we have a classification result for each of the chant-parts. Table 2 summarizes the resulting classification performance. While the overall-performance is not bad, discerning the chant-parts from BEN and MIL appears to be less successful.

There is no clear sign of overfitting. Therefore, we train a tree on the entire data set, which represents the differences between the traditions. The tree is depicted in Figure 4. It is apparent from the tree that the amount of step-wise motion in the melodies is one of the most important characteristics to isolate the GRE chants. These chants show the lowest amount of steps. Furthermore, the number of syllables with only one note, the amount of descending minor thirds, and the amount of descending minor seconds are of

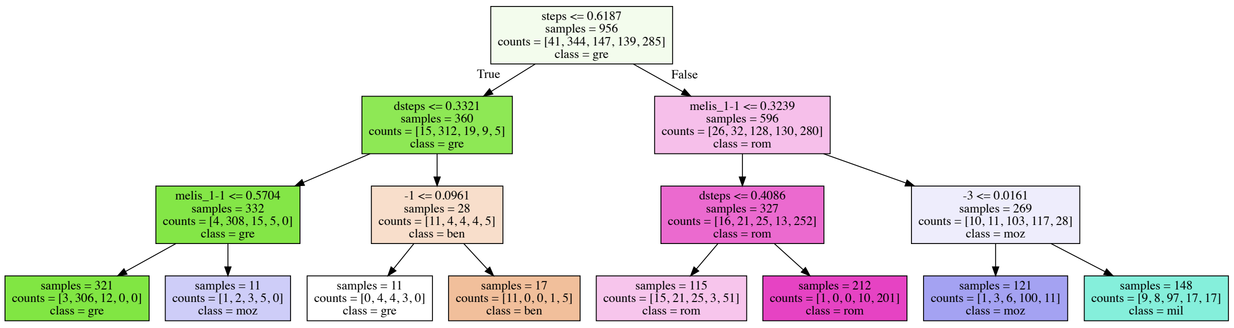


Figure 4. Decision tree as learned from the data set with global feature values. The order of the classes in the 'counts' field is: [BEN, GRE, MIL, MOZ, ROM]. The values indicate the number of chant-parts from the respective tradition that are 'in' the leave of tree.

tradition	precision	recall	F1-score	support
BEN	0.71	0.59	0.64	41
GRE	0.90	0.92	0.91	344
MIL	0.73	0.73	0.73	147
MOZ	0.91	0.88	0.90	139
ROM	0.93	0.94	0.94	285
avg/total	0.88	0.88	0.88	956

Table 3. Classification results using the Random Forest classifier on the data set with global features.

tradition	precision	recall	F1-score	support
BEN	0.93	0.98	0.95	41
GRE	0.97	0.98	0.98	344
MIL	0.96	0.95	0.96	147
MOZ	0.95	0.94	0.95	139
ROM	1.00	0.98	0.99	285
avg/total	0.97	0.97	0.97	956

Table 4. Classification results using the Random Forest classifier on the perplexity data.

importance. With just these features, it appears possible to separate the traditions to a moderately high degree.

4.3 Random Forest Classification

To examine whether it is possible to get higher classification accuracy, we also train a Random Forest Classifier, which trains a number of trees on random subsets of the data [3]. This does not lead to an easily interpretable model, but this procedure is known to typically show higher performance than a single decision tree. We set the number of trees to 10 and we follow the same procedure using 10-fold cross validation. The results are presented in Table 3. The results show significant improvement, but still with weaknesses for BEN and MIL.

4.4 Classification with Perplexity values

Since the perplexity of a chant-part given the n-gram model of a tradition also can be considered a global feature, we assemble another data set with for each chant-part the five perplexities for the five traditions, as computed in Section 3, as features. The classification results for a Random Forest Classifier are shown in Table 4.

Based on the perplexity values, we obtain a very accurate classifier with a F1-score as high as 0.97. Even for the minority class BEN we obtain very good results. Such a classifier can be of particular interest in tracing chants whose origins are unclear.

We performed an analysis of the chant-parts that are mis-classified by our best-performing classifier, the random forest trained on the perplexity data. Due to space constraints, it is not possible to give a full account of the analysis here, but in general we can state that many of the mis-classified parts are remarkable cases, including the outliers that have been discussed in Section 3.3.2, but also some other chants with debatable origin.

5. CONCLUSION AND FUTURE WORK

We presented an n-gram method to examine relations between medieval chant repertories, touching on central questions in chant scholarship. Our method shows in a quantitatively precise way that the body of Gregorian offertory melodies is characterized by a higher internal diversity than the offertories from the other four traditions. We also presented a highly accurate classification method. Outliers and misclassifications in both cases pointed at known problems in chant scholarship. Future work will concentrate on the refinement of our approaches for separate chant genres within traditions.

6. REFERENCES

[1] *Le Codex VI 34 de la Bibliothèque capitulaire de Bénévent, Graduel avec Prosaire & Tropaire*. Number XV in Paléographie Musicale. Abbaye Saint-Pierre de Solesmes, Solesmes, 1992.

[2] Terence Bailey. Ambrosian chant. `http://www.oxfordmusiconline.com/subscriber/article/grove/music/00754`.

[3] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[4] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13:359–394, 1999.

[5] Ángel Fernández Collado, Alfredo Rodríguez González, and Isidoro Castañeda Tordera, editors. *Los Cantorales Mozárabes de Cisneros, Catedral de Toledo*. Cabildo de la Catedral Primada, 2011.

[6] Ike de Loos. *Duitse en Nederlandse Muzieknotaties in de 12e en 13e eeuw*. PhD thesis, Universiteit Utrecht, Utrecht, 1996.

[7] Joseph Dyer. The roman offertory, an introduction and some hypotheses. In *The Offertory and its Verses: Research, Past, Present and Future*. Tapir Academic Press, Trondheim, 2007.

[8] Max Haas. *Mündliche Überlieferung und altrömischer Choral, Historische und analytische computergestützte Untersuchungen*. Peter Lang, Bern, 1997.

[9] David Halperin. *Contribution to a Morphology of Ambrosian Chant, A computer-aided analysis of the pars hiemalis according to the British Museum manuscript add. 34209 together with a package of computer programs for the analysis of monophonic music*. PhD thesis, Tel Aviv University, Tel Aviv, 1986.

[10] Finn Egeland Hansen. *The Grammar of Gregorian Tonality, An Investigation Based on the Repertory in Codex H 159, Montpellier*. Dan Fog Musikforlag, Copenhagen, 1979.

[11] Kenneth Heafield. Kenlm language model toolkit. `http://kheafield.com/code/kenlm/`.

[12] Katherine Eve Helsen. *The Great Responsories of the Divine Office, Aspects of Structure and Transmission*. PhD thesis, Universität Regensburg, Regensburg, 2008.

[13] David Hiley. *Gregorian Chant*. Cambridge University Press, Cambridge, 2009.

[14] Helmuth Hucke and Joseph Dyer. Old roman chant. `http://www.oxfordmusiconline.com/subscriber/article/grove/music/11725`.

[15] Thomas Forrest Kelly. Beneventan chant. `http://www.oxfordmusiconline.com/subscriber/article/grove/music/02678`.

[16] Thomas Forrest Kelly and Matthew Peattie, editors. *Monumenta Monodica Medii Aevi IX, The Music of the Beneventan Rite*. Bärenreiter, Kassel, 2016.

[17] Kenneth Levy. A new look at old roman chant. *Early Music History*, 19:81–104, 2000.

[18] Kenneth Levy. A new look at old roman chant - ii. *Early Music History*, 20:173–198, 2001.

[19] Geert Maessen and Peter Van Kranenburg. A semi-automatic method to produce singable melodies for the lost chant of the mozarabic rite. In *Proceedings of the 7th International Workshop on Folk Music Analysis*, Málaga, 2017.

[20] Rebecca Maloy. *Inside the Offertory, Aspects of Chronology and Transmission*. Oxford University Press, Oxford, 2010.

[21] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, 1999.

[22] James McKinnon. *The Advent Project, The Later-Seventh-Century Creation of the Roman Mass Proper*. University of California Press, Berkeley, 2001.

[23] Karl Ott and Rupert Fischer, editors. *Offertoriale Triplex cum Versiculis*. Abbaye Saint-Pierre de Solesmes, Solesmes, 1985.

[24] Marcus Thomas Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, City University, London, London, 2005.

[25] 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*, 12:2825–2830, 2011.

[26] Andreas Pfisterer. *Cantilena Romana, Untersuchungen zur Überlieferung des gregorianischen Chorals*. Ferdinand Schöningh, Paderborn, 2002.

[27] Don Michael Randel. Mozarabic chant. `http://www.oxfordmusiconline.com/subscriber/article/grove/music/19269`.

[28] Nikolaĭ Vasilevich Smirnov. Tables for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, 19(2):279–281, 1948.

[29] Gregorio Maria Suñol, editor. *Antiphonale Missarum juxta ritum Sanctae Ecclesiae Mediolanensis*. Desclée et Socii, Rome, 1935.

[30] Jessica Thompson, Andrew Hankinson, and Ichiro Fujinaga. Searching the liber usualis: using couchdb and elasticsearch to query graphical musical documents. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, Miami, 2011.

[31] John Wilder Tukey. *Exploratory Data Analysis*. Addison-Wesley, Readin, PA, 1977.

COUNTERPOINT BY CONVOLUTION

Cheng-Zhi Anna Huang*† **Tim Cooijmans***† **Adam Roberts**$^{\sharp}$
Aaron Courville† **Douglas Eck**$^{\sharp}$
* Equal contributions † MILA, Université de Montréal $^{\sharp}$ Google Brain
chengzhiannahuang@gmail.com, cooijmans.tim@umontreal.ca

ABSTRACT

Machine learning models of music typically break up the task of composition into a chronological process, composing a piece of music in a single pass from beginning to end. On the contrary, human composers write music in a nonlinear fashion, scribbling motifs here and there, often revisiting choices previously made. In order to better approximate this process, we train a convolutional neural network to complete partial musical scores, and explore the use of blocked Gibbs sampling as an analogue to rewriting. Neither the model nor the generative procedure are tied to a particular causal direction of composition.

Our model is an instance of orderless NADE [36], which allows more direct ancestral sampling. However, we find that Gibbs sampling greatly improves sample quality, which we demonstrate to be due to some conditional distributions being poorly modeled. Moreover, we show that even the cheap approximate blocked Gibbs procedure from [40] yields better samples than ancestral sampling, based on both log-likelihood and human evaluation.

1. INTRODUCTION

Counterpoint is the process of placing notes against notes to construct a polyphonic musical piece. [9] This is a challenging task, as each note has strong musical influences on its neighbors and notes beyond. Human composers have developed systems of rules to guide their compositional decisions. However, these rules sometimes contradict each other, and can fail to prevent their users from going down musical dead ends. Statistical models of music, which is our current focus, is one of the many computational approaches that can help composers try out ideas more quickly, thus reducing the cost of exploration [8].

Whereas previous work in statistical music modeling has relied mainly on sequence models such as Hidden Markov Models (HMMs [2]) and Recurrent Neural Networks (RNNs [31]), we instead employ convolutional neural networks due to their invariance properties and em-

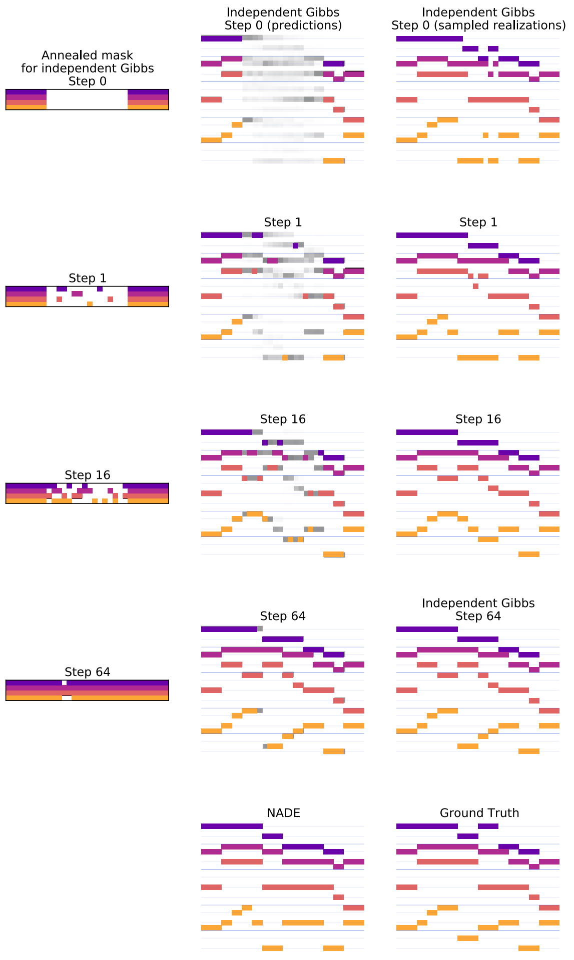


Figure 1. Blocked Gibbs inpainting of a corrupted Bach chorale by COCONET. At each step, a random subset of notes is removed, and the model is asked to infer their values. New values are sampled from the probability distribution put out by the model, and the process is repeated. Left: annealed masks show resampled variables. Colors distinguish the four voices. Middle: grayscale heatmaps show predictions $p(\mathbf{x}_j \mid \mathbf{x}_C)$ summed across instruments. Right: complete pianorolls after resampling the masked variables. Bottom: a sample from NADE (left) and the original Bach chorale fragment (right).

phasis on capturing local structure. Nevertheless, they have also been shown to successfully model large-scale structure [37, 38]. Moreover, convolutional neural networks have shown to be extremely versatile once trained, as demonstrated by a variety of creative uses such as DeepDream [29] and style transfer [10].

We introduce COCONET, a deep convolutional model trained to reconstruct partial scores. Once trained, COCONET provides direct access to all conditionals of the form $p(\mathbf{x}_i \mid \mathbf{x}_C)$ where C selects a fragment of a musical score $\mathbf{x}$ and $i \notin C$ is in its complement. COCONET is an instance of deep orderless NADE [36], which learns an ensemble of factorizations of the joint $p(\mathbf{x})$, each corresponding to a different ordering. A related approach is the multi-prediction training of deep Boltzmann machines (MP-DBM) [12], which also gives a model that can predict any subset of variables given its complement.

However, the sampling procedure for orderless NADE treats the ensemble as a mixture and relies heavily on ordering. Sampling from an orderless NADE involves (randomly) choosing an ordering, and sampling variables one by one according to the chosen ordering. This process is called *ancestral sampling*, as the order of sampling follows the directed structure of the model. We have found that this produces poor results for the highly structured and complex domain of musical counterpoint.

Instead, we propose to use blocked-Gibbs sampling, a Markov Chain Monte Carlo method to sample from a joint probability distribution by repeatedly resampling subsets of variables using conditional distributions derived from the joint probability distribution. An instance of this was previously explored by [40] who employed a NADE in the transition operator for a Markov Chain, yielding a Generative Stochastic Network (GSN). The transition consists of a corruption process that masks out a subset $\mathbf{x}_{\neg C}$ of variables, followed by a process that independently resamples variables $\mathbf{x}_i$ (with $i \notin C$) according to the distribution $p_\theta(\mathbf{x}_i \mid \mathbf{x}_C)$ emitted by the model with parameters θ. Crucially, the effects of independent sampling are amortized by annealing the probability with which variables are masked out. Whereas [40] treat their procedure as a cheap approximation to ancestral sampling, we find that it produces superior samples. Intuitively, the resampling process allows the model to iteratively *rewrite* the score, giving it the opportunity to correct its own mistakes.

COCONET addresses the general task of completing partial scores; special cases of this task include "bridging" two musical fragments, and temporal upsampling and extrapolation. Figure 1 shows an example of COCONET populating a partial piano roll using blocked-Gibbs sampling. Code and samples are publically available. [1] Our samples on a variety of generative tasks such as rewriting, melodic harmonization and unconditioned polyphonic music generation show the versatility of our model. In this work we focus on Bach chorales, and assume four voices are active at all times. However, our model can be easily adapted to the more general, arbitrarily polyphonic representation as used in [4].

Section 2 discusses related work in modeling music composition, with a focus on counterpoint. The details of our model and training procedure are laid out in Section 3. We discuss evaluation under the model in Section 4, and sampling from the model in Section 5. Results of quantitative and qualitative evaluations are reported in Section 6.

2. RELATED WORK

Computers have been used since their early days for experiments in music composition. A notable composition is Hiller and Issacson's string quartet Illiac Suite [18], which experiments with statistical sequence models such as Markov chains. One challenge in adapting such models is that music consists of multiple interdependent streams of events. Compare this to typical sequence domains such as speech and language, which involve modeling a single stream of events: a single speaker or a single stream of words. In music, extensive theories in counterpoint have been developed to address the challenge of composing multiple streams of notes that coordinate. One notable theory is due to Fux [9] from the Baroque period, which introduces *species counterpoint* as a pedagogical scheme to gradually introduce students to the complexity of counterpoint. In first species counterpoint only one note is composed against every note in a given fixed melody (*cantus firmus*), with all notes bearing equal durations and the resulting vertical intervals consisting of only consonances.

Computer music researchers have taken inspiration from this pedagogical scheme by first teaching computers to write species counterpoint as opposed to full-fledged counterpoint. Farbood [7] uses Markov chains to capture transition probabilities of different melodic and harmonic transitions rules. Herremans [16, 17] takes an optimization approach by writing down an objective function that consists of existing rules of counterpoint and using a variable neighbourhood search (VNS) algorithm to optimize it.

J.S. Bach chorales has been the main corpus in computer music that serves as a starting point to tackle full-fledged counterpoint. A wide range of approaches have been used to generate music in the style of Bach chorales, for example rule-based and instance-based approaches such as Cope's recombinancy method [6]. This method involves first segmenting existing Bach chorales into smaller chunks based on music theory, analyzing their function and stylistic signatures and then re-concatenating the chunks into new coherent works. Other approaches range from constraint-based [30] to statistical methods [5]. In addition, [8] gives a comprehensive survey of AI methods used not just for generating Bach chorales, but also algorithmic composition in general.

Sequence models such as HMMs and RNNs are natural choices for modeling music. Successful application of such models to polyphonic music often requires serializing or otherwise re-representing the music to fit the sequence paradigm. For instance, Liang in BachBot [27] serializes four-part Bach chorales by interleaving the parts,

[1]
Code:	`https://github.com/czhuang/coconet`
Data:	`https://github.com/czhuang/JSB-Chorales-dataset`
Samples:	`https://coconets.github.io/`

while Allan and Williams [1] construct a chord vocabulary. Boulanger et al. [4] adopt a piano roll representation, a binary matrix $\mathbf{x}$ where $\mathbf{x}_{it} = 1$ iff some instrument is playing pitch i at time t. To model the joint probability distribution of the multi-hot pitch vector $\mathbf{x}_t$, they employ a Restricted Boltzmann Machine (RBM [19, 32]) or Neural Autoregressive Distribution Estimator (NADE [25]) at each time step. Similarly Goel et al. [11] employ a Deep Belief Network [19] on top of an RNN.

Hadjeres et al. [14] instead employ an undirected Markov model to learn pairwise relationships between neighboring notes up to a specified number of steps away in a score. Sampling involves Markov Chain Monte Carlo (MCMC) using the model as a Metropolis-Hastings (MH) objective. The model permits constraints on the state space to support tasks such as melody harmonization. However, the Markov assumption can limit the expressivity of the model.

Hadjeres and Pachet in DeepBach [13] model note predictions by breaking down its full context into three parts, with the past and the future modeled by stacked LSTMs going in the forward and backward directions respectively, and the present harmonic context modeled by a third neural network. The three are then combined by a fourth neural network and used in Gibbs sampling for generation.

Lattner et al. imposes higher-level structure by interleaving selective Gibbs sampling on a convolutional RBM [26] and gradient descent that minimizes cost to template piece on features such as self-similarity. This procedure itself is wrapped in simulated annealing to ensure steps do not lower the solution quality too much.

We opt for an orderless NADE training procedure which enables us to train a mixture of all possible directed models simultaneously. Finally, an approximate blocked Gibbs sampling procedure [40] allows fast generation from the model.

3. MODEL

We employ machine learning techniques to obtain a generative model of musical counterpoint in the form of piano rolls. Given a dataset of observed musical pieces $\mathbf{x}^{(1)} \dots \mathbf{x}^{(n)}$ posited to come from some true distribution $p(\mathbf{x})$, we introduce a model $p_\theta(\mathbf{x})$ with parameters θ. In order to make $p_\theta(\mathbf{x})$ close to $p(\mathbf{x})$, we maximize the data log-likelihood $\sum_i \log p_\theta(\mathbf{x}^{(i)})$ (an approximation of $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \log p_\theta(\mathbf{x})$) by stochastic gradient descent.

The joint distribution $p(\mathbf{x})$ over D variables $\mathbf{x}_1 \dots \mathbf{x}_D$ is often difficult to model directly and hence we construct our model $p_\theta(\mathbf{x})$ from simpler factors. In the NADE [25] framework, the joint $p_\theta(\mathbf{x})$ is factorized autoregressively, one variable at a time, according to some ordering $o = o_1 \dots o_D$, such that

$$p_\theta(\mathbf{x}) = \prod_d p_\theta(\mathbf{x}_{o_d} \mid \mathbf{x}_{o_{<d}}). \tag{1}$$

For example, it can be factorized in chronological order:

$$p_\theta(\mathbf{x}) = p_\theta(\mathbf{x}_1)p_\theta(\mathbf{x}_2|\mathbf{x}_1)\dots p_\theta(\mathbf{x}_D|\mathbf{x}_{D-1}\dots\mathbf{x}_1) \tag{2}$$

In general, NADE permits any one fixed ordering, and although all orderings are equivalent from a theoretical perspective, they differ in practice due to effects of optimization and approximation.

Instead, we can train NADE for all orderings o simultaneously using the orderless NADE [36] training procedure. This procedure relies on the observation that, thanks to parameter sharing, computing $p_\theta(\mathbf{x}_{o_{d'}} \mid \mathbf{x}_{o_{<d}})$ for all $d' \geq d$ is no more expensive than computing it only for $d' = d$.[2] Hence for a given o and d we can simultaneously obtain partial losses for all orderings that agree with o up to d:

$$\mathcal{L}(\mathbf{x}; o_{<d}, \theta) = -\sum_{o_d} \log p_\theta(\mathbf{x}_{o_d} \mid \mathbf{x}_{o_{<d}}, o_{<d}, o_d) \tag{3}$$

An orderless NADE model offers direct access to all distributions of the form $p_\theta(\mathbf{x}_i|\mathbf{x}_C)$ conditioned on any set of contextual variables $\mathbf{x}_C = \mathbf{x}_{o_{<d}}$ that might already be known. This gives us a very flexible generative model; in particular, we can use these conditional distributions to complete arbitrarily partial musical scores.

To train the model, we sample a training example $\mathbf{x}$ and context C such that $|C| \sim U(1, D)$, and update θ based on the gradient of the loss given by Equation 3. This loss consists of $D - d + 1$ terms, each of which corresponds to one ordering. To ensure all orderings are trained evenly we must reweight the gradients by $1/(D - d + 1)$. This correction, due to [36], ensures consistent estimation of the joint negative log-likelihood $\log p_\theta(\mathbf{x})$.

In this work, the model $p_\theta(x)$ is implemented by a deep convolutional neural network [23]. This choice is motivated by the locality of contrapuntal rules and their near-invariance to translation, both in time and in pitch space.

We represent the music as a stack of piano rolls encoded in a binary three-dimensional tensor $\mathbf{x} \in \{0,1\}^{I\times T\times P}$. Here I denotes the number of instruments, T the number of time steps, P the number of pitches, and $\mathbf{x}_{i,t,p} = 1$ iff the ith instrument plays pitch p at time t. We will assume each instrument plays exactly one pitch at a time, that is, $\sum_p \mathbf{x}_{i,t,p} = 1$ for all i, t.

Our focus is on four-part Bach chorales as used in prior work [1,4,11,14,27]. Hence we assume $I = 4$ throughout. We constrain ourselves to only the range that appears in our training data (MIDI pitches 36 through 88). Time is discretized at the level of 16th notes for similar reasons. To curb memory requirements, we enforce $T = 128$ by randomly cropping the training examples.

Given a training example $\mathbf{x} \sim p(\mathbf{x})$, we present the model with values of only a strict subset of its elements $\mathbf{x}_C = \{\mathbf{x}_{(i,t)} \mid (i,t) \in C\}$ and ask it to reconstruct its complement $\mathbf{x}_{\neg C}$. The input $\mathbf{h}^0 \in \{0,1\}^{2I\times T\times P}$ is obtained by masking the piano rolls $\mathbf{x}$ to obtain the context $\mathbf{x}_C$ and concatenating this with the corresponding mask:

$$\mathbf{h}^0_{i,t,p} = \mathbb{1}_{(i,t)\in C}\mathbf{x}_{i,t,p} \tag{4}$$

$$\mathbf{h}^0_{I+i,t,p} = \mathbb{1}_{(i,t)\in C} \tag{5}$$

[2] Here $\mathbf{x}_{o_{<d}}$ is used as shorthand for variables $\mathbf{x}_{o_1} \dots \mathbf{x}_{o_{d-1}}$ that occur earlier in the ordering.

where the time and pitch dimensions are treated as spatial dimensions to convolve over. Each instrument's piano roll $\mathbf{h}_i^0$ and mask $\mathbf{h}_{\mathbf{I+i}}^{\mathbf{0}}$ is treated as a separate channel and convolved independently.

With the exception of the first and final layers, all convolutions preserve the size of the hidden representation. That is, we use "same" padding throughout and all activations have the same number of channels H, such that $\mathbf{h}^l \in \mathbb{R}^{H \times T \times P}$ for all $1 < l < L$. Throughout our experiments we used $L = 64$ layers and $H = 128$ channels. After each convolution we apply batch normalization [21] (denoted by $\mathrm{BN}(\cdot)$) with statistics tied across time and pitch. Batch normalization rescales activations at each layer to have mean β and standard deviation γ, which greatly improves optimization. After every second convolution, we introduce a skip connection from the hidden state two levels below to reap the benefits of residual learning [15].

$$\mathbf{a}^l = \mathrm{BN}(\mathbf{W}^l * \mathbf{h}^{l-1}; \gamma^l, \beta^l) \tag{6}$$

$$\mathbf{h}^l = \begin{cases} \mathrm{ReLU}(\mathbf{a}^l + \mathbf{h}^{l-2}) \\ \qquad \text{if } 3 < l < L-1 \text{ and } l \bmod 2 = 0 \\ \mathrm{ReLU}(\mathbf{a}^l) \text{ otherwise} \end{cases}$$

$$\mathbf{h}^L = \mathbf{a}^L \tag{7}$$

The final activations $\mathbf{h}^L \in \mathbb{R}^{I \times T \times P}$ are passed through the softmax function to obtain predictions for the pitch at each instrument/time pair:

$$p_\theta(\mathbf{x}_{i,t,p} \mid \mathbf{x}_C, C) = \frac{\exp(h^L_{i,t,p})}{\sum_p \exp(h^L_{i,t,p})} \tag{8}$$

The loss function from Equation 3 is then given by

$$\begin{aligned} \mathcal{L}(\mathbf{x}; C, \theta) &= -\sum_{(i,t) \notin C} \log p_\theta(\mathbf{x}_{i,t} \mid \mathbf{x}_C, C) \qquad (9) \\ &= -\sum_{(i,t) \notin C} \sum_p \mathbf{x}_{i,t,p} \log p_\theta(\mathbf{x}_{i,t,p} \mid \mathbf{x}_C, C) \end{aligned}$$

where p_θ denotes the probability under the model with parameters $\theta = \mathbf{W}^1, \gamma^1, \beta^1, \ldots, \mathbf{W}^{L-1}, \gamma^{L-1}, \beta^{L-1}$. We train the model by minimizing

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \mathbb{E}_{C \sim p(C)} \frac{1}{|\neg C|} \mathcal{L}(\mathbf{x}; C, \theta) \tag{10}$$

with respect to θ using stochastic gradient descent with step size determined by Adam [22]. The expectations are estimated by sampling piano rolls $\mathbf{x}$ from the training set and drawing a single context C per sample.

4. EVALUATION

The log-likelihood of a given example is computed according to Algorithm 1 by repeated application of Equation 8. Evaluation occurs one frame at a time, within which the model conditions on its own predictions and does not see the ground truth. Unlike notewise teacher-forcing, where the ground truth is injected after each prediction, the framewise evaluation is thus sensitive to accumulation of error. This gives a more representative measure of quality of the generative model. For each example, we repeat the evaluation process a number of times to average over multiple orderings, and finally average across frames and examples. For chronological evaluation, we draw only orderings that have the t_ls in increasing order.

Algorithm 1 Framewise log-likelihood evaluation

```
Given a piano roll $\mathbf{x}$
$L_{m,i,t} \leftarrow 0$ for all $m, i, t$
**for** multiple orderings $m = 0 \ldots M$ **do**
    $C \leftarrow \emptyset, \widehat{\mathbf{x}} \leftarrow \mathbf{x}$
    Sample an ordering $t_1, t_2 \ldots t_T$ over frames
    **for** $l = 0 \ldots T$ **do**
        Sample an ordering $i_1, i_2 \ldots i_I$ over instruments
        **for** $k = 0 \ldots I$ **do**
            $\pi_p \leftarrow p_\theta(\mathbf{x}_{i_k,t_l,p} \mid \widehat{\mathbf{x}}_C, C)$ for all $p$
            $L_{m,i_k,t_l} \leftarrow \sum_p \mathbf{x}_{i_k,t_l,p} \log \pi_p$
            $\widehat{\mathbf{x}}_{i_k,t_l} \sim \mathrm{Cat}(P, \pi)$
            $C \leftarrow C \cup (i_k, t_l)$
        **end for**
        $\widehat{\mathbf{x}}_C \leftarrow \mathbf{x}_C$
    **end for**
**end for**
**return** $-\frac{1}{T} \sum_t \log \frac{1}{M} \sum_m \exp \sum_i L_{m,i,t}$
```

5. SAMPLING

We can sample from the model using the orderless NADE ancestral sampling procedure, in which we first sample an ordering and then sample variables one by one according to the ordering. However, we find that this yields poor samples, and we propose instead to use Gibbs sampling.

5.1 Orderless NADE Sampling

Sampling according to orderless NADE involves first randomly choosing an ordering and then sampling variables one by one according to the chosen ordering. We use an equivalent procedure in which we arrive at a random ordering by at each step randomly choosing the next variable to sample. We start with an empty (zero everywhere) piano roll $\mathbf{x}^0$ and empty context C^0 and populate them iteratively by the following process. We feed the piano roll $\mathbf{x}^s$ and context C^s into the model to obtain a set of categorical distributions $p_\theta(\mathbf{x}_{i,t} | \mathbf{x}^s_{C^s}, C^s)$ for $(i,t) \notin C^s$. As the $\mathbf{x}_{i,t}$ are not conditionally independent, we cannot simply sample from these distributions independently. However, if we sample from one of them, we can compute new conditional distributions for the others. Hence we randomly choose one $(i,t)^{s+1} \notin C^s$ to sample from, and let $\mathbf{x}^{s+1}_{i,t}$ equal the one-hot realization. Augment the context with $C^{s+1} = C^s \cup (i,t)^{s+1}$ and repeat until the piano roll is populated. This procedure is easily generalized to tasks such as melody harmonization and partial score completion by starting with a nonempty piano roll.

Unfortunately, samples thus generated are of low quality, which we surmise is due to accumulation of errors.

Model	**Temporal resolution**		
	quarter	eighth	sixteenth
NADE [4]	7.19		
RNN-RBM [4]	6.27		
RNN-NADE [4]	5.56		
RNN-NADE (our implementation)	5.03	3.78	2.05
COCONET (chronological)	7.79 ± 0.09	4.21 ± 0.05	2.22 ± 0.03
COCONET (random)	5.03 ± 0.06	1.84 ± 0.02	0.57 ± 0.01

Table 1. Framewise negative log-likelihoods (NLLs) on the Bach corpus. We compare against [4], who used quarter-note resolution. We also compare on higher temporal resolutions (eighth notes, sixteenth notes), against our own reimplementation of RNN-NADE. COCONET is an instance of orderless NADE, and as such we evaluate it on random orderings. However, the baselines support only chronological frame ordering, and hence we evaluate our model in this setting as well.

This is a well-known weakness of autoregressive models. [3, 20, 24, 39] While the model provides conditionals $p_\theta(\mathbf{x}_{i,t}|\mathbf{x}_C, C)$ for all $(i,t) \notin C$, some of these conditionals may be better modeled than others. We suspect in particular those conditionals used early on in the procedure, for which the context C consists of very few variables. Moreover, although the model is trained to be order-agnostic, different orderings invoke different distributions, which is another indication that some conditionals are poorly learned. We test this hypothesis in Section 6.2.

5.2 Gibbs Sampling

To remedy this, we allow the model to revisit its choices: we repeatedly mask out some part of the piano roll and then repopulate it. This is a form of blocked Gibbs sampling [28]. Blocked sampling is crucial for mixing, as the high temporal resolution of our representation causes strong correlations between consecutive notes. For instance, without blocked sampling, it would take many steps to snap out of a long-held note. Similar considerations hold for the Ising model from statistical mechanics, leading to the Swendsen-Wang algorithm [33] in which large clusters of variables are resampled at once.

We consider two strategies for resampling a given block of variables: *ancestral* sampling and *independent* sampling. Ancestral sampling invokes the orderless NADE sampling procedure described in Section 5.1 on the masked-out portion of the piano roll. Independent sampling simply treats the masked-out variables $\mathbf{x}_{\neg C}$ as independent given the context $\mathbf{x}_C$.

Using independent blocked Gibbs to sample from a NADE model has been studied by [40], who propose to use an annealed masking probability $\alpha_n = \max(\alpha_{\min}, \alpha_{\max} - n(\alpha_{\max} - \alpha_{\min})/(\eta N))$ for some minimum and maximum probabilities $\alpha_{\min}, \alpha_{\max}$, total number of Gibbs steps N and fraction η of time spent before settling onto the minimum probability $\alpha_{\min}$. Initially, when the masking probability is high, the chain mixes fast but samples are poor due to independent sampling. As α_n decreases, the blocked Gibbs process with independent resampling approaches standard Gibbs where one variable at a time is resampled, thus amortizing the effects of independent sampling. N is a hyperparameter which as a rule of thumb we set equal to IT; it can be set lower than that to save computation at a slight loss of sample quality.

[40] treat independent blocked Gibbs as a cheap approximation to ancestral sampling. Whereas plain ancestral sampling (5.1) requires $O(IT)$ model evaluations, ancestral blocked Gibbs requires a prohibitive $O(ITN)$ model evaluations and independent Gibbs requires only $O(N)$, where N can be chosen to be less than IT. Moreover, we find that independent blocked Gibbs sampling in fact yields *better* samples than plain ancestral sampling.

6. EXPERIMENTS

We evaluate our approach on a popular corpus of four-part Bach chorales. While the literature features many variants of this dataset [1, 4, 14, 27], we report results on that used by [4]. As the quarter-note temporal resolution used by [4] is frankly too coarse to accurately convey counterpoint, we also evaluate on eighth-note and sixteenth-note quantizations of the same data.

It should be noted that quantitative evaluation of generative models is fundamentally hard [34]. The gold standard for evaluation is qualitative comparison by humans, and we therefore report human evaluation results as well.

6.1 Data Log-likelihood

Table 4 compares the framewise log-likelihood of the test data under variants of our model and those reported in [4]. We find that the temporal resolution has a dramatic influence on the performance, which we suspect is an artifact of the performance metric. The log-likelihood is evaluated by teacher-forcing, that is, the prediction of a frame is conditioned on the ground truth of all previously predicted frames. As temporal resolution increases, chord changes become increasingly rare, and the model is increasingly rewarded for simply holding notes over time.

We evaluate COCONET on both chronological and random orderings, in both cases averaging likelihoods across an ensemble of $M = 5$ orderings. The chronological orderings differ only in the ordering of instruments within each frame. We see in Table 4 that fully random orderings lead to significantly better performance. We believe the members of the more diverse random ensemble are more mutually complementary. For example, a forward ordering is uncertain at the beginning of a piece and more certain toward the end, whereas a backward ordering is more certain at the beginning and less certain toward the end.

6.2 Sample Quality

In Section 5 we conjectured that the low quality of NADE samples is due to poorly modeled conditionals $p_\theta(\mathbf{x}_{i,t} \mid \mathbf{x}_C, C)$ where C is small. We test this hypothesis by evaluating the likelihood under the model of samples generated by the ancestral blocked Gibbs procedure with C chosen according to independent Bernoulli variables. When we set the inclusion probability ρ to 0, we obtain NADE. Increasing ρ increases the expected context size $|C|$, which should yield better samples if our hypothesis is true. The results shown in Table 6.2 confirm that this is the case. For these experiments, we used sample length $T = 32$ time steps and number of Gibbs steps $N = 100$.

Sampling scheme	Framewise NLL
Ancestral Gibbs, $\rho = 0.00$ (NADE)	1.09 ± 0.06
Ancestral Gibbs, $\rho = 0.05$	1.08 ± 0.06
Ancestral Gibbs, $\rho = 0.10$	0.97 ± 0.05
Ancestral Gibbs, $\rho = 0.25$	0.80 ± 0.04
Ancestral Gibbs, $\rho = 0.50$	0.74 ± 0.04
Independent Gibbs [40]	0.52 ± 0.01

Table 2. Mean (± SEM) NLL under model of unconditioned samples generated from model by various schemes.

Figure 2 shows the convergence behavior of the various Gibbs procedures, averaged over 100 runs. We see that for low values of ρ (small C), the chains hardly make progress beyond NADE in terms of likelihood. Higher values of ρ (large C) enable the model to get off the ground and reach significantly better likelihood.

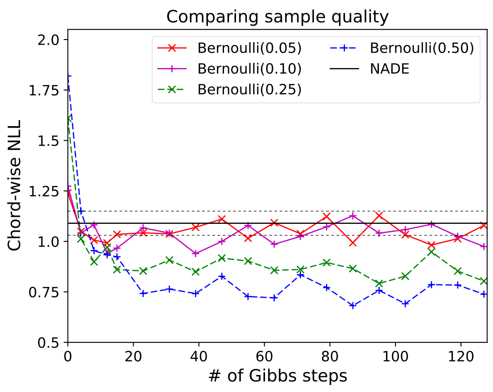


Figure 2. Likelihood under the model for ancestral Gibbs samples obtained with various context distributions $p(C)$. NADE (Bernoulli(0.00)) is included for reference.

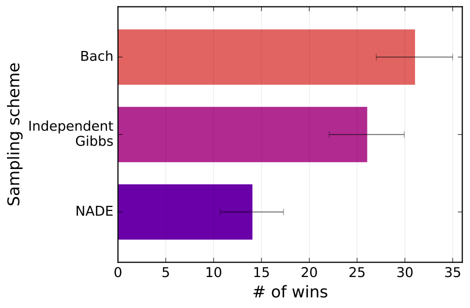


Figure 3. Human evaluations from MTurk on how many times a sampling procedure or Bach is perceived as more Bach-like. Error bars show the standard deviation of a binomial distribution fitted to each's binary win/loss counts.

6.3 Human Evaluations

To further compare the sample quality of different sampling procedures, we carried out a listening test on Amazon's Mechanical Turk (MTurk). The procedures include orderless NADE ancestral sampling and independent Gibbs [40], with each we generate four unconditioned samples of eight-measure lengths from empty piano rolls. To have an absolute reference for the quality of samples, we include first eight measures of four random Bach chorale pieces from the validation set. Each fragment lasts thirty-four seconds after synthesis.

For each MTurk hit, participants are asked to rate on a Likert scale which of the two random samples they perceive as more Bach-like. A total of 96 ratings were collected, with each source involved in 64 (=96*2/3) pairwise comparisons. Figure 3 shows the number of times each source was perceived as closer to Bach's style. We perform a Kruskal-Wallis H test on the ratings, $\chi^2(2) = 12.23, p < 0.001$, showing there are statistically significant differences between models. A post-hoc analysis using the Wilcoxon signed-rank test with Bonferroni correction showed that participants perceived samples from independent Gibbs as more Bach-like than ancestral sampling (NADE), $p < 0.05/3$. This confirms the loglikelihood comparisons on sample quality in 6.2 that independent Gibbs produces better samples. There was also a significance difference between Bach and ancestral samples but not between Bach and independent Gibbs.

7. CONCLUSION

We introduced a convolutional approach to modeling musical scores based on the orderless NADE [35] framework. Our experiments show that the NADE ancestral sampling procedure yields poor samples, which we have argued is because some conditionals are not captured well by the model. We have shown that sample quality improves significantly when we use blocked Gibbs sampling to iteratively rewrite parts of the score. Moreover, annealed independent blocked Gibbs sampling as proposed by [40] is not only faster but in fact produces better samples.

Acknowledgments

We thank Kyle Kastner, Guillaume Alain, Gabriel Huang, Curtis (Fjord) Hawthorne, the Google Brain Magenta team, as well as Jason Freidenfelds for helpful feedback, discussions, suggestions and support. We also thank Calcul Québec and Compute Canada for computational support.

8. REFERENCES

[1] Moray Allan and Christopher KI Williams. Harmonising chorales by probabilistic inference. *Advances in neural information processing systems*, 17:25–32, 2005.

[2] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[4] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *International Conference on Machine Learning*, 2012.

[5] Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35. Citeseer, 2003.

[6] David Cope. Computers and musical style. 1991.

[7] Mary Farbood and Bernd Schöner. Analysis and synthesis of palestrina-style counterpoint using markov chains. In *ICMC*, 2001.

[8] Jose D Fernández and Francisco Vico. Ai methods in algorithmic composition: A comprehensive survey. *Journal of Artificial Intelligence Research*, 48:513–582, 2013.

[9] Johann Joseph Fux. *The study of counterpoint from Johann Joseph Fux's Gradus ad Parnassum*. Number 277. WW Norton & Company, 1965.

[10] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[11] Kratarth Goel, Raunaq Vohra, and JK Sahoo. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn. In *International Conference on Artificial Neural Networks*, pages 217–224. Springer, 2014.

[12] Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Multi-prediction deep boltzmann machines. In *Advances in Neural Information Processing Systems*, pages 548–556, 2013.

[13] Gaëtan Hadjeres and François Pachet. Deepbach: a steerable model for bach chorales generation. *arXiv preprint arXiv:1612.01010*, 2016.

[14] Gaëtan Hadjeres, Jason Sakellariou, and François Pachet. Style imitation and chord invention in polyphonic music with exponential families. *arXiv preprint arXiv:1609.05152*, 2016.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[16] Dorien Herremans and Kenneth Sörensen. Composing first species counterpoint with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts*, 6(4):169–189, 2012.

[17] Dorien Herremans and Kenneth Sörensen. Composing fifth species counterpoint music with a variable neighborhood search algorithm. *Expert systems with applications*, 40(16):6427–6437, 2013.

[18] Lejaren A Hiller Jr and Leonard M Isaacson. Musical composition with a high speed digital computer. In *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

[19] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[20] Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[24] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609, 2016.

[25] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, volume 1, page 2, 2011.

[26] Stefan Lattner, Maarten Grachten, and Gerhard Widmer. Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints. *arXiv preprint arXiv:1612.04742*, 2016.

[27] Feynman Liang. Bachbot: Automatic composition in the style of bach chorales. *Masters thesis, University of Cambridge*, 2016.

[28] Jun S Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994.

[29] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015.

[30] François Pachet and Pierre Roy. Musical harmonization with constraints: A survey. *Constraints*, 6(1):7–19, 2001.

[31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[32] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document, 1986.

[33] Robert H Swendsen and Jian-Sheng Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical review letters*, 58(2):86, 1987.

[34] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

[35] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *arXiv preprint arXiv:1605.02226*, 2016.

[36] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *ICML*, pages 467–475, 2014.

[37] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[38] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1747–1756, 2016.

[39] Arun Venkatraman, Martial Hebert, and J Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI*, pages 3024–3030, 2015.

[40] Li Yao, Sherjil Ozair, Kyunghyun Cho, and Yoshua Bengio. On the equivalence between deep nade and generative stochastic networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 322–336. Springer, 2014.

Poster Session 1

A FRAMEWORK FOR DISTRIBUTED SEMANTIC ANNOTATION OF MUSICAL SCORE: "TAKE IT TO THE BRIDGE!"

David M. Weigl and Kevin R. Page
Oxford e-Research Centre
University of Oxford, United Kingdom
{david.weigl, kevin.page}@oerc.ox.ac.uk

ABSTRACT

Music notation expresses performance instructions in a way commonly understood by musicians, but printed paper parts are limited to encodings of static, a priori knowledge. In this paper we present a platform for multi-way communication between collaborating musicians through the dynamic modification of digital parts: the Music Encoding and Linked Data (MELD) framework for distributed real-time annotation of digital music scores. MELD users and software agents create semantic annotations of music concepts and relationships, which are associated with musical structure specified by the Music Encoding Initiative schema (MEI). Annotations are expressed in RDF, allowing alternative music vocabularies (e.g., popular vs. classical music structures) to be applied. The same underlying framework retrieves, distributes, and processes information that addresses semantically distinguishable music elements. Further knowledge is incorporated from external sources through the use of Linked Data. The RDF is also used to match annotation types and contexts to rendering actions which display the annotations upon the digital score. Here, we present a MELD implementation and deployment which augments the digital music scores used by musicians in a group performance, collaboratively changing the sequence within and between pieces in a set list.

1. INTRODUCTION

Music is a fundamental channel of communication [7], between musicians and an audience, but also among musicians performing together, and as a record of a performance. Inter-performer communication can support *semi-structured* performances such as jam sessions, where the set list is not entirely pre-determined, and repetitions and variations can be added within pieces. These decisions are made and communicated as the performance unfolds.

Music is richly structured, and annotations may serve to interlink musical content along such a structure. Annotations must be able to specifically *address* elements within this structure if they are to be described or related. In multimedia information systems, this is typically achieved using timeline anchors, offsets along a reference recording specified, e.g., in milliseconds. Such timed offsets are not intrinsically *musically* meaningful without context, limiting their use when no reference recording is available.

We can address part of this issue using the Music Encoding Initiative XML schema (MEI; [6]). MEI comprehensively expresses the classes, attributes, and data types required to encode a broad range of musical documents and structures. It does not, however, include or reference concepts, relationships, or existing descriptive forms of multimedia Linked Data external to its schema.

We present the Music Encoding and Linked Data (MELD) framework and implementation architecture that augments and extends MEI structures with semantic Web Annotations capable of addressing musically meaningful score sections. Through its use of Linked Data, our approach deploys knowledge structures expressing relationships unconstrained by boundaries of encoding schema, musical sub-domain, or use-case context, supporting retrieval of a wide range of music information. We employ the flexible and extensible Web Annotation model. New kinds of annotations are easily incorporated through customisation of the MELD JavaScript web-client via drop-in rendering and interaction handlers. Annotations, captured in the context of the performance session with provenance information, can seamlessly reference external data sources, and can in turn be referenced for external analysis, reuse, and repurposing in other contexts.

To demonstrate the feasibility of our approach, we present a prototypical implementation of a performance scenario which collects, distributes, and displays semantic annotations of digital music score in a live jam session.

2. RELATED WORK

Previous projects have applied digital notation to music performance scenarios for just-in-time composition and computer-assisted generation of musical score (e.g., [5, 23, 24]). While fascinating, these approaches are concerned with generating, rather than augmenting, musical score in real-time performance scenarios. Our work, in contrast, concerns the flexible targetting and interlinking of music resources and resource fragments within a framework of meaningfully structured music information.

2.1 Addressing Musical Content

Although typically delivered in a linear rendition, music is richly structured at various levels, from individual notes and performance directions to higher-level musical sections (intro, verse, chorus, bridge; all terms established in western popular music). Viewed as hyperstructures [2], these concepts can be annotated with extra-musical information. Such annotations may be anchored to representations of the music using media fragments [1] [20] expressing temporal positions along a reference timeline in milliseconds, beat instances, or MIDI clock ticks (e.g., [4, 16]).

Temporal anchors are widely used in multimedia information systems – for instance to link to specific scan positions within YouTube videos [2]. As such anchors are not musically meaningful, their usefulness is limited when targeting music in conceptual terms, rather than in terms of a reference recording, such as when annotating music score.

The Music Encoding Initative (MEI; [6]) provides an XML schema encompassing a comprehensive representation of musical structure. Content is cleanly separated from presentation [14], allowing the identification and addressing of (elements of) a musical work – from an entire composition, to a collection of notes constituting a phrase within a particular measure on a specific instrumental part. MEI arranges musical elements, each of which may be named with an XML identifier, within a well-specified hierarchy. These named elements provide anchor points for annotations targeting a musically meaningful structure.

The Open MEI Addressability Service (OMAS; [21]) addresses granular portions of music notation using offsets employing units of musical structure (measures, staves, beats), rather than temporal units. OMAS responds to such an offset specification (supplied via a templated URI syntax) by generating MEI documents containing copies of the specified portion of source MEI, resulting in new resources containing only the portions of music to be addressed. This enables the addressing of musical score without requiring a reference timeline. However, it is not equivalent to addressing a fragment of a resource within its (source) context, a requirement when using the score as a dynamic communication framework between multiple performers. Although specifying offsets in musical terms, OMAS does not directly address musically meaningful sections of the score (e.g., "verse 1," "chorus 2," "bridge").

2.2 Expressing Musical Relationships

Linked Data extends the structure of the World Wide Web by employing URIs to specify directed relationships between data instances. These data instances are themselves encoded by URIs or represented by literal values. In the music information domain, Linked Data has been employed to describe musical resources in terms of associated catalogue metadata (e.g., [8, 22]); to publish features derived from audio-signal content along with associated provenance metadata [12, 13]; and to transcribe symbolic music content [11]. Throughout this article, we apply Linked Data to express annotations about musical structure from a music performance perspective.

Several ontologies – Linked Data formalisations of classes, properties, and relationships within musical subdomains – support the relation, interlinking, reuse, and repurposing of music information within and between data sets and associated applications. We now discuss three pertinent examples: the Music Ontology [17]; the Segment Ontology [3]; and the Common Hierarchical Abstract Representation of Music (CHARM) ontology [9].

While it does not primarily focus on music *performance*, the Music Ontology is a widely used data model describing terms and relationships around the production of musical works, actors (e.g., artists, composers), items (e.g., recordings, published scores), and events (e.g., performances). Its classes extend the Functional Requirements for Bibliographic Records (FRBR) ontology [3], discriminating between musical entities at different levels of abstraction, ranging from (at the most abstract level) the intellectual conception of a musical *work*, to its *expression* (conceptualised, e.g., as musical score), to an embodied *manifestation* (e.g., a publication of the musical score), to (at the most concrete level) a physical *item* representing a single exemplar of a manifestation (e.g., a musician's personal copy of the published score).

The Segment Ontology represents music as comprised of segments ordered along an abstractly defined axis (the *segment line*). These music-generic segments are bridged to different ontological structures expressing elements of musical form appropriate to specific musical sub-domains (e.g., intro, verse, chorus, bridge; or sonata, minuet, trio, or fugue). This separation of concerns supports cross-application to different musical domains and use cases.

CHARM describes music at a fundamental level of pitches, times, and durations, expressing statements as logical formulae operating upon abstract data types. In this paper, we apply more concrete conceptualisations of musical score sufficient for the presented use case; but we invite the prospect of ontological mappings from CHARM to our framework, which could offer intriguing opportunities for re-use and extension for music analytical purposes.

3. MELD FRAMEWORK

The MELD [4] semantic framework combines and augments pertinent subsets of a number of ontologies in a semantic scaffold supporting dynamic distributed annotation of musical score (Figure 1). The Music and Segment Ontologies describe a musical *work*, the music score – a collection of musical segments ordered along a segment line that *express* the work (in FRBR terms) – and finally its *manifestation* as a *published* score encoded as MEI. Collections of MEI fragments, manifestations of musical segments embodied within the published score, anchor annotations processed by rendering and interaction handlers (Section 4). These elements form the core of the MELD semantic framework.

[1] http://www.w3.org/TR/media-frags/

[2] https://developers.google.com/youtube/player_parameters#start

[3] http://vocab.org/frbr/core.html

[4] http://github.com/oerc-music/meld

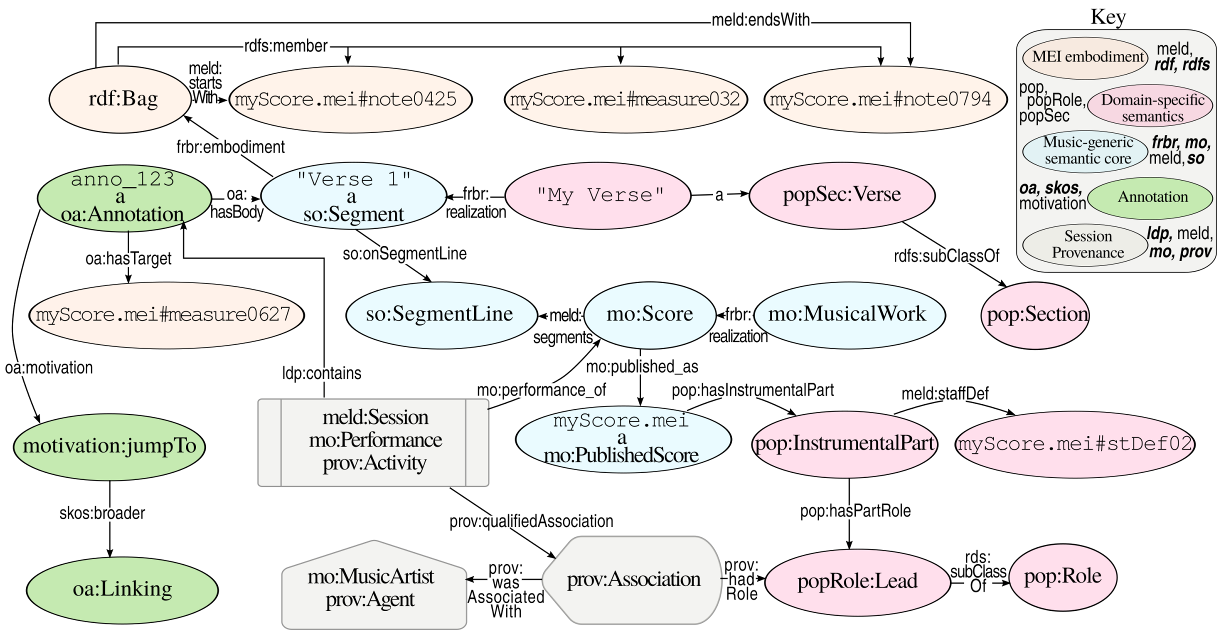


Figure 1. MELD semantic framework. Annotations address music segments embodied in a published score (MEI) resource. A music-generic core is linked to but seperable from domain-specific entities instantiating concrete music sections. Key (top right): External ontologies in ***bold italics***. ***frbr:*** Functional Requirements for Bibliographic Records Core; ***ldp***: Linked Data Platform Vocabulary; *meld*: MELD Vocabulary; ***mo:*** Music Ontology; *motivation:* MELD-specific oa motivations; ***oa:*** Web Annotation Ontology; *pop, popRole, popSec*: Pop music domain-specific semantics; ***prov***: PROV Namespace; ***rdf:*** RDF Concepts; ***rdfs:*** RDF Schema; ***skos:*** Simple Knowledge Organization System; ***so:*** Segment Ontology.

These music-generic structures are linked with, but separable from, components expressing domain-specific entities associated with the concrete instantiations of musical segments. Here, we have specified a taxonomy of popular music performance terms sufficient to accomodate our use case. Notably, these domain- and use-case-specific taxonomies may be modularly replaced by other ontological structures reflecting different domains (e.g., popular vs. classical music) or use cases (e.g., annotations supporting music performance vs. musicological scholarship) without modification of the core.

As each entity, class, and relationship in the framework is assigned its own URI, the entire ontological structure, as well as the generated annotation and session metadata, is part of a wider web of Linked Data. This enables the seamless inclusion of external information within MELD annotations, as well as the referencing, reuse, and repurposing of the generated information by external services.

3.1 MELD Annotations

MELD annotations build upon the Web Annotation Data Model,[5] a W3C recommendation providing an extensible, interoperable, machine-readable means of creating annotations by asserting relationships between a set of connected resources, typically an annotation *body* and a *target* (or target resource fragment).

Web Annotations may be associated with an explicit *motivation* formalising the given annotation's intended purpose. In MELD, domain- and use-case-specific rendering and interaction clients (Section 4.2) make use of this information to map the annotation to corresponding rendering and interaction handlers to effect changes to the score displayed to the user. By defining MELD-specific motivations subclassing generic ones specified within the Web Annotation model, we promote reuse and repurposing of MELD annotations in external contexts implementing Web Annotation standards.

Web Annotations may also be associated with an intended *audience* to whom a given annotation applies. In MELD, this information is used to address annotations to only certain specified participants – for instance, the player of a given part within a session. By default, annotations that do not specify an audience are made available to every client associated with a given session.

3.2 Domain Ontologies

We have specified a taxonomy describing sections and part roles of popular music. Abstract, music-generic ordered segments of a score are associated with more concrete notions of musical sections appropriate to popular music (e.g., "intro", "verse", "chorus", "bridge") via the Segment Ontology. In its original conception, this ontology bridges music-generic and domain-specific conceptions of musical segmentation by mapping an abstract segment line to a concrete reference timeline manifested along a musical recording. To avoid the requirement of such a reference, we instead anchor music-generic segments to

[5] `https://www.w3.org/TR/annotation-model/`

domain-specific musical sections, embodied as manifestations within the published score. These are represented as collections of media fragment URIs specifying the named MEI elements that comprise the given section. Where reference recordings form a part of the use case, both approaches are applicable, enabling flexible, complementary structuring of music information via temporal, symbolic, and semantic anchors. The simplicity of repurposing the Segment Ontology's bridging mechanism within our novel context is afforded by our use of Linked Data.

The published score, represented in MELD by an MEI resource, is also associated with domain-specific part roles (e.g., "lead", "bass", "rhythm") anchored within the MEI via fragment URIs specifying the corresponding MEI staff definition container element. We employ the PROV Ontology to express and track the provenance of relationships associating specific musicians with such part roles within the context of a particular performance session.

By virtue of the clean separation between *music-generic* and *popular-music-performance-specific* ontological structures, the domain-specific structures may be swapped out to address other use contexts while retaining the rest of the presented framework, e.g. for an analytical ontology of musicological terms supporting the use of digital score annotations to illustrate points in scholarly musicological arguments. This flexibility of ontological schema is another key affordance of Linked Data.

4. MELD ARCHITECTURE

The MELD architecture (Figure 2) implements server- and client-side components: RESTful web services are used to manage session and annotation resources; client-side rendering and interaction handlers display and update annotated digital score parts relevant to each user, as well as capturing user interactions and updating server-side resources with interaction outcomes.

4.1 Web Services

MELD annotation and session management services are implemented using a Python web server capable of handling operations on RDF and JSON-LD datasets encoding the collection of MELD sessions, as well as the performer part-roles and annotations associated with each session.

4.1.1 Performance Session Service

The server exposes a RESTful web service providing access to a resource representing the list of all MELD sessions available to a user (requested via HTTP GET). In order to create a new session, a Linked Data representation of a basic session resource, related by a `mo:performance_of` predicate to the published score MEI resource, is posted (via HTTP POST) to the list. The server mints a URI to represent the new session. A 'join' resource is exposed to establish qualified associations between the respective performer and an instrumental part in the session context.

4.1.2 Annotation Service

Clients interact with the annotation service using an API based on the Web Annotation Protocol[6], which specifies transport mechanisms for creating and managing annotations that are consistent with Web Architecture and REST best practices. This involves casting each session as an *annotation container*, a form of Linked Data Platform (LDP) container[7] with additional constraints derived from the Web Annotation data model. New annotations are posted to the annotation container, where they are associated with the session using `ldp:contains` relationships.

MELD extends the Web Annotation model by tracking the *state* of each annotation associated with each performer, in order to support the dynamic, real-time nature of the distributed annotation activity. Annotations are created in a *raw* state. Upon the occurrence of certain events (e.g., user interaction), clients may optionally effect a *processed* annotation state (via HTTP PATCH), signifying that the annotation has been handled and is no longer of relevance within the session context. This approach is preferable to simply deleting the handled annotation (e.g., via HTTP DELETE), as it may remain relevant in external contexts – for instance for post-session review by the performers, or by other interested observers.

4.2 Rendering and Interaction Clients

A JavaScript web client is responsible for rendering MEI score parts to the user. The client dynamically augments this display with currently relevant annotations; handles user interactions; and communicates interaction outcomes using the MELD web services (Section 4.1).

The procedure is illustrated in Figure 2. The client processes a JSON-LD [19] representation of the graph associated with the session resource, framed [18] to include only relevant annotations by filtering on audience and state (Sections 3.1 and 4.1.2). It retrieves the MEI resource associated with the session (HTTP GET), and renders the encoded score using Verovio [15], a tool that produces SVG engravings of MEI-encoded music notation. Crucially, Verovio retains the MEI hierarchy and element identifiers into the produced SVG output, supporting the addressing of graphical score elements for visual markup and dynamic interaction through the web browser.

Each annotation is mapped to a corresponding handler, which may be customised to a specific motivation and use-case using CSS styling and drop-in JavaScript functions. User-interaction outcomes trigger AJAX calls using the web services to POST a new annotation, or to PATCH an annotation's state from *raw* to *processed*.

Each client continuously polls the session resource (using HTTP GET), maintaining the annotated score at its latest state in near real-time. Simple resource hashes (HTTP entity tags) and atomic server-side file writing reduce network traffic and address race conditions inherent in distributed real-time interaction: the former by supplying

[6] `https://www.w3.org/TR/annotation-protocol/`

[7] `https://www.w3.org/TR/ldp-primer/`. LDP provides read-write access to RDF datasets via RESTful HTTP services.

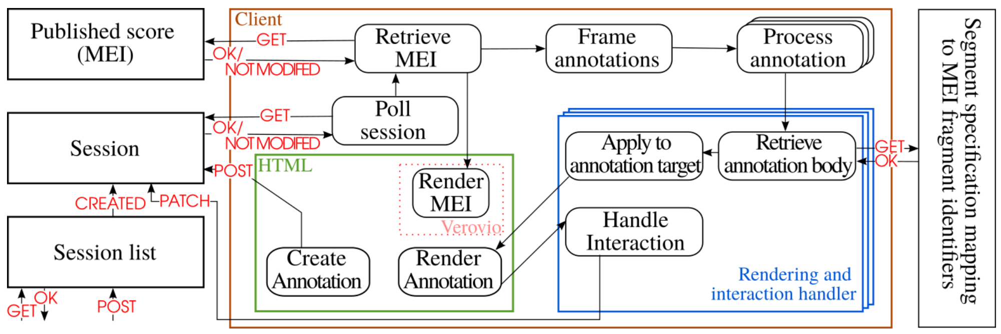


Figure 2. MELD architecture: performance session and annotation web services; MELD client; and rendering and interaction handlers. A tight polling loop, employing entity tagging techniques to reduce network load and address concurrency concerns, distributes and maintains annotation state between performers in near-real-time.

lightweight HTTP 304 (“Not Modified”) responses that exclude the resource body when a resource has not changed from its state last seen by the client; and the latter by the server rejecting changes (via HTTP 412 – “Precondition Failed”) upon an entity tag mismatch, prompting the client to GET the latest version of the session resource before reattempting its modification.

5. SCENARIO

To validate the feasibility of our proposed approach, we have produced an implementation of MELD to support a simple scenario where musical performers collaborate within a semi-structured performance environment.

5.1 Motivation

The selected use case is a performance – such as a jam session – where collaborating musicians make fluid, ad-hoc decisions about song repetitions and transitions, rather than adhering to a pre-determined set list. Musicians may add directions to change dynamic or stylistic elements of the performance, e.g., to reference a particular prior recording they wished the group to emulate in style or interpretations; or they may incorporate structural directions, e.g., to repeat a chorus or a verse, or to move to a bridge section. Such directions transcend the symbolic representation of the music being performed, and may draw upon significant contextual information external to the performance itself, for instance to adopt the style of a certain artist, or to transition to another song by this artist.

5.2 Implementation

A session, corresponding to the performance of a given song, is represented as an LDP container. It contains annotations, and connects participating performers to instrumental part roles via qualified associations (Figure 1). The corresponding MEI staff definition elements are used to filter the music structure in order to display only the relevant instrumental parts to each performer on a personal touchscreen (Figure 4). Only annotations pertinent to their performance are displayed, using the Web Annotation *audience* property as a filtering mechanism (Section 3.1).

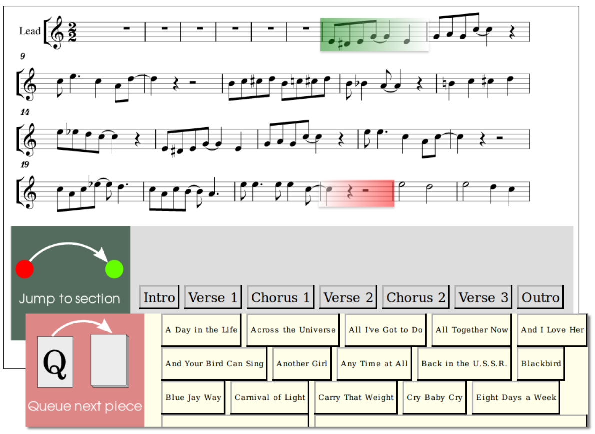


Figure 3. The MELD client displays the annotated digital score. A modal action pane enables users to generate annotations enacting jumps within a piece, or queueing actions determining the next piece to be performed.

Rendering and interaction handlers operate on annotations that enable performers to collaboratively change the performance sequence within and between musical scores. Annotations are generated by using the touch screen to interact with a modal action pane situated below the musical score (Figure 3).

Navigational changes of sequence within a score – e.g., a jump to the bridge section – are requested by specifying a jump *source* (highlighted on the score in red) and *destination* (highlighted in green). The former is specified by tapping on a measure of the score. The action pane then displays a list of musical sections representing potential destinations. This list is automatically retrieved from the Linked Data, by traversal of the segments on the segment line associated with the current session’s score. When a selection is made, an annotation is POSTed to the session, specifying an annotation target (the fragment URI of the jump source measure); an annotation body (the URI of the selected destination); and `motivation:jumpTo` (a custom specialisation of Web Annotation’s `oa:linking`) as the motivation. All performer clients retrieve the annotation upon the next polling cycle.

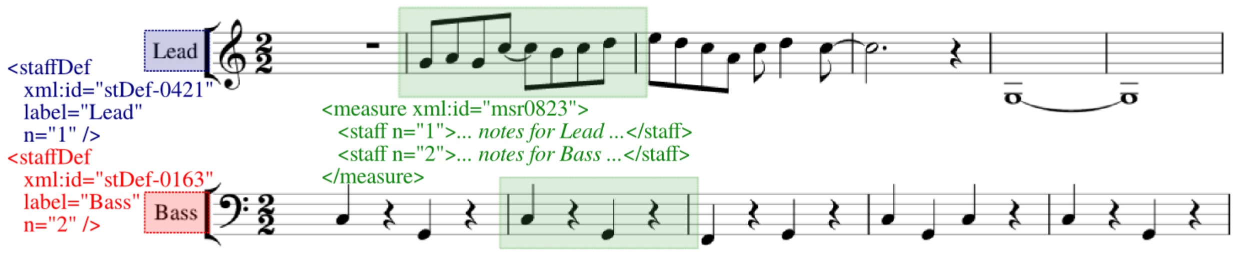


Figure 4. MEI staff definition element identifiers are associated with each performer in a session resource (Figure 1), enabling filtering according to role. Here, an annotation targeting a segment embodied as `myScore.mei#msr0823` retrieves the corresponding notes for Lead or Bass, according to the performer's part-role association within the session.

Upon tapping on the red source measure, the rendering and interaction handler flips to the score page containing the destination measure. The highlights then fade, and the performer's copy of the annotation is PATCHed as *processed* to avoid rerendering of stale information. [8]

Annotations representing queueing instructions for navigation between songs may target individual source measures as described above, or the URI of the current score (in which case the rendering and interaction handler creates a "next piece" button in place of the "next page" button on the last page of the current piece). Songs to transition to can be selected according to a criteria such as "More songs by this artist". These parameterise SPARQL queries [9], in this case with the URI of the artist from the current score's MEI responsibility statement. The SPARQL query retrieves a list of songs by this artist from DBPedia [10] [1], matched against a cache of available MEI resources stored on a local triplestore (Linked Data database). Upon selecting a song, the annotator's client requests the creation of a new session associated with the corresponding MEI resource. It then posts an annotation to the current session, instructing all performers' clients to join the next session when an interaction event on the annotation target (the jump source) is handled.

6. CONCLUSIONS AND FUTURE WORK

We have presented the MELD framework and architecture applying musical structure as a semantic spine for real-time annotation of digital music score. RESTful web services manage the retrieval and distribution of annotations created by user interactions in a performance session. Annotations address MEI-encoded score elements using Linked Data, enabling flexible reuse, repurposing, and interlinking of the generated information in external contexts. The framework and associated implementation architecture comprise separable music-generic and domain-specific semantic structures, and modular rendering and interaction handlers, allowing components to target different music domains and use cases. We have validated the feasibility of the proposed framework through a MELD implementation supporting multi-way communication between musicians collaboratively changing the performance sequence within and between pieces in a group performance.

In future work, we will focus on extending the capabilities of the described framework to incorporate annotations enacting modifications of the MEI structure, for instance requesting changes in dynamics. Such modifications are readily incorporated within the client polling cycle (Section 4.2), given the speedy rendering performance of Verovio and the relation of the polled session resource (via `mo:performance_of`) to the MEI notation being rendered. However, session management complexities including MEI resource duplication, versioning, and the capture of provenance information will need to be accomodated, and potential licensing issues carefully considered.

While physical interactions with musical score are commonplace in music performance – consider the necessity of paging through paper parts – care must be taken in ongoing interface development to minimise additional cognitive load upon the performer. We have developed an alternative, multimodal interaction mechanism by integrating the MELD framework with a technology supporting real-time triggering of HTTP actions in response to specified patterns matched to an audio or MIDI stream. This variation of MELD has been successfully applied to drive a gamified composition for disklavier and electronics [10], demonstrating a more complex performance interaction than the simple scenario used to explain the MELD framework in this paper. Other interaction paradigms are imaginable, for instance by means of foot pedals, or voice commands.

Finally, we are exploring the application of MELD in non-performance-related use cases. MELD annotations may be specified to target MEI and other digital media, including audio, images, and textual commentary. Thus interlinked, such media resources, annotated to target musically meaningful sections, may be used to illustrate scholarly musicological arguments. Together with the work presented here, these applications demonstrate the utility and flexibility of adopting a semantic framework anchored in addressable musical structure to express, retrieve, and distribute information in a variety of contexts and use cases.

[8] Measures are defined above staves within the MEI hierarchy, meaning that the notes contained in the targeted measures, and the pages they appear on, likely differ between performers, according to their instrumental part associations

[9] SPARQL is a query language for Linked Data resources, analogous to SQL queries against relational databases, but capable of retrieving data spanning local and external data sets

[10] DBPedia publishes Wikipedia information as structured Linked Data

7. ACKNOWLEDGEMENTS

This work is undertaken as part of the Fusing Audio and Semantic Technologies for Intelligent Music Production and Consumption project, funded by the UK Engineering and Physical Sciences Research Council under grant number EP/L019981/1, a collaboration between Queen Mary University of London, the University of Nottingham, and the University of Oxford.

We gratefully acknowledge the support of our colleagues. Particularly, we thank David Lewis for guidance and insight during the development of the MELD framework, and his collaboration in extending our approach to applications in musicological scholarship; Graham Klyne for consultation regarding PROV-O mappings; and Leia Yeomans for her valuable assistance in realising the graph layout for Figure 1 of this paper.

8. REFERENCES

[1] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sren Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154 – 165, 2009. The Web of Data.

[2] David C. De Roure, Don G. Cruickshank, Danius T. Michaelides, Kevin R. Page, and Mark J. Weal. On hyperstructure and musical structure. In *Proceedings of the thirteenth ACM conference on Hypertext and hypermedia*, pages 95–104. ACM, 2002.

[3] Ben Fields, Kevin Page, David De Roure, and Tim Crawford. The segment ontology: Bridging music-generic and domain-specific. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011.

[4] Martin Gasser, Andreas Arzt, Thassilo Gadermaier, Maarten Grachten, and Gerhard Widmer. Classical music on the web–user interfaces and data representations. In *Proc. 16th International Society for Music Information Retrieval Conference*, 2015.

[5] Georg Hajdu. Real-time composition and notation in network music environments. In *International Computer Music Conference*, 2008.

[6] Andrew Hankinson, Perry Roland, and Ichiro Fujinaga. The Music Encoding Initiative as a Document-Encoding Framework. In *Proc. 12th International Society for Music Information Retrieval Conference*, pages 293–298, 2011.

[7] David J. Hargreaves, Raymond MacDonald, and Dorothy Miell. How do people communicate using music? In *Musical communication*, pages 1–25. Oxford University Press Oxford, 2005.

[8] Kurt Jacobson, Simon Dixon, and Mark Sandler. Linkedbrainz: Providing the musicbrainz next generation schema as linked data. In *Late-breaking demo session at the 11th International Society for Music Information Retrieval Conference*, 2010.

[9] Kurt Jacobson, Simon Dixon, and Mark Sandler. An ontology for abstract, hierarchical music representation. In *Late-breaking demo session at the 16th International Society for Music Information Retrieval Conference*, 2015.

[10] Maria Kallionpää, Chris Greenhalgh, Adrian Hazzard, David M. Weigl, Kevin R. Page, and Steve Benford. Composing and realising a game-like performance for disklavier and electronics. In *New Interfaces for Musical Expression (NIME)*, 2017.

[11] Albert Meroño-Peñuela and Rinke Hoekstra. The song remains the same: Lossless conversion and streaming of MIDI to RDF and back. In *International Semantic Web Conference*, pages 194–199. Springer, 2016.

[12] Kevin R. Page, Sean Bechhofer, Georgy Fazekas, David M. Weigl, and Thomas Wilmering. Realising a layered digital library: Exploration and analysis of the live music archive through linked data. In *Joint Conference on Digital Libraries*, 2017.

[13] Kevin R. Page, Benjamin Fields, Bart J. Nagel, Gianni O'Neill, David C. De Roure, and Tim Crawford. Semantics for music analysis through linked data: How country is my country? In *e-Science (e-Science), 2010 IEEE Sixth International Conference on*, pages 41–48. IEEE, 2010.

[14] Laurent Pugin, Johannes Kepper, Perry Roland, Maja Hartwig, and Andrew Hankinson. Separating Presentation and Content in MEI. In *Proc. 13th International Society for Music Information Retrieval Conference*, pages 505–510, 2012.

[15] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. Verovio: A library for engraving MEI music notation into SVG. In *Proc. 15th International Society for Music Information Retrieval Conference*, pages 107–112, 2014.

[16] Fazilatur Rahman and Jawed Siddiqi. Semantic annotation of digital music. *Journal of Computer and System Sciences*, 78(4):1219 – 1231, 2012.

[17] Yves Raimond, Samer A Abdallah, Mark B Sandler, and Frederick Giasson. The music ontology. In *8th International Conference on Music Information Retrieval*, pages 417–422, 2007.

[18] Manu Sporny, Gregg Kellogg, Dave Longley, and Markus Lanthaler. JSON-LD framing 1.1: An application programming interface for the JSON-LD syntax. Community group draft report, W3C, 2017. `http://json-ld.org/spec/latest/json-ld-framing/`.

[19] Manu Sporny, Dave Longley, Gregg Kellogg, and Markus Lanthaler. JSON-LD 1.0: A JSON-based serialization for linked data. Recommendation, W3C, January 2014. `http://www.w3.org/TR/json-ld/`.

[20] Raphaël Troncy, Lynda Hardman, Jacco Van Ossenbruggen, and Michael Hausenblas. Identifying spatial and temporal media fragments on the web. In *W3C Video on the Web Workshop*, pages 4–9, 2007.

[21] Raffaele Viglianti. The music addressability API: A draft specification for addressing portions of music notation on the web. In *Proceedings of the 3rd International Workshop on Digital Libraries for Musicology*, DLfM 2016, pages 57–60, New York, NY, USA, 2016. ACM.

[22] David M. Weigl, David Lewis, Tim Crawford, Ian Knopke, and Kevin R. Page. On providing semantic alignment and unified access to music-library metadata. *International Journal on Digital Libraries*. Accepted for publication: Special issue on Digital Libraries for Musicology.

[23] Gerhard E Winkler. The realtime-score: A missing-link in computer-music performance. *Sound and Music Computing*, 4, 2004.

[24] Lonce Wyse and Jude Yew. A real-time score for collaborative just-in-time composition. *Organised Sound*, 19(3):260–267, 2014.

A MUSIC PLAYER WITH SONG SELECTION FUNCTION FOR A GROUP OF PEOPLE

Jun'ichi Suzuki and Tetsuro Kitahara
College of Humanities and Sciences, Nihon University, Tokyo 156-8550, Japan
{junichi,kitahara}@kthrlab.jp

ABSTRACT

There are often situations in which a group of people gather and listen to the same songs. However, majority of existing studies related to music information retrieval (MIR) have focused on personalization for individual users, and there have been only a few studies related to MIR intended for a group of people. Here, we present an Android music player with a music selection function for people who are listening to the same songs in the same place. We assume that each user owns his/her favorite songs on his/her Android device. Once a group of users gathers each user can launch this player on his/her smartphone. Then, the player running on each device starts to communicate with other devices via Bluetooth. Information about songs stored in every device, along with the playback history, is collected to a device referred to as the master device. Then, the master device estimates each user's preference for every song based on playback history and music similarity. The master device then extracts songs that are highly preferred and sends a command to start playback to the devices storing these songs. Our experimental results demonstrate the successful estimation of music preferences based on music similarity.

1. INTRODUCTION

In situations such as parties and carpooling, it is common that a group of people gather and listen to the same background music. However, it is not easy to select songs in such situations: if a particular member selects songs based on his/her musical preferences, other members with different musical preferences may be unsatisfied. To resolve this problem, we need a mechanism to extract each member's musical preferences and select songs taking into account those preferences.

Although music information retrieval (MIR) has a long history of technology development [3], relatively few attempts have been made to develop MIR techniques for a group of people. MusicFX, developed by Jseph et al. [6], selects a music broadcasting station from 91 stations specializing in different music genres, and the selection process is indirectly influenced by the musical preference of each member present at that place. This system uses a preference database consisting of every member's preference for 91 genres on a scale of -2 to 2. After detecting who is present based on each member's electronic badge, the system determines a broadcasting station using this database with a group preference arbitration algorithm. This algorithm basically computes a group preferences as a squared summation of individual preferences. Crossen et al. [4] developed a similar system called Flytrap. This system also attempts to play songs that are pleasing to every person present. The system detects the people who are present and sends information about each member's previous music choices to a server. Based on a voting mechanism in which high votes are given to those that have been listened to previously, songs to be played back are determined. A web application developed by Popescu et al. [8], called GroupFun, helps a group of friends agree on a common music playlist. With GroupFun, users can listen to and rate their own songs as well as their friends' songs. The system then arbitrates between the users' preferences using four different algorithms to determine which songs to play. BlueMusic, developed by Mahato et al. [1], uses Bluetooth to send an indivdual's musical preference data to a public music playback system. Individual users enter their musical preferences into a web form in advance and obtain strings, such as "Bm+A1R3EST3," that encodes their musical preferences. Their mobile devices then broadcast these data as Bluetooth device names. The public music playback system collects such strings to determine which songs to play back. In addition, some researchers developed music recommendation systems for a group of people [5,7]. These systems typically assume that (1) information on individual users' musical preferences is collected (or estimated from playback histories) in advance and (2) songs to be played back are stored in a server or public playback system.

Here, we develop a music selection and playback application under the following assumptions:

- Individual users own songs inside their own smartphones (or mobile devices).

  This means that songs to be played back are stored separately in multiple devices. Appropriate songs should be selected from a music collection dispersing at multiple devices and should be played back without manually switching any settings.

- No server or special equipment is necessary.

  The application should run on individual users' devices and songs are never copied to a server to avoid copyright issues.

- An individual device possesses information about only its playback history.

  The application running on each device has no prior knowledge about how much the device owner favors each song stored in other devices. The appliaction has to estimate this information from data that the device has.

To meet these assumptions, we design the application based on the following policy:

- One of the devices is regarded as the master devices, and every device communicates the list of songs stored in it and its playback history (in particular, how many times the device owner has played back each song) to that device.

- The master device does not collect the waveforms of songs stored in other devices but rather commands the device storing a song to be played back to connect itself directly to the Bluetooth speaker. If a song stored in a different device is selected next, this device will be automatically connected to the speaker after the current device is disconnected from the speaker.

- Each user's preference for songs stored in others' devices is estimated based on the similarity to songs stored in his/her own device.

The rest of the paper is organized as follows: In Section 2, we describe the overview of our application and present a method for estimating the degree of preference noted above. In Section 3, we report the system implementation and experiments. Finally, we conclude the paper in Section 4.

2. SYSTEM OVERVIEW

This application aim to select songs from a collection separately stored in different devices and play the songs back seamlessly. As discussed in the Introduction section, we designed the application based on the following policies:

- Every device communicates information about songs (not the waveforms themselves) to the master device.

- After the song selection process occurs, the master device commands the device storing the selected song to connect itself directly to the Bluetooth speaker to avoid copying the waveform somewhere.

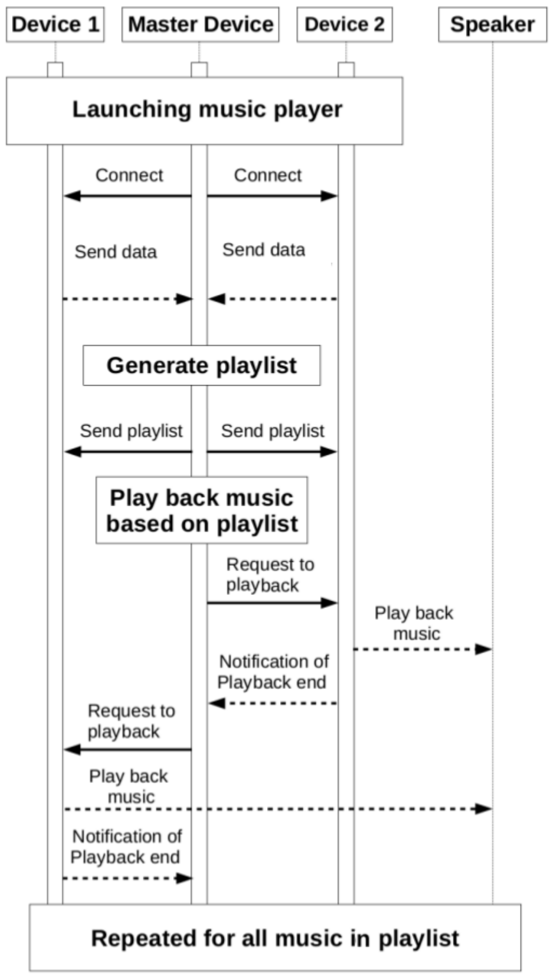


Figure 1. Overview of system flow in the networked playback mode

- Each user's preferences for songs stored on other users' devices is estimated based on the similarity between these songs and the songs stored in his/her device.

The application has two different modes. One is the normal playback mode, in which the user listens to songs in a normal way. This mode provides basically the same functionalities as a typical music player and is used to store the playback history, particularly the number of plays (i.e., how many times the user has listened to each song). The other mode is a networked playback mode, which is the primary mode of this application. Once users gather and launch the application on their devices, the devices start to communicate with one other via Bluetooth, and one of the devices is set to be the master device. After establising aBluetooth connection, the master device collects information about the list of songs stored in each device and the number of plays (how many times each song is played back) from the other devices. Then, the master device generates a playlist and commands the device storing each song in the playlist to play back the song.

Below, we illustrate the procedure of the networked playback mode (Figure 1).

2.1 Launching the Application

Let $\mathcal{U} = \{u_1, \cdots, u_n\}$ and $\mathcal{D} = \{d_1, \cdots, d_n\}$ be a group of users and a set of the users' devices, respectively, where d_1 is the master device. Each device d_i communicates the list of songs $M_{(d_i)} = \{m_{(d_i,1)}, \cdots, m_{(d_i,n(d_i))}\}$ and the number of plays for every song $F(m_{(d_i,k)})$ ($k = 1, \cdots, n(d_i)$) to the master device d_1.

2.2 Calculating the Degree of Preference

For each user u_i, the degree of preference for every song is calculated. Let $W(u_i, m_{(d_j,k)})$ be the degree of preference of user u_i for song $m_{(d_j,k)}$. We formulate the degree of preference based on the following assumptions:

1) If a user listen to a song frequently, his/her preference for that song should be high.
2) If Song A is similar to Song B, which is highly favored, Song A should also highly favored.

Based on the first assumption, we can calculate the degree of a user's preference for songs stored in his/her own device using the number of plays. On the other hand, the degree of preference for songs stored in others' devices cannot be calculated based on the number of plays because such information is not available. Based on the second assumption, we calculate the degree of preference for such songs using that for similar songs owned by oneself.

2.2.1 The Degree of Preference for Owned Songs

The degree of preference for songs stored in a user's own device is calculated based on the number of plays. Here we prepare two different definitions:

$$W(u_i, m_{(d_i,k)}) = 1 - \frac{1}{\{F(m_{(d_i,k)}) + 1\}^{\alpha}} \tag{1}$$

and

$$W(u_i, m_{(d_i,k)}) = \cosh \beta F(m_{(d_i,k)}), \tag{2}$$

where α and β are parameters.

2.2.2 The Degree of Preference for Songs That Are Not Owned

Here, we calculate the degree of preference for songs stored in other users' devices $W(u_i, m_{(d_j,k)})$ ($i \neq j$).

First of all, the similarity of $m_{(d_j,k)}$ to every song stored in d_i is calculated. Musical similarity is a very difficult concept and its calculation is still an open problem. However, various similarity measures have been developed from different points of view (e.g., [2]). Here, we combine two different similarity measures: acoustic similarity and (socially obtained) artist similarity.

To calculate acoustic similarity between two songs, $m_{(d_j,k)}$ and $m_{(d_i,l)}$, a sequence of 20-dimensional mel-frequency cepstral coefficient (MFCC) vectors is calculated from each song by adopting a shift by 160 samples after the waveform is resampled to 16 kHz. The Earth mover's distance between the two sequences, denoted by $D(m_{(d_j,k)}, m_{(d_i,l)})$, is calculated. The similarity $\underset{\text{MFCC}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)})$ is then calculated by using

$$\underset{\text{MFCC}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}) = \frac{1}{1 + D(m_{(d_j,k)}, m_{(d_i,l)})}.$$

The artist similarity between $m_{(d_j,k)}$ and $m_{(d_i,l)}$ is calculated based on the Last.fm API[1] . The Last.fm API has a function, *artist.getSimilar*, which returns up to 100 similar artists to a specified artist, along with similarity values on a scale of 0 to 1. Let $a_{(d_j,k)}$ be the artist of the song $m_{(d_j,k)}$. The list of artists similar to $a_{(d_j,k)}$, denoted by $\mathcal{A}_{(d_j,k)}$, and their similarities $\underset{\text{last.fm}}{\text{sim}}(a_{(d_j,k)}, a')$ ($a' \in \mathcal{A}_{(d_j,k)}$) are obtained using Last.fm. In general, $\underset{\text{last.fm}}{\text{sim}}(\cdot,\cdot)$ is not symmetric. We therefore define the artist similarity as follows:

i) When $a_{(d_j,k)} = a_{(d_i,l)}$,

$$\underset{\text{artist}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}) = 1.0.$$

ii) When $a_{(d_j,k)} \in \mathcal{A}_{(d_i,l)}$ and $a_{(d_i,l)} \in \mathcal{A}_{(d_j,k)}$,

$$\begin{aligned}&\underset{\text{artist}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)})\\ &= \frac{1}{2}\left\{\underset{\text{last.fm}}{\text{sim}}(a_{(d_j,k)}, a_{(d_i,l)}) + \underset{\text{last.fm}}{\text{sim}}(a_{(d_i,l)}, a_{(d_j,m)})\right\}\end{aligned}$$

iii) When $a_{(d_j,k)} \in \mathcal{A}_{(d_i,l)}$ and $a_{(d_i,l)} \notin \mathcal{A}_{(d_j,k)}$,

$$\underset{\text{artist}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}) = \underset{\text{last.fm}}{\text{sim}}(a_{(d_j,k)}, a_{(d_i,l)})$$

iv) When $a_{(d_j,k)} \notin \mathcal{A}_{(d_i,l)}$ and $a_{(d_i,l)} \in \mathcal{A}_{(d_j,k)}$,

$$\underset{\text{artist}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}) = \underset{\text{last.fm}}{\text{sim}}(a_{(d_i,l)}, a_{(d_j,k)})$$

v) When $a_{(d_j,k)} \notin \mathcal{A}_{(d_i,l)}$ and $a_{(d_i,l)} \notin \mathcal{A}_{(d_j,k)}$,

$$\underset{\text{artist}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}) = \varepsilon,$$

where ε is basically zero but can be set to a very small positive value to take into account the possibility of sparseness in similar artist responses ($\varepsilon = 0.01$ in the current implementation).

The similarity between two songs, denoted by $\text{sim}(m_{(d_j,k)}, m_{(d_i,l)})$, can be calculated as

$$\begin{aligned}&\text{sim}(m_{(d_j,k)}, m_{(d_i,l)})\\ &= \underset{\text{MFCC}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}) \cdot \underset{\text{artist}}{\text{sim}}(m_{(d_j,k)}, m_{(d_i,l)}).\end{aligned}$$

Using this similarity measure, the degree of preference can be calculated as follows:

$$W(u_i, m_{(d_j,k)}) = \frac{\sum_l \text{sim}(m_{(d_j,k)}, m_{(d_i,l)}) W(u_i, m_{(d_i,l)})}{\sum_l \text{sim}(m_{(d_j,k)}, m_{(d_i,l)})}.$$

[1] http://www.last.fm/api

2.3 Integration of Degrees of Preference

The degree of every user's preference is integrated as follows:

$$W_{\text{all}}(m_{(d_j,k)}) = \prod_{q=1}^{n} W(u_q, m_{(d_j,k)})$$

2.4 Generating a Playlist

After the integrated degrees of preference $W_{\text{all}}(m_{(d_j,k)})$ for all songs stored in all of the devices are calculated, a necessary number of songs are selected in order of the integrated degree of preference.

2.5 Playing Songs

Let $L = \{m_1, \cdots, m_c\}$ be the list of the selected songs and let $d(m_i)$ be the device storing the song m_i. For each song m_i, the following steps are executed:

1) The master device commands $d(m_i)$ to connect itself to the Bluetooth speaker.

2) The device $d(m_i)$ starts to play back m_i.

3) The master device broadcasts the information (title, artist name, etc.) of the song being played to all devices to alert users about which song is being played.

4) Once the playback ends, the device $d(m_i)$ disconnects from the Bluetooth speaker and sends the master device a message communicating the end of playback.

5) Return to 1) for the next song.

Advanced Audio Distribution Profile (A2DP) is used for the connection between each device and the speaker. For communication between devices, Serial Port Profile (SPP) is used.

3. IMPLEMENTATION AND EXPERIMENTS

3.1 Implementation

We implemented this music player on Android 5.0 smartphones. Screenshots are shown in Figure 2. A demo video of this application is available at `https://www.youtube.com/watch?v=gcOWjkBc_EA`. For roughly one hour, we confirmed that this application sucessfully selected songs, switched the connection to the Bluetooth speaker, and played the selected songs without any troubles.

3.2 Evaluation of the Degree of Preference for Owned Songs

We confirmed the appropriateness of the calculation of the degrees of preference for a user's own songs by checking how these degrees calculated with our method matched the actual level of preference reported by the users.

3.2.1 Dataset

Data of playback histories and preferences were obtained from Last.fm. Using the Last.fm API, we obtained a user profile containing playback histories and evaluations (the "Liked" tag or no tag) for vairous songs. We collected user profiles for 45,745 users, who were specified by choosing one user at random and then traversing "Friend" links recursively. However, many users did not apply the "Liked" tag to any songs, we therefore chose to focus on the 11,074 users who gave "Liked" tags to 20–5000 songs. The total number of the playback histories is 98,504,128.

3.2.2 Results

We analyzed the ratio of songs with the "Liked" tag with respect to the numbers of plays binned by 5 (Figure 3). From this figure, one can see that as the number of plays increases, the ratio of songs with the "Liked" tag increases. Regarding this ratio as the ground truth of the degree of preference, we evaluated the degree of preference calculated with Equations (1) and (2). The results are shown in Figure 4. One can see that using Equation (1) with $\alpha = 0.01$ approximates the ground truth well. For songs with less than 20 times of plays, Equation (1) with $\alpha = 0.10$ and Equation (2) with $\beta = 0.005$ approximate better.

3.3 Evaluation of the Degree of Preference for Songs That Are Not Owned

Next, we confirm the appropriateness of the calculation of the degree of preference for songs not owned by a user.

3.3.1 Dataset

We used the Last.fm Dataset[2] , which consists of playback histories of 1,000 users. Of the songs contained in this database, some songs were regarded as owned songs and other songs were regarded as unowned songs and their playback histories were accordingly hidden. From the playback histories of the songs regared as unowned, the degrees of preference were estimated. Ideally, the estimated degrees should be compared with real favor values (e.g., from questionnaires), but such data were not available from this database. We therefore regarded the degrees of preference calculated from the playback histories as the quasi ground truth. Because this database does not include the waveforms themselves, a 30-second version of every song is downloaded via 7digital API[3] for feature extraction for calculating acoustic similarity.

3.3.2 Procedure

We extracted 118 three-user groups such that the number of songs listened to by three users of every group exceeded 300. For each group, we divided the songs listened to by the group members into three sets; each set was regarded as being owned by each member. We calculated the degrees of preference for unowned songs of each member using the

[2] http://ocelma.net/
[3] https://www.7digital.com/

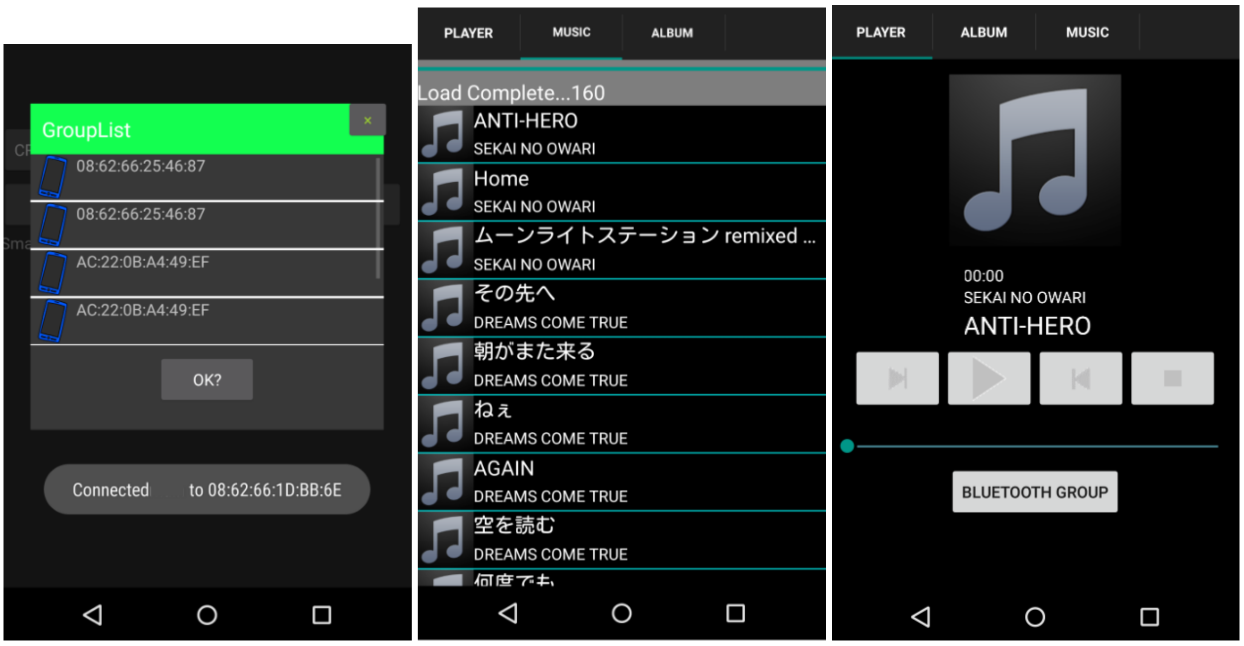


Figure 2. Screenshot of our Android music player. Left: search of other devices via Bluetooh, Center: playlist display, Right: music playback

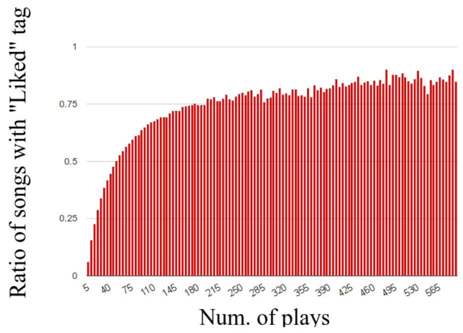


Figure 3. The ratio of songs with the "Liked" tag with respect to the numbers of plays.

proposed method. At the same time, the degrees of preference for the same songs were calculated using Equation (1) or (2) and were regarded to be the quasi ground truth.

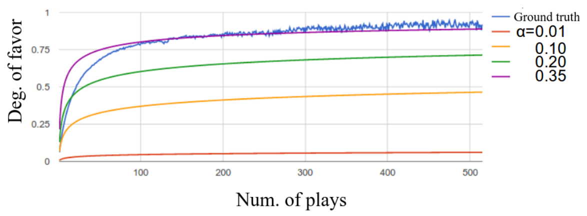
3.3.3 *Results*

The correlations between the degrees of preference calculated using our method and the quasi ground truth are listed in Table 1. These data show that the correlation is approximately 0.6 with almost any parameters and therefore that the degree of preference is fairly appropriately estimated. Figure 5 shows the correlations for each user. The correlations were higher than 0.5 for roughly half of the users.

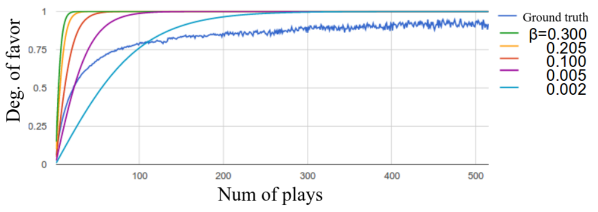
4. CONCLUSION

We have proposed an Android application that makes it possible to seamlesssly enjoy songs that are separately stored on different smartphones or devices. This application has two features. One feature is networked playback. The master device commands other devices to connect to or disconnect from the Bluetooth speaker. Users are accordingly freed from manually switching the device connection. The other feature is music selection. Using each user's playback history and musical similarity, the application estimates the degree of each user's preference even for songs that have not been listened to. Experimental results reveal that the degrees of preference estimated by our method are correlated with the quasi ground truth.

There is still a lot of future work. First, the current



(a) With Equation (1)



(b) With Equation (2)

Figure 4. Estimation of degree of preference for owned songs

(a) With Equation (1) for owned songs

α	0.01	0.1	0.2	0.3	0.4
Corr.	0.60548	0.60419	0.60217	0.5960	0.59658

(b) With Equation (2) for owned songs

β	0.002	0.005	0.100	0.205	0.300
Corr.	0.56625	0.56708	0.60060	0.60132	0.59658

Table 1. Estimation of degree of preference for unowned songs

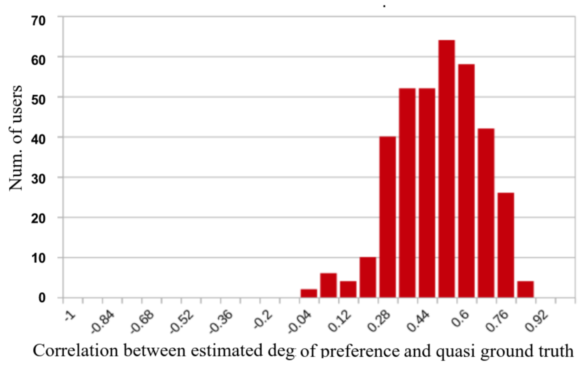


Figure 5. Histogram of per-user correlations between the estimated degree of preference and the quasi ground truth.

playlist generation process—simply placing songs in order of degree of preference—is too naive. We have to improve this process to maintain the users' satisfaction from the beginning to the end of the playlist. We should also conduct usability tests because experiments presented here were focused only on the music selection process. In addition, we plan to distribute this application on Google Play to obtain user feedback for further improvements.

5. ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Numbers 26240025, 26280089, 16K16180, 16H01744, 16KT0136, and 17H00749. We approiate Prof. Tomonobu Ozaki for his fruitful comments.

6. REFERENCES

[1] Implicit personalization of public environments using Bluetooth. In *Proceedings of CHI EA '08 Extended Abstracts on Human Factors in Computing Systems*, pages 3093–3098, 2008.

[2] J.-J. Aucouturier and F. Pachet. Tools and architecture for the evaluation of similarity measures: Case study of timbre similarity. In *Proc. ISMIR*, pages 198–203, 2004.

[3] M. Casey. General sound classification and similarity in mpeg-7. *Organaized Sound*, 6(2), 2002.

[4] A. Crossen and J. Budzik an K. J. Hammond. Flytrap: Intelligent group music recommendation. In *Proceedings of International Conference on Intelligent User Interfaces (IUI '02)*, pages 184–185, 2002.

[5] Pedro Dias and João Magalhães. Music recommendations for groups of users. In *Proceedings of the 2013 ACM international workshop on Immersive media experiences*, pages 21–24. ACM, 2013.

[6] Joseph F. McCarthy and Theodore 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 (CSCW '98)*, pages 363–372, 1998.

[7] Auste Piliponyte, Francesco Ricci, and Julian Koschwitz. Sequential music recommendations for groups by balancing user satisfaction. In *UMAP Workshops*, 2013.

[8] George Popescu and Pearl Pu. What's the best music you have? designing music recommendation for group enjoyment in GroupFun. In *Works-in-Progress of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*, pages 1673–1678, 2012.

A POST-PROCESSING PROCEDURE FOR IMPROVING MUSIC TEMPO ESTIMATES USING SUPERVISED LEARNING

Hendrik Schreiber
tagtraum industries incorporated
hs@tagtraum.com

Meinard Müller
International Audio Laboratories Erlangen
meinard.mueller@audiolabs-erlangen.de

ABSTRACT

Tempo estimation is a fundamental problem in music information retrieval and has been researched extensively. One problem still unsolved is the tendency of tempo estimation algorithms to produce results that are wrong by a small number of known factors (so-called octave errors). We propose a method that uses supervised learning to predict such tempo estimation errors. In a post-processing step, these predictions can then be used to correct an algorithm's tempo estimates. While being simple and relying only on a small number of features, our proposed method significantly increases accuracy for state-of-the-art tempo estimation methods.

1. INTRODUCTION

Tempo-related tasks are well established in *music information retrieval* (MIR) [1]. One common task is to estimate the *tempo* humans "tap" along to a beat when listening to music. Another task, *beat tracking*, attempts to determine the exact times at which beats occur. In this paper, we deal with tempo estimation exclusively. While in some genres—like Romantic music—local tempo changes are common [11], Pop, Rock, and Dance music often have one steady, global tempo, i.e. it can be represented by a single number usually specified in *beats per minute* (BPM). The method proposed in this paper is only suitable for music with such a global tempo.

Over the years, many different approaches to tempo estimation have been taken. Gouyon et al. [9] provided a comparative evaluation of the systems that participated in the ISMIR 2004 contest. Five years later, Zapata and Gómez gave an updated overview [30]. To our knowledge, the most recent comprehensive evaluations are presented in [2, 23, 24]. For a textbook-style introductory overview describing different approaches, see [20] by Müller.

Many methods divide the estimation problem into two phases. First, via an *onset strength signal* (OSS) or novelty curve, beat candidates are found. Second, one or more periodicities are extracted from the OSS. Methods for finding periodicities are often based on the Fourier transform, but also include autocorrelation [23], tempograms [29], the interonset interval (IOI) histograms [26], and resonating comb filters [2]. The decision for a final result is based on simple heuristics, genre classification [15, 25], secondary tempo estimates [24], the discrete cosine transform of IOI histograms [6], or a feature-based learning approach like *Gaussian mixture models* (GMM) [22], *support vector machines* (SVM) [8, 23], *k-nearest neighbor classification* (k-NNC) [29], and neural networks [5].

For evaluation, results are typically compared with a ground truth allowing a 4% tolerance. This measure is called *Accuracy1*. Because many algorithms have a tendency to under- or over-estimate the true value by a factor of 2 or 3, a second measure called *Accuracy2* has been introduced. *Accuracy2* allows for errors that correspond to a factor of 2, 3, ½, or ⅓, also known as *octave errors*. Despite evidence that algorithms as well as humans can distinguish between slow and fast music [13, 18], *Accuracy1* values for state-of-the-art algorithms are still below *Accuracy2*. One way to change this may lie in genre- or style-related knowledge [4]. Many genres are partially defined by a certain tempo or tempo range, which can be exploited to pick the right octave. Schuller et al. [25] demonstrated this for the Ballroom dataset and Hörschläger et al. [15] did the same for the GiantSteps tempo dataset. The fact that the excellent method by Böck et al. [2] scores remarkable 95% when trained on the Ballroom dataset with 8-fold cross validation, but reaches only 66.8% on the mixed genre GTZAN dataset further supports this notion.

In this paper we describe a supervised learning approach to correct common tempo estimation errors. This is achieved by re-framing error correction as a classification problem. We are able to demonstrate that the proposed method performs better or as well as state-of-the-art algorithms when combined with a simple tempo estimation method. Furthermore, because our error correction approach can be trained for any tempo detection method, we are able to show improvements in *Accuracy1* for previously published algorithms via post-processing.

The remainder of this paper is structured as follows: in Section 2 we describe a simple tempo estimation algorithm, test datasets, measures, and investigate common estimation error classes. Then, in Section 3, we explain our post-processing procedure, which corrects tempo estimates using supervised learning based on a small number of audio features. In Section 4 we evaluate the proposed features, and then compare our results with those from other methods. Finally, in Section 5, we present our conclusions.

2. TEMPO ESTIMATION

To lay the groundwork for our error correction method, we first describe a simple tempo estimation algorithm, then introduce several test datasets and discuss common pitfalls. In Section 2.5, we introduce performance metrics and describe observed errors.

2.1 Algorithm

To estimate the dominant pulse we follow the approach taken in [24], which is similar to [23, 28]: We first convert the signal to mono and downsample to 11025 Hz. Then we compute the power spectrum Y of 93 ms windows with half overlap, by applying a Hamming window and performing an STFT. The power for each bin $k \in [0:K] := \{0,1,2,\ldots,K\}$ at time $m \in [0:M] := \{0,1,2,\ldots,M\}$ is given by $Y(m,k)$, its positive logarithmic power $Y_{\ln}(m,k) := \ln(1000 \cdot Y(m,k)+1)$, and its frequency by $F(k)$ given in Hz. We define the onset signal strength $\mathrm{OSS}(m)$ as the sum of the bandwise differences between the logarithmic powers $Y_{\ln}(m,k)$ and $Y_{\ln}(m-1,k)$ for those k where the frequency $F(k) \in [30,720]$ and $Y(m,k)$ is greater than $\alpha Y(m-1,k)$ (see [16]):

$$
\begin{aligned}
I(m,k) &= \begin{cases} 1 & \text{if } Y(m,k) > \alpha Y(m-1,k) \\ & \text{and } F(k) \in [30,720], \\ 0 & \text{otherwise} \end{cases} \qquad (1)\\
\mathrm{OSS}(m) &= \sum_k \left(Y_{\ln}(m,k) - Y_{\ln}(m-1,k)\right) \cdot I(m,k)
\end{aligned}
$$

Both the factor $\alpha = 1.76$ and the frequency range were found experimentally [24].

The $\mathrm{OSS}(m)$ is transformed using a DFT with length 8192. At the given sample rate, this ensures a resolution of 0.156 BPM. The peaks of the resulting beat spectrum B represent the strength of BPM values in the signal [7], but do not take harmonics into account [10, 21]. Therefore we derive an enhanced beat spectrum B_{E} that boosts frequencies supported by harmonics:

$$
B_{\mathrm{E}}(k) = \sum_{i=0}^{2} |B(\lfloor k/2^i + 0.5 \rfloor)| \qquad (2)
$$

Similar to an enhanced beat histogram [28], B_{E} incorporates harmonics by simply adding to each bin the magnitudes of the bins denoted by half and by a quarter of its own frequency—or, if not available—the closest available bin. We choose to use fractions instead of multiples for modeling harmonics and thus essentially model the fourth harmonic, not the first. This allows us to take advantage of the full DFT resolution without oversampling, as each bin for the first harmonic is mapped to four different bins for the fourth harmonic. To estimate the tempo T of the dominant pulse (or periodicity in the OSS), we determine the

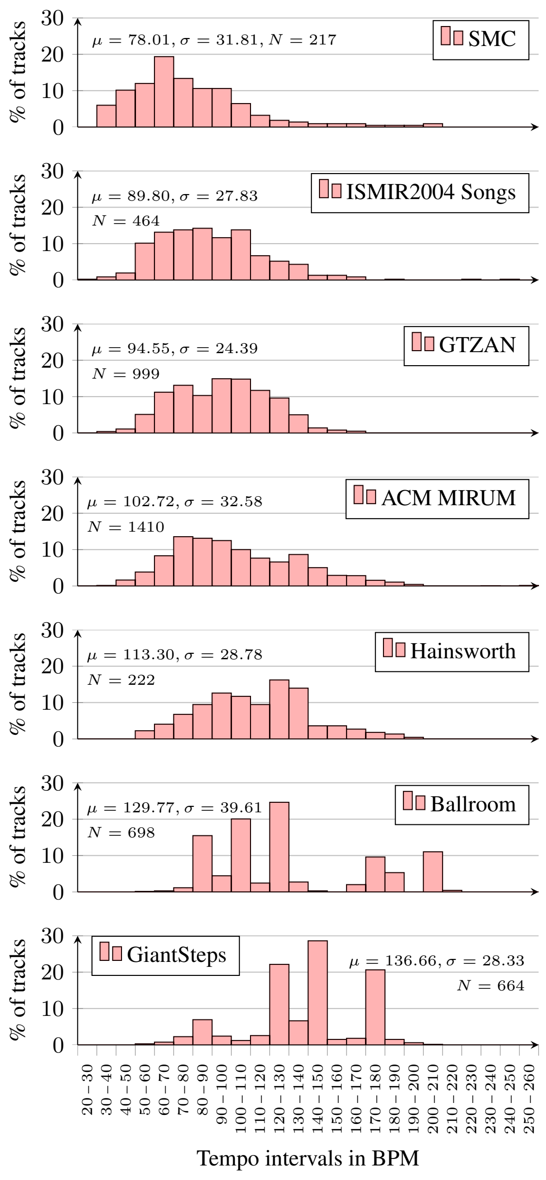


Figure 1. Tempo distributions for the test datasets.

highest value of B_{E}, divide its frequency by 4 to find the first harmonic, and finally convert its associated frequency to BPM:

$$
T = F(\underset{k}{\operatorname{argmax}}\ B_{\mathrm{E}}(k)) \cdot \frac{60}{4} \qquad (3)
$$

To ensure meaningful results for most kinds of Western music, we constrain T to $[40, 250]$ by halving or doubling its value, if necessary.

2.2 Test Datasets

It has become customary to benchmark tempo estimation methods with results reported for a small set of well known datasets. These are ACM MIRUM [18, 22], Ballroom [9], GTZAN [27], Hainsworth [12], ISMIR04 Songs [9], and SMC [14]. The latter was specifically designed to be diffi-

Dataset	E_0	E_1	E_2	$E_{1/2}$	E_3	$E_{1/3}$	E_4	$E_{1/4}$	$E_{3/2}$	$E_{2/3}$	$E_{5/4}$	$E_{4/5}$	$E_{4/3}$	$E_{3/4}$
ACM MIRUM	0.7	73.5	16.0	7.0	0.8	0.0	0.1	0.0	1.8	0.1	0.0	0.0	0.0	0.1
Ballroom	0.4	64.3	1.0	29.7	0.0	1.1	0.0	0.7	0.1	2.3	0.0	0.0	0.0	0.3
Hainsworth	8.6	64.4	1.4	19.4	0.0	0.5	0.0	0.0	1.8	1.8	0.9	0.5	0.5	0.5
GTZAN	4.0	72.2	15.7	5.1	0.4	0.1	0.0	0.0	1.0	0.4	0.1	0.4	0.5	0.1
ISMIR04 Songs	4.3	64.7	19.4	6.3	1.1	0.2	0.0	0.0	2.4	0.4	0.4	0.2	0.2	0.4
SMC	29.0	37.8	6.0	10.6	0.0	2.3	0.0	0.0	5.1	2.3	0.5	2.3	0.9	3.2
Combined	3.9	68.1	12.3	11.2	0.5	0.4	0.0	0.1	1.5	0.8	0.1	0.3	0.2	0.4
GiantSteps	7.1	63.1	4.1	21.5	0.0	0.0	0.0	0.0	0.6	0.3	0.8	0.9	0.5	1.2

Table 1. Error class distribution for tempo estimates T (given in BPM) for different datasets in percent.

Dataset	Sweet Oct.	Cov.	90%	95%
ACM MIRUM	69 − 138	72.8	50 − 152	50 − 170
Ballroom	71 − 142	71.1	84 − 204	82 − 204
Hainsworth	79 − 158	82.4	58 − 150	57 − 167
GTZAN	66 − 132	80.9	55 − 130	52 − 138
ISMIR04 Songs	59 − 118	74.1	48 − 131	36 − 136
SMC	51 − 102	68.7	32 − 115	32 − 143
Combined	69 − 138	72.9	40 − 150	50 − 180
GiantSteps	91 − 182	88.1	85 − 175	80 − 180

Table 2. Sweet octaves and their respective coverage in percent for the test datasets (left). Shortest BPM intervals required to achieve a test set coverage of 90% or 95% (right).

cult for beat trackers. Where applicable, we used the corrected annotations from [23]. We refer to the union of these six datasets as the *Combined* dataset. Additionally, we test against the recently published GiantSteps dataset for electronic dance music (EDM) [17]. It is not included in *Combined* to allow direct comparisons with older literature.

Not surprisingly, all mentioned datasets differ in their composition (Figure 1). The mean tempo ranges from 78 BPM (SMC) to 137 BPM (GiantSteps) and the standard deviation spans from 24 (GTZAN) to 40 (Ballroom). Furthermore, the tempo distributions of Ballroom and GiantSteps contain some distinct spikes, while the other datasets more closely resemble normal distributions. None of the datasets have uniformly distributed tempi.

2.3 Octave Bias

If a dataset's tempo distribution is not uniform and most values fall into a relatively small interval, constraining results to this interval may lead to fewer octave errors. We call deliberately choosing such an interval *octave bias*.

To illustrate this, assume an algorithm for the GiantSteps dataset with 50% *Accuracy1*, but 100% *Accuracy2*. Further assume that all errors are by a factor of 2 or $1/2$. 88.1% of all tempi in GiantSteps happen to be in $[91, 182)$. If we constrained results to this interval by halving and doubling, *Accuracy1* would increase from 50% to 88.1%.

Each described dataset has such a *sweet octave*, i.e. a tempo interval $[j, 2j)$ that contains more of the dataset's songs than any other octave (Table 2). In the absence of a uniform test set, it is therefore important to test the same algorithm against datasets with different sweet octaves, thus revealing the effects of octave bias. On the positive side, a specialized or genre-aware algorithm may benefit from exploiting knowledge about the test dataset (e.g. EDM-specific tempi [15]). Additionally to sweet octaves, Table 2 lists the shortest BPM intervals required to achieve a certain test set coverage. For example, to cover 90% of the tempi in the ACM MIRUM test set, one only needs to look at the interval $[50, 152]$ and not at the considerably larger interval $[37, 257]$ required for full coverage.

2.4 Genre Bias

While octave bias describes how algorithms can exploit constraining results to certain tempo intervals, *genre bias* describes a technique for algorithms to constrain their output to a relatively small set of distinct tempi that are characteristic for the genres in the dataset.

A good example for this is the Ballroom dataset. Even though the dataset contains 698 songs, only 63 different tempi occur. Assuming an unbiased algorithm with integer precision is constrained to the $[40, 250]$ BPM interval, it solves a task equivalent to choosing one out of 210 classes. An algorithm trained on the Ballroom dataset using k-fold cross validation "knows" that there are only 63 classes and therefore has a considerably easier task to solve.

2.5 Measures

As mentioned above, tempo estimation algorithms are usually evaluated with two metrics: *Accuracy1*, defined as the percentage of correct estimates with 4% tolerance, and *Accuracy2*, the percentage of correct estimates ignoring errors caused by the factors 2, 3, $1/2$, and $1/3$.

Because we aim to correct estimation errors, we need to test our tempo estimation method against the test datasets and record not just accuracies, but also the kinds of errors. To do so, we define the error classes E_2, E_3, E_4, $E_{3/2}$, $E_{5/4}$, $E_{4/3}$ and their reciprocals with the index indicating the error factor. Just like *Accuracy1* and *Accuracy2* we allow a 4% tolerance. Since not all estimates are wrong and some errors are not covered by the mentioned classes, we define E_1 for correct estimates (equivalent to *Accuracy1*) and E_0 for errors not described otherwise. This leads to a total of 14 classes forming the label set $\mathbb{E} := \{E_0, E_1, \ldots\}$. Table 1 shows the distribution of estimated tempi over $\mathbb{E}$ for the test datasets using the tempo estimation method from Section 2.1. For *Combined*, 12.3% of all tempi are in E_2, while 11.2% are in $E_{1/2}$. Only 3.9% of all estimated values cannot be explained by one of the defined factors and thus are collected under the label E_0. This implies an upper bound of 96.1% *Accuracy1* for any error correction scheme based on $\mathbb{E}$ w.r.t. the *Combined* datasets.

3. TEMPO ERROR CORRECTION

As we have seen, most wrong tempo estimates are off by a limited number of factors. Therefore the correction of T can be re-framed as a classification problem, which is solvable using supervised machine learning. Knowing the error class for an estimate then allows us to calculate the true tempo. In the following subsections we describe the features used for classification, the training dataset, and the tempo correction procedure.

3.1 Features

In order to keep the algorithm simple, we use as features only T and a very small set of audio features. While not attempting to specifically model genres, the features we use aim at characterizing rhythm, tonality and beat intensity. Combined, we expect them to capture essential information about a musical piece.

3.1.1 Log Beat Spectrum

The tempi corresponding to the most common estimation classes $E_{1/2}$, E_1, and E_2 fall onto a logarithmic scale. To mirror this, we use a *logarithmic beat spectrum* (LBS) to describe the different periodicities in the signal. LBS is computed by resampling/interpolating B into 10 logarithmically spaced bands representing tempi ranging from 40 to 500 BPM. Subsequently it is normalized so that its highest value is 1.

While LBS provides a more complete picture of the periodicities than just the dominant tempo T, it does not add any information about the frequency bands these periodicities stem from. As a second modification to B, we create different versions of LBS based on the five slightly overlapping bands $[30, 110]$, $[100, 220]$, $[200, 440]$, $[400, 880]$ and $[800, 1600]$ Hz. Combined, these spectra form the *multiband logarithmic beat spectrum* (LBS$_\mathrm{M}$). For a given song, LBS$_\mathrm{M}$ consists of $5 \times 10 = 50$ features.

3.1.2 Spectral Flatness

To represent tonality we use *spectral flatness* (SF), also known as Wiener entropy. It is defined as the ratio between the geometric and the arithmetic mean of the power spectrum:

$$\mathrm{SF}(m) \quad = \quad \frac{\left(\prod_{k=0}^{K-1} Y(m,k)\right)^{\frac{1}{K}}}{\frac{1}{K}\sum_{k=0}^{K-1} Y(m,k)} \qquad (4)$$

To determine $\mathrm{SF}(m)$, we re-use the power spectra $Y(m, k)$ already computed in Section 2.1. For increased robustness against low sample rates, we limit k to $F(k) \in [30, 3000]$. As the two features for a given song we use both the mean and the variance of all its $\mathrm{SF}(m)$.

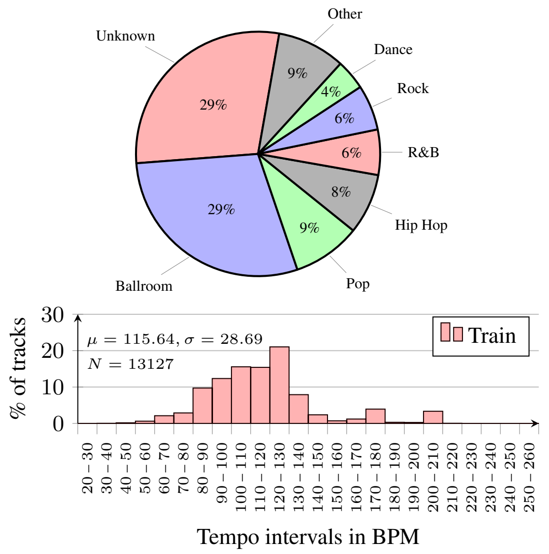


Figure 2. Distribution of genres and tempi for *Train*.

3.1.3 Temporal Flatness

To represent onset intensity we use a feature called *temporal flatness* (TF). Instead of calculating the Wiener entropy along the frequency axis of $Y(m, k)$, as we did for SF, we calculate it over a window of length ℓ along the time axis:

$$\mathrm{TF}(m,\ell,k) \quad = \quad \frac{\left(\prod_{i=0}^{\ell-1} Y(m+i,k)\right)^{\frac{1}{\ell}}}{\frac{1}{\ell}\sum_{i=0}^{\ell-1} Y(m+i,k)} \qquad (5)$$

To compute TF values, we again re-use Y and limit k to $F(k) \in [30, 3000]$. Y is split into non-overlapping windows with length $\ell = 100$. For each bin k in a given window we compute TF. We then calculate the average $\mathrm{TF}_\mathrm{W}(m, \ell)$ over all k. As the two features for a song we use the mean and the variance of all its $\mathrm{TF}_\mathrm{W}(m, \ell)$ values.

3.2 Training Dataset

To avoid learning the test datasets, we use a dataset for training the classifier that has been created separately. *Train* is the union of an annotated, private music collection and the Extended Ballroom dataset [19] minus the 354 songs also occurring in the regular Ballroom set. Genre labels are available for 71% of the recordings. The genre as well as the tempo distribution are shown in Figure 2.

3.3 Correcting the Tempo Estimate

Differences between the training dataset's true tempo values and the estimated tempo values let us derive error class labels. With those and the proposed features, we can train a classifier. Using the classifier, we are then able to predict an estimated error class $E \in \mathbb{E}$ for any song for which we also have features and a tempo estimate. Note that this estimate does not have to stem from our own algorithm introduced in Section 2.1. One main idea of this paper, indeed,

Features	*Accuracy1*	*Accuracy2*
base	69.00-	**94.31**
LBS	75.84-	94.16
LBS + SF	76.68	94.09
LBS + TF	**77.41**	94.11
LBS + SF + TF	77.31	94.04
LBS_M	76.66	93.87
LBS_M + SF	75.91-	94.21
LBS_M + TF	77.31	93.97
LBS_M + SF + TF	76.21	94.14

Table 3. *Accuracy1* and *Accuracy2* for different feature combinations trained on *Train* and tested against *Combined*. The '−' signs indicate a statistically significant difference between the marked results and LBS + TF.

is that the classification model is algorithm-specific. In other words, the classifier must be trained for each tempo estimation algorithm. Once trained, it can be used to correct octave errors inherent to the given tempo estimation algorithm.

For the prediction process itself, we use a Random Forest [3] with 300 trees and a maximum depth of 25.

Given the estimated tempo T and the predicted error E the calculation of the corrected tempo $T_{\text{corrected}}$ is straight forward:

$$\begin{aligned} J(T,E) &= \begin{cases} 1 & \text{if } E = E_0 \\ i & \text{for } E_i \end{cases} \qquad (6) \\ T_{\text{corrected}} &= T \cdot J(T,E) \end{aligned}$$

4. EVALUATION

In a first evaluation step, we compute *Accuracy1* for different feature combinations. We then compare the best combination with publicly available algorithms as well as other simple correction schemes.

4.1 Feature Evaluation

We trained the classifier using the dataset *Train* with different combinations of the proposed audio features and measured the performance against the dataset *Combined*. All tested feature combinations clearly outperformed the baseline algorithm base (T without correction) by at least 6pp (percentage points) for *Accuracy1* (Table 3). As was to be expected, *Accuracy2* didn't change significantly. The best performing feature combination was LBS + TF with an *Accuracy1* of 77.41%. When testing for significance with McNemar's test and a significance level of $p < 0.01$ [30], we found that LBS + TF performed significantly better w.r.t. *Accuracy1* than LBS, LBS_M + SF, and base. In the following we refer to the error classifier trained with LBS + TF as new. If no tempo estimation algorithm is explicitly mentioned, the method from Section 2.1 is otherwise implied.

4.2 Comparative Evaluation

We compared our method base+new to its baseline base and the three publicly available algorithms böck, stem, and schr using the test datasets described in Section 2.2.

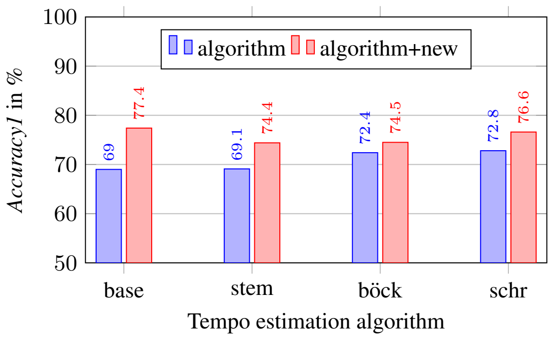


Figure 3. *Accuracy1* for *Combined* for different algorithms with and without new error correction. All algorithms reach significantly higher scores when combined with new.

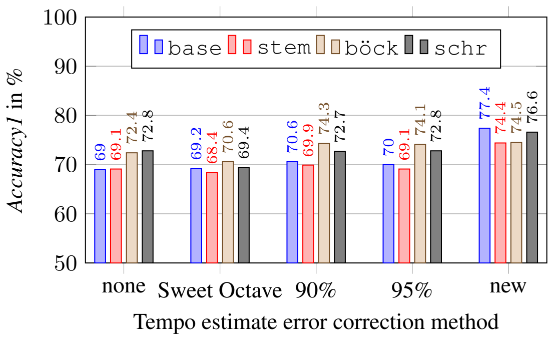


Figure 4. *Accuracy1* for *Combined* using no error correction, constraint-based correction, and new correction for various tempo estimation algorithms.

böck [1] is the algorithm published by Böck et al. in [2], but trained with different datasets—among them our test data, i.e. the algorithm is "familiar" with the test sets. According to the authors, this configuration participated in MIREX 2016. stem is an algorithm aiming for low computational complexity published by Percival et al. [23]. We used the implementation contained in Marsyas 0.5.0. [2] Lastly, schr [3] was published by Schreiber et al. in [24]. Since our error estimation and correction method can be used as a post-processor for any tempo estimator, we also trained the classifier for each of these three tempo estimation algorithms to investigate potential improvements.

Figure 3 shows the *Accuracy1* results for the four algorithms when tested against *Combined*, both with and without new post-processing. All of them score significantly higher values when combined with new than in their plain form (McNemar, $p < 0.01$). The algorithms base (77.4%, increase of +8.4pp) and stem (74.7%, +5.3pp) clearly benefit the most, but also böck (74.5%, +2.1pp) and schr (76.6%, +3.9pp) gain several percent-

[1] `https://github.com/CPJKU/madmom/`
[2] `http://marsyas.info/`
[3] `http://www.tagtraum.com/download/schreiber_icassp2014.zip`

Dataset	base	base+new	stem	stem+new	böck	böck+new	schr	schr+new
ACM MIRUM	73.6	**81.8+**	74.3	79.9+	74.5	76.1+	76.3	81.3+
ISMIR04 Songs	65.5	69.2	60.1	62.3	55.4	58.4+	**74.1**	70.7-
Ballroom	64.3	85.1+	64.0	81.8+	84.8	**90.4+**	67.0	85.0+
Hainsworth	64.9	73.4+	69.8	74.8+	84.2	**85.6**	73.0	75.7
GTZAN	73.5	**78.8+**	77.9	76.9	70.7	71.1	78.0	76.2
SMC	45.2	39.6	29.5	29.5	51.1	**51.6**	41.5	35.5-
Dataset Average	64.5	71.3	62.6	67.5	70.1	**72.2**	68.3	70.7
Combined	69.0	**77.4+**	69.1	74.4+	72.4	74.5+	72.8	76.6+
GiantSteps	64.5	64.0	47.0	65.0+	61.5	**70.9+**	58.0	60.1
GiantSteps+Combined	68.4	**75.5+**	66.0	73.1+	70.8	74.0+	70.7	74.3+

Table 4. Tempo results for *Accuracy1* in percent. The '+' and '−' signs indicate a statistically significant difference between an algorithm and the same algorithm enhanced with new. Bold numbers mark the best-performing algorithm(s) for a dataset. *Dataset Average* is the mean of the algorithms' results for each dataset except GiantSteps.

age points.

As discussed in Section 2.3, a simple error correction scheme can be based on octave bias exploiting statistical properties of the test dataset. We therefore compared our method with such a constraint-based scheme, where tempi below a lower interval bound are doubled and above an upper bound are halved. For intervals we used the sweet octave and those listed in Table 2 with 90% and 95% coverage. Results are shown in Figure 4. Except for böck, none of the algorithms benefitted much from the simple correction—perhaps a certain bias is already built-in. When comparing böck+new and böck+90% we were not able to observe a significant difference. It appears, as if böck's octave errors are harder to predict and correct than those of the other algorithms, perhaps because they are less systematic in nature.

Table 4 provides a detailed overview of *Accuracy1* results for each of the test algorithms for all test datasets. As mentioned, base+new reaches the highest score for the *Combined* dataset (77.4%). To the best of our knowledge, this is the highest *Accuracy1* score reported for *Combined* to date. For four of the six *Combined* datasets, base+new reaches significantly higher values than base (indicated by '+' signs in Table 4). The largest improvement was achieved for the Ballroom test set. The score for base+new is more than 20pp higher than base's. The fact that 29% of *Train* consists of ballroom tracks certainly plays a role here. While the base+new score for ISMIR04 Songs is 3.7pp higher than base's, the improvement is not significant. Similarly, the change for the SMC dataset (−5.6pp) is not significant, but noteworthy. We believe that both octave and genre bias may play a role here. Tracks in SMC are very different in style from those in *Train*. And compared to SMC, *Train* contains relatively few examples for slow tracks with 60 BPM or less. Informal tests confirm that choosing a different training dataset leads to better results.

Dataset-specific scores for böck+new are all higher than those for böck—more than half of them significantly. The largest increase can be observed for the GiantSteps dataset. Plain böck scores 61.5%—combined with new it reaches 70.9% (+9.4pp). To the best of our knowledge, this is the highest reported value for an unbiased, non-commercial algorithm to date.[4]

Dataset Average is the mean of the results for each of the six datasets in *Combined*. Because it is an unweighted average, it is not dominated by the larger datasets. But just like for *Combined*, we can observe higher scores for all algorithms when combined with new. With 72.2% (+2.1pp) böck+new reaches the highest score, closely followed by base+new with 71.3% (+6.8pp). stem (67.5%, +4.9pp) and schr (70.7%, +2.4pp) benefitted as well.

Though not the topic of this paper, we also measured *Accuracy2*. As expected, the results did not surprise and stayed stable.

5. CONCLUSIONS

We have shown that the proposed error correction method based on supervised learning of tempo estimation errors is capable of significantly improving *Accuracy1* results for existing tempo estimation algorithms. It does so in an algorithm-specific post-processing step. Combined with a simple tempo estimation algorithm, it outperforms other state-of-the-art algorithms for most of the tested datasets. We believe the error correction method can be enhanced even further by carefully selecting and incorporating other genre-related features.

We also discussed different kinds of biases that can have a large influence on the accuracy of tempo estimation algorithms. Ideally, evaluations of general purpose tempo estimators should be based on datasets with a mostly uniform tempo and genre distribution. Because better training data potentially leads to better results, training datasets should be an integral part of the comparison to make fair benchmarking possible. Defined train/test splits for existing datasets could be a first step in this direction.

Additional Material:
Binaries and other material are available at http://www.tagtraum.com/tempo_estimation.html.

Acknowledgments:
The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

[4] http://www.cp.jku.at/datasets/giantsteps/

6. REFERENCES

[1] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[2] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 625–631, Málaga, Spain, 2015.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Nick Collins. Towards a style-specific basis for computational beat tracking. In *Proceedings of the 9th International Conference on Music Perception and Cognition (ICMPC9) and 6th Triennial Conference of the European Society for the Cognitive Sciences of Music (ESCOM)*, Bologna, Italy, 2006.

[5] Anders Elowsson. Beat tracking with a cepstroid invariant neural network. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, pages 351–357, New York, NY, USA, 2016.

[6] Anders Elowsson and Anders Friberg. Modeling the perception of tempo. *The Journal of the Acoustical Society of America*, 137(6):3163–3177, 2015.

[7] Jonathan Foote and Shingo Uchihashi. The beat spectrum: A new approach to rhythm analysis. In *Proceedings of the International Conference on Multimedia and Expo (ICME)*, Los Alamitos, CA, USA, 2001.

[8] Aggelos Gkiokas, Vassilios Katsouros, and George Carayannis. Reducing tempo octave errors by periodicity vector coding and svm learning. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 301–306. FEUP Edições, 2012.

[9] Fabien Gouyon, Anssi P. Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[10] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram – a mid-level tempo representation for music signals. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5522 – 5525, Dallas, Texas, USA, 2010.

[11] Peter Grosche, Meinard Müller, and Craig Stuart Sapp. What makes beat tracking difficult? A case study on Chopin Mazurkas. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 649–654, Utrecht, The Netherlands, 2010.

[12] Stephen Webley Hainsworth. *Techniques for the Automated Analysis of Musical Audio*. PhD thesis, University of Cambridge, UK, September 2004.

[13] Jason Hockman and Ichiro Fujinaga. Fast vs slow: Learning tempo octaves from user data. In *Proceedings of the 11th International Conference on Music Information Retrieval (ISMIR)*, pages 231–236, Utrecht, The Netherlands, 2010.

[14] Andre Holzapfel, Matthew EP Davies, José R Zapata, João Lobato Oliveira, and Fabien Gouyon. Selective sampling for beat tracking evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2539–2548, 2012.

[15] Florian Hörschläger, Richard Vogl, Sebastian Böck, and Peter Knees. Addressing tempo estimation octave errors in electronic music by incorporating style information extracted from wikipedia. In *Proceedings of the Sound and Music Computing Conference (SMC)*, Maynooth, Ireland, 2015.

[16] Anssi P. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3089–3092, Washington, DC, USA, 1999.

[17] Peter Knees, Ángel Faraldo, Perfecto Herrera, Richard Vogl, Sebastian Böck, Florian Hörschläger, and Mickael Le Goff. 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)*, pages 364–370, Málaga, Spain, October 2015.

[18] Mark Levy. Improving perceptual tempo estimation with crowd-sourced annotations. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 317–322, 2011.

[19] Ugo Marchand and Geoffroy Peeters. The extended ballroom dataset. In *Late Breaking Demo of the International Conference on Music Information Retrieval (ISMIR)*, New York, NY, USA, 2016.

[20] Meinard Müller. *Fundamentals of Music Processing – Audio, Analysis, Algorithms, Applications*. Springer Verlag, 2015.

[21] Geoffroy Peeters. Template-based estimation of time-varying tempo. *EURASIP Journal on Advances in Signal Processing*, 2007(1):158–158, 2007.

[22] Geoffroy Peeters and Joachim Flocon-Cholet. Perceptual tempo estimation using GMM-regression. In *Proceedings of the second international ACM workshop*

on Music information retrieval with user-centered and multimodal strategies, MIRUM '12, pages 45–50, New York, NY, USA, 2012. ACM.

[23] Graham Percival and George Tzanetakis. Streamlined tempo estimation based on autocorrelation and cross-correlation with pulses. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):1765–1776, 2014.

[24] Hendrik Schreiber and Meinard Müller. Exploiting global features for tempo octave correction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 639–643, Florence, Italy, 2014.

[25] Björn Schuller, Florian Eyben, and Gerhard Rigoll. Tango or waltz?: Putting ballroom dance style into tempo detection. *EURASIP Journal on Audio, Speech, and Music Processing*, 2008:12, 2008.

[26] Jarno Seppänen. Tatum grid analysis of musical signals. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 131–134, 2001.

[27] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[28] George Tzanetakis and Graham Percival. An effective, simple tempo estimation method based on self-similarity and regularity. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vancouver, Canada, 2013.

[29] Fu-Hai Frank Wu and Jyh-Shing Roger Jang. A supervised learning method for tempo estimation of musical audio. In *Control and Automation (MED), 2014 22nd Mediterranean Conference of*, pages 599–604. IEEE, 2014.

[30] Jose R. Zapata and Emilia Gómez. Comparative evaluation and combination of audio tempo estimation approaches. In *42nd AES Conference on Semantic Audio*, Ilmenau, Germany, 2011.

A STATISTICAL ANALYSIS OF *GAMAKA*S IN CARNATIC MUSIC

Venkata Subramanian Viraraghavan[1,2] **R Aravind**[2] **Hema A Murthy**[3]
[1] TCS Research and Innovation, Embedded Systems and Robotics, Bangalore, India
[2] Department of Electrical Engineering, Indian Institute of Technology, Madras
[3] Department of Computer Science and Engineering, Indian Institute of Technology, Madras
venkatasubramanian.v@tcs.com, aravind@ee.iitm.ac.in, hema@cse.iitm.ac.in

ABSTRACT

Carnatic Music, a form of classical music prevalent in South India, has a central concept called *rāga*s, defined as melodic scales and/or a set of characteristic melodic phrases. These definitions also account for the continuous pitch movement in *gamaka*s and micro-tonal adjustments to pitch values.

In this paper, we present several statistics of *gamaka*s to arrive at a model of Carnatic music. We draw upon the two-component model of Carnatic Music, which splits it into a slowly varying 'stage' and a detail, called 'dance'. Based on the statistics, we propose slightly altered definitions of two similar components called constant-pitch notes and transients. An automated implementation of these definitions is used in collecting statistics from 84 concert renditions.

We then suggest that the constant-pitch notes and transients can be considered as context and detail respectively of the *rāga*, but add that both are necessary for defining the *rāga*. This is verified by performing listening tests on only the constant-pitch notes and transients independently.

1. INTRODUCTION

Carnatic music, a classical form of music prevalent in South India, has evolved into a sophisticated art and a professional field. It uses *svara*s that roughly correspond to the twelve notes of the Western-music scale. *Svara*s are defined with respect to a base pitch called the *ādhāra ṣaḍja*, or the tonic. A subset of the 12 notes are played in specific orders and grouped together as phrases. These scales and phrases are used in defining *rāga*s.

Indian musicians use the regions 'between the notes' as a means of introducing sophistication over monophonic music. This can be seen in Figure 1 (reproduced with permission from [6]). The scales are thus notes connected by *gamaka*s that traverse frequencies between them; they can glide over even intermediate notes to reach notes further away.

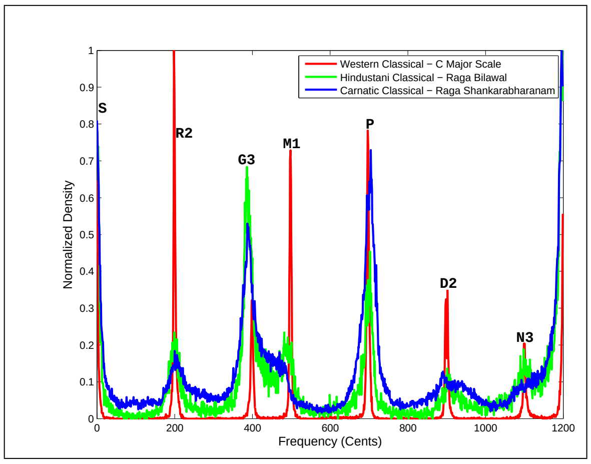


Figure 1: Histogram of notes of Carnatic, Hindustani and Western music renditions. Notice notes in Indian music span the regions between the notes of Western music.

The term *svara*, initially named the seven major notes in an octave – Sa, Ri (Carnatic)/Re (Hindustani), Ga, Ma, Pa, Da/Dha and Ni respectively corresponding to Do, Re, Mi, Fa, So, La and Ti in Western music. However, it has now expanded in meaning to identify even the transients between a sequence of notes.

The concepts of *gamaka* and *rāga* are central to Indian classical music. There is a perceived identity of *rāga*s that needs to be preserved. The characteristics of a *rāga* are determined in terms of its melodic phrases. Our observation is that a melodic phrase can be thought of as being made up of a sequence of constant-pitch notes (or CP-notes), and transients. Specific combinations of CP-notes and trajectories of the transients together give a *rāga* its unique identity. Further, we show that the constant-pitch notes serve as a context to interpret the transients in *gamaka*s. While the focus is primarily on Carnatic music, the methodology described could perhaps be extended to other genres that include significant continuous pitch movement, such as Hindustani music and possibly Jazz music.

The rest of the paper is organized as follows. Section 2 lists previous work and in Section 3, an extensive analysis of CP-notes and transients from concert recordings motivates our proposed model. Relevant previous work is described in terms of our model in Section 4.1. The results of a listening experiment are presented in Section 4.2 to verify the model and Section 5 concludes the paper.

2. PREVIOUS WORK

One of the earliest attempts of analyzing *gamaka*s in Carnatic music using signal processing techniques was [16]. After an initial analysis of a few *rāga*s, the author proposed the use of 'melodic atoms' to describe music. Again, as in the case of *gamaka*s, these are intuitive to the Carnatic musician, but there was no automated way of extracting them. At a similar time, *gamaka*s were analyzed for music information retrieval in [22]. This used the concept of 'extrema' of the pitch contour, which were called 'stationary points' in [8] for discovering *rāga* motifs.

Given the importance of *gamaka*s as borne out by Figure 1, it is not surprising that they were the subject of analysis in the CompMusic project [2]. An initial attempt was based on several types of *gamaka*s identified by musicologists [5]. A Carnatic musician, especially an instrumentalist, can intuitively relate to the descriptions of the movements, but looking for these pitch patterns in audio samples was not fruitful. Instead, *rāga* motifs (e.g. [8,14]) were identified. *Rāga* motifs can be seen as signature pitch contour movements of a *rāga* and occur only in that *rāga* and no other. This exclusivity is an important aspect because, in Carnatic music, *rāga*s do share many common phrases and the unique phrases are expected to be small in number. A more recent, completely automated method of *rāga* identification based on motifs is presented in [13].

In [21], we find manual analysis of Carnatic music *gamaka*s, mainly for use in synthesis. However, it is based on one rendition of one *varṇam*. At a high level, it views Carnatic music as two components: the stage, which has a slowly varying pitch-contour, and the dance – a detail on top of the stage. We show in this paper that the variations in detail are quite short (typically under 400 ms) and propose different definitions of the two components so that automatic analysis is possible. In fact, [10] extends the stable part of the pitch contour to segments of varying pitch for Hindustani music retrieval; however, it is incomplete because significant information about (especially Carnatic) *rāga*s is in the segments of audio where the pitch varies (as we will show in Figure 5 later). A more recent result [11] uses quantized forms of these segments for Hindustani music, but does not characterize the variations.

In Section 4.2, the two-component theory, motifs and the context-detail aspects of Carnatic music will be revisited in the light of the model we propose.

3. ANALYSIS OF *GAMAKA*S IN CONCERT RECORDINGS

We analyzed 84 Carnatic music concert pieces in seven *rāga*s. The *rāga*-wise split of these pieces is given in Table 1. They are from the CompMusic database [1]. We chose seven *rāga*s from those considered as the major in Carnatic music [4] – *tōḍī*, *bhairavī*, *kharaharapriyā*, *kāmbhōji*, *śankarābharaṇam*, *varāḷī*, and *kalyāṇī*.

Rāga	Number of pieces	
	Male	Female
Tōḍī	7	5
Bhairavī	6	6
Kharaharapriyā	7	5
Kāmbhōji	8	4
Śankarābharaṇam	6	6
Varāḷī	7	5
Kalyāṇī	9	3

Table 1: *Rāga*-wise split of the 84 vocal-concert renditions.

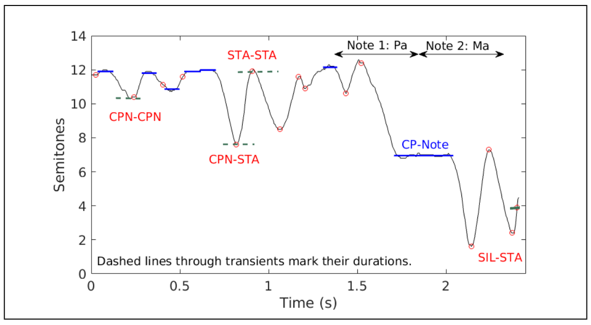


Figure 2: Examples of CP-notes (solid lines) and four types of stationary points (unfilled circles). The two notes marked exemplify EOb3.

3.1 Some Definitions

3.1.1 Constant-pitch Notes

Constant-pitch notes serve as a starting point for various musical forms including Carnatic music and we provide a working definition below, where the prefix EOb stands for 'Empirical Observation':

EOb1 A constant-pitch note is one whose pitch does not vary from its mean pitch by more than Δ semitones and lasts for at least $C_{\min}$ seconds.

where, nominally, $\Delta = 0.3$ and $C_{\min} = 80$ milliseconds (ms). The value of 80 ms is intuitive: it is the length of the shortest note played without *gamaka*s, in Carnatic music pieces that are extremely fast (*gamaka*s are not perceived at these speeds). This is possible on instruments for *rāga*s such as *mōhanam* (e.g. [3]) and *hamsadhwani*. Such renditions from a personal collection were analyzed. The duration of a CP-note at the fastest speed was found to be around 87 ms in one rendition and 80 ms in another. Only $\Delta = 0.3$ was consistent with this observation. For example, with $\Delta = 0.1$, the shortest CP-notes lasted more than 200 ms. Note that with other choices for Δ and $C_{\min}$, while the details of the statistics obtained may vary, the meaningful split of Carnatic music into CP-notes and transients is expected to hold.

3.1.2 Stationary Points

Stationary points were defined and used extensively in [7, 9, 22]. These are pitch positions where a continuous pitch curve changes direction. An example of these two features is shown in Figure 2. In the figure, a CP-note is easily recognized as a solid line drawn for its duration at its mean pitch value. The stationary points are marked by unfilled circles. (Note that statements 7 to 14 in Algorithm 2 remove redundant stationary points.) The same figure also shows a region of silence at the end of the time-axis. During silence, there is no prominent melodic pitch (i.e. neither of the voice nor of the accompanying violin), but a drone may be present. Four out of the six possible types of pitch curves – called transients – are marked in the figure. These types are defined by the neighbours of a stationary point and are in fact, the six combinations possible from the set {CPNote, STAtionary point, SILence} (the letters in upper-case are the short-forms used later) :

1. CP-note on one side of the transient and, on the other, one of {Another CP-note; A curve, which has a stationary point; Silence}.
2. A curve with a stationary point on one side of the transient, and on the other, one of {A curve with its own stationary point, Silence}.

Our analysis counts the number of occurrences of each of the above and is presented next [1].

3.2 Method

For each of the renditions in Table 1, the prominent pitch and tonic extracted for the work reported in [12] were reused. They had been found using the algorithms described in [18], with a window length of 46 ms and a window shift of 2.9 ms [19]. Sometimes, the tracked pitch could be that of accompanying instruments and is prone to some octave-errors. However, unlike in transcription (e.g. [17]), these errors are inconsequential in identifying the type and duration of each transient (Section 3.1.2) or CP-note. The only impact was at most the splitting of a CP-note into more than one. Manual inspection of several pieces and their pitch tracks suggested that this occurred infrequently enough to be neglected.

Algorithm 1 was used to identify CP-notes and Algorithm 2, stationary points [2]. Each stationary point is also marked with one of the types described in Section 3.1.2.

Statements 4 to 9 of Algorithm 1 look for the longest possible CP-note from the current pitch value (at index i). However, in statement 6 , a direct application of the definition of CP-notes (Section 3.1.1) resulted in too many false alarms. For example, if the pitch rose by 0.6 semitones within 80 ms, but was part of a continuous pitch movement, it would still get detected as a CP-note. To eliminate these cases, a threshold of one semitone per second was used on the slope of the best-fit line. This is one-tenth the nominal slope of continuous pitch variation reported in [20].

The utility of defining CP-notes also emerges from Figures 3 and 4. Both *rāga*s show that the distribution of CP-notes is much sharper than all notes put together. (Ignore the stationary- and remaining-points for now.) For example, in Figure 3, notice when the notes are all plotted together (dashed line), they show a significant value (defined as occurring more often than 2% of the maximum occurrence) over G2 (3 semitones from Sa). However, the CP-notes show no such peak. This means there is no CP-note peak masked by the flat sections of the histogram. Further, there are no peaks found at 'incorrect' locations, say at 9.5 semitones. In fact, this behavior is sufficiently consistent to attach the name of the nearest semitone to the *svara*. That is, in a system of music with continuous pitch variation, *whether the CP-notes occur as part of svara*s *with gamaka*s *or without*, the CP-notes cluster around the *rāga*'s scale notes. This result is intuitive, but it is not obvious from, say, Figure 1.

In Figure 3, the automatically identified *svara*s are a subset of the written *ārohaṇa* and *avarohaṇa* of *śankarābharaṇam*. An interesting exception occurs in the *rāga tōḍī* (Figure 4) where the peak at R2 (2 semitones from Sa), is not part of its *ārohaṇa* and *avarohaṇa*.

[1] Silence on both sides of the transient was seen in under 0.005% of the cases and is excluded.

[2] A more sophisticated algorithm can be found in [7], but the inferences do not depend on precise locations of the stationary points.

Algorithm 1 Algorithm to find CP-notes from the pitch curve.

```
 1: Track the pitch of each piece according to [18], which
    also identifies silence regions. These result in pitch
    values, $f[i], i \in \{0, \dots, L-1\}$.
 2: In the regions of music (i.e. not silence), with the tonic
    of the piece as $f_0$, find the pitch in semitones with re-
    spect to the tonic as $n[i] = 12\log_2(\frac{f[i]}{f_0})$
 3: $i \leftarrow$ Index of the first pitch sample in the first non-
    silence region
 4: while $i < L$ do
 5:   flagCpNoteFound $\leftarrow$ FALSE
 6:   while $n[j], j \in \{i + C_{\min}, \dots k\}, j < L$ is a CP-
      note and the slope is below the threshold do
 7:     $k \leftarrow k + 1$
 8:     flagCpNoteFound $\leftarrow$ TRUE
 9:   end while
10:   if flagCpNoteFound = TRUE then
11:     Mark the region from $i$ to $k - 1$ as a CP-note
12:   end if
13:   $i \leftarrow k$
14:   if $i$ is in a silence region then
15:     $i \leftarrow$ the index of the first pitch sample in the next
        non-silence region.
16:   end if
17: end while
```

3.3 Key Observations

One more step in the method remains to be described, but it needs to be motivated by a very significant observation by looking ahead at the results. Of all the $\approx 975,000$ transients that were found using Algorithm 2, only 1.25% were of the type SIL-STA-STA-STA-SIL, i.e. the central stationary point was flanked on both sides by two stationary points that ended in silence segments. Further, when one of the renditions was examined closely by ear (by a semi-professional musician), *all* such instances were false

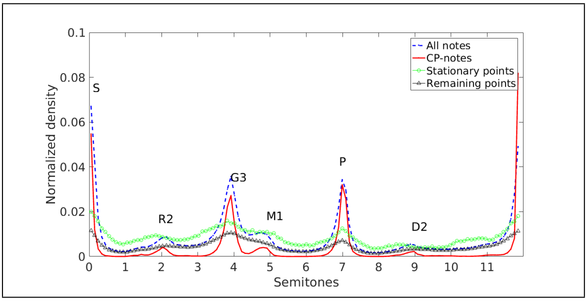


Figure 3: Histograms of all notes, CP-notes, stationary points and the remaining points in transients. *Svara* names are marked for significant peaks of the CP-notes' distribution. These peaks are much sharper than others. This is for the *rāga śankarābharaṇam*, whose nominal scale is S, R2, G3, M1, P, D2, N3, but N3 is not a significant peak.

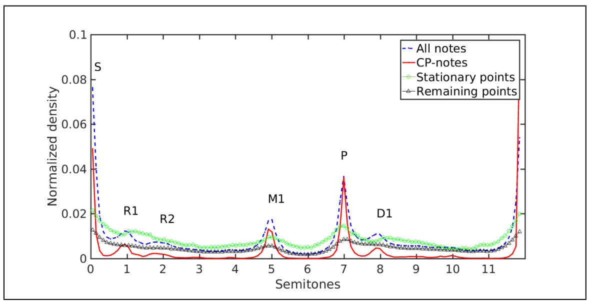


Figure 4: As in Figure 3, but this is for the *rāga tōḍī*. The nominal scale is S, R1, G2, M1, P, D1, N2, but G2 and N2 are not significant peaks, while R2 (not in the scale) is one.

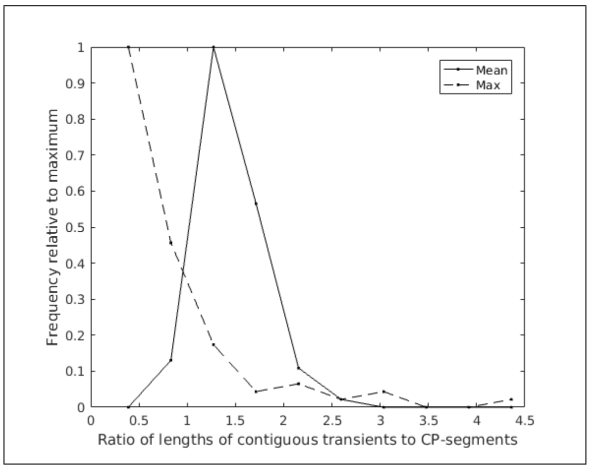


Figure 5: Ratios of maximum and mean durations of CCTs to that of CP-notes

alarms due to errors in marking silence-regions [3]. Thus, we may posit that:

EOb2 *Transients do not occur in isolation.* They occur along with a CP-note or 'chained' to other transients.

Further, from the intuitive experience of producing individual *svara*s *especially at very slow speeds*, we impose that individual *svara*s, when sung in isolation, need to have at least one CP-note segment. If this were not true, we should have had more instances of isolated transients. We use the term 'anchor note(s)' to denote the CP-note(s) associated with transient(s) in *svara*s.

Algorithm 2 Algorithm to find stationary points.

```
 1: for Each segment of music (i.e. not silence) between CP-notes and silence (see Algorithm 1) do
 2:     Smooth the pitch contour with a moving-average filter of length $L = \frac{C_{\min}}{4}$.
 3:     Find the peaks and troughs of the pitch curve in the segment.
 4:     for Each peak (trough) do
 5:         Retain only the maximum peak (minimum trough) in a window of length $L$.
 6:     end for
 7:     for Each remaining peak OR trough, $s$, with pitch value $n_s$ do
 8:         if A nearest neigbhour of $s$ is a CP-note with mean pitch value $n_c$ and $|n_c - n_s| < \Delta$ then
 9:             Discard stationary point $s$.
10:         else if The preceding neighbour is a stationary point $s'$, with pitch value $n_{s'}$ and $|n_s - n_{s'}| < \Delta$ then
11:             Discard stationary point $s'$.
12:         end if
13:     end for
14:     Discard stationary points with pitch values between those of adjacent stationary points/CP-notes.
15: end for
```

If transients do not occur in isolation, it is instructive to see how long a 'contiguous chain of transients (CCTs)' can last without any CP-note appearing. A CCT is defined as a segment of music from the end of a silence segment/CP-note to the start of another. An example corresponds to the pitch curve in Figure 2 from $t \approx 0.65$ sec till $t \approx 1.3$ sec. The histograms of the ratios of the maximum and mean lengths of CCTs and CP-notes in the 84 pieces are shown in Figure 5. It reveals that *the maximum length of a contiguous chain of transients is comparable to the maximum duration of a CP-note* in that piece. Although there are cases where this ratio is > 2, in around 80% of the cases, it is < 1.5. Considering the mean durations: *Over* 50% *of the CCTs do not last longer than* 1.5 *times the mean CP-note duration*. Note that in this 'consistency check', the

[3] A similar percentage of CP-notes or transients flanked by SIL may thus be erroneous, but this does not affect the inferences made later.

definitions of the types of transients (Section 3.1.2) do not matter, so it is an independent corroboration of the utility of EOb2.

However, it does suggest the need to allow phrases to be concatenated by merging *svara*s. This type of chaining of phrases is familiar to musicians [15]. We further specify in detail:

EOb3 When two *svara*s are sung in sequence, their anchor notes of the same pitch can be merged.

Thus, while individual *svara*s need at least one anchor note each, S concatenated *svara*s can have $N(\leq S)$ CP-note segments in them. For example, the two notes – Pa and Ma – marked in Figure 2 share a common anchor note.

3.4 Statistical Analysis

Based on the key observations in Section 3.3, specific statistics relating to transients were collected. First, it is possible for merged anchor notes to reduce in duration and become stationary points in the limit. Thus, the starting point of a transient is counted as the nearest among {the end of a previous CP-note segment; another, earlier stationary point; or the end of a preceding silence segment}. Similarly, the end of the transient is nearest among the starting points of {a following CP-note segment; another, later stationary point; or the beginning of a succeeding silence segment}. These are marked in Figure 2. The distributions of the transient- and CP-note-durations across all *rāga*s are shown in Figure 6. The following observations are made.

1. The maximum duration of the transients is under 1 second, while the maximum duration of CP-notes is over 10 seconds. This result depends upon the definitions of starting points and ending points given above and Figure 5 vindicates the definitions.
2. The long transients are quite rare; more than 90% of them are shorter than 400 ms.
3. There is a variation in the ranges of the transient-durations across *rāga*s, but the mean values of the transient durations are remarkably similar ($\approx$ 100 ms) and is quite close to the 80 ms value set for the parameter $C_{\min}$. See Figures 7 and 8.
4. About 40% of the transients have a very small duration and need to be investigated further, but a major contributing factor are the 'attacks' of notes and where syllables are pronounced. During attacks, the spectrum changes and the pitch curve stops conforming to the definition of a CP-note (Section 3.1.1). Thus, *not all transients are perceived as gamakas.* Some may be perceived as *svara*-attacks.
5. The distribution of different types of stationary points defined in Section 3.1.2 is given in Table 2. This reaffirms the observations in Figures 3 and 4, where stationary points and other non-CP non-stationary points densities show wider peaks. The density for stationary points has been deliberately raised so that it is visible in the figures.

4. MODEL VERIFICATION

4.1 Relation to the Two-Component Model

The narrow peaks of the histograms of CP-notes (Figures 3 and 4) and the durations of transients being comparable to that of CP-notes (Figure 5) suggest that the CP-notes and transients identified by our method are meaningful. Thus, they can be compared with the model presented in [21], where the authors view Carnatic music as consisting of two components called the 'stage' and 'dance'. Our model was derived independently and it is quite significant that many aspects are similar, but there are important practical differences. First, 'stage' in [21] approximately maps to anchor notes, but we restrict anchor notes to be governed by EOb1. This enables automatic segmentation of the audio into CP-notes, transients and silence. Second, the stable and sustained focal pitches of the 'dance' may get classified as a CP-note in our model if they satisfy EOb1. The transient focal pitches would get classified as our transients. In general, focal pitches that do not satisfy EOb1 will be counted as transients. However, similar to [21], it is possible to view the CP-notes (approximately stage) as context of the music and the transients (at least the dance component) as its detail. Similarly, the oscillatory continuity condition of [20] can be seen as a special case of EOb3.

Note that while [21] was based on a largely manual analysis of one rendition of one *varṇam* in one *rāga* (in aid of synthesis), we have found approximate stage-like and dance-like components of 84 pieces in seven major *rāga*s using an automated method (Sections 3.2 and 3.3).

4.2 Listening Tests

We report the results of experiments designed to evaluate the dependence of transients and CP-notes [4]. Approximately 30-second snippets of violin *ālāpana*s (non-*tāḷa*-bound extempore improvisations in a *rāga*) were chosen for the test. Songs were excluded because listeners could have recognized the song's tune rather than the *rāga*. Each snippet was then split into a CP-notes-only and transients-only parts according to Algorithms 1 and 2. *Rāga*s *tōḍī*, *bhairavī*, *śankarābharaṇam*, *varāḷī*, and *kalyāṇī* were chosen and in each, two snippets were picked manually. The listener was asked to identify the *rāga*s of these 30 pieces, while a superset of these *rāga*s was given as possible choices. Listeners could also choose 'Cannot make out' if they actually could not or if they felt the *rāga* played was not in the list. The order of pieces was randomized separately for the transients-only (this set of 10 was played first), CP-notes only (played next) and the unedited snippets (played last). Fifty participants took the test.

The overall results point to the CP-notes' and transients' dependencies on each other: 81.4% of the clips were identified correctly, but only 58.4% of the transients-only snippets, and 68.6% of the CP-notes only. *Rāga*-wise results are presented in Table 3, which are restricted to the expert participants – those who could identify the *rāga*s of

[4] `http://www.iitm.ac.in/donlab/wermusic/index.html?owner=venkat&testid=test1&wer=30`

Type	CPN ∘ CPN	CPN∘STA	CPN∘SIL	STA∘STA	STA∘SIL
Percentage	14.5	17.5	5.3	49.0	13.7

Table 2: Relative occurrence of stationary-point types.

Rāga	Percentage of participants	
	Transients only	CP-notes only
Tōḍī	95.0	81.3
Bhairavī	43.2	81.1
Śankarābharaṇam	21.6	85.1
Varāḷī	86.5	89.2
Kalyāṇī	89.7	68.0

Table 3: Accuracy of identification by experts

all unedited snippets of that *rāga*. Only one of them could identify all instances of snippets with transients only and CP-notes only. Among the *rāga*s, *tōḍī* and *varāḷī* fared best. *Bhairavī* was surprisingly not easily identifiable with only transients, even though it is considered a *gamaka*-heavy *rāga*. *Kalyāṇī* was easier to recognize by transients than only CP-notes, but *śankarābharaṇam* could be identified from transients-only by only about 20% of the experts. Overall, the results suggest that CP-notes and transients need each other in Carnatic music.

Some qualitative results are also significant. A few participants expressed surprise when presented with only CP-notes or transients. Several said that the edited pieces were quite unpleasant to the ear (edited vocal pieces sounded worse). The edited pieces were noticeably fragmented, but the artefacts were identical in both types of clips. Yet. interestingly, the transients-only clips – a crucial component of *gamaka*s – were perceived as more unpleasant. This clearly suggests the importance of context for transients in Carnatic music, which is provided, at least partially, by the CP-notes.

5. CONCLUSION

The qualitative stage and dance model model for Carnatic music is corroborated with an equivalent CP-note, transient model. An analysis of CP-notes and transients shows that the CP-notes can last much longer than transients in duration. We proposed that *svara*s can be viewed as CP-notes providing the context for any transients. When combined with the chaining rule, these observations can explain *rāga* motifs.

In a listening test, while 28 experts correctly identified all original, unedited audio clips in five *rāga*s, only one correctly identified all CP-notes-only and transients-only clips. Several listeners reported that the latter pieces were quite unpleasant on the ear.

Thus, the CP-notes and transients model is potentially a good model of Carnatic music, with our analyses suggesting that there is significant contextual *rāga* information in CP-notes; they are, in fact, crucial for a pleasant listening experience of the transients in the profusion of *gamaka*s in Carnatic music.

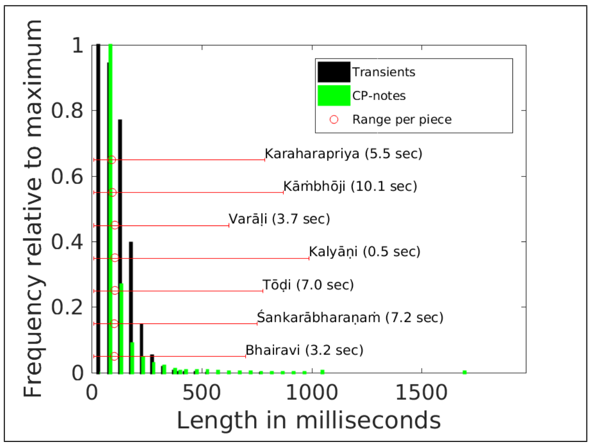


Figure 6: Distribution of the durations of transients and CP-notes across *rāga*s described in Table 1. Each horizontal line shows the *rāga*-specific range of transient-durations, and the circles, their mean durations. Numbers in brackets are the durations of the longest CP-note.

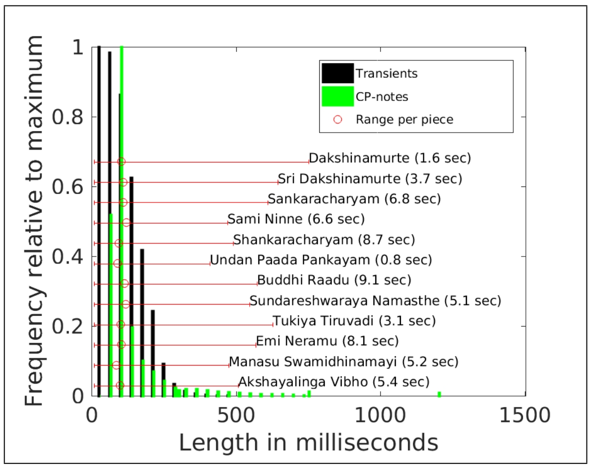


Figure 7: Distribution as in Figure 6 restricted to the *rāga* *śankarābharaṇam*. The range and means of transient-durations correspond to each piece in the *rāga*.

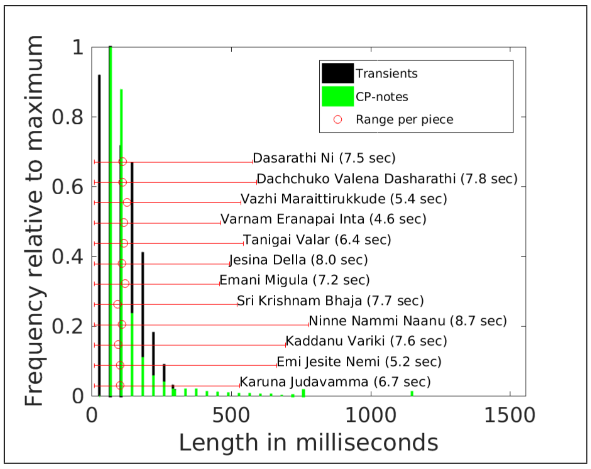


Figure 8: Distribution as in Figure 7 restricted to the *gamaka*-laden *rāga* *tōḍī*. Its transient-duration means are similar to those of *śankarābharaṇam*.

6. ACKNOWLEDGEMENTS

This research was partly funded by the European Research Council under the European Unions Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583). V Viraraghavan thanks Ms. Anju Leela Thomas for her help in setting up the listening test, all participants of the test, and Dr. Sankalp Gulati for providing, and helping with, the CompMusic database.

7. REFERENCES

[1] Carnatic Corpus in Dunya. http://dunya.compmusic.upf.edu/carnatic/?a=10&r=55. Accessed: 2017-04-16.

[2] CompMusic Project. http://compmusic.upf.edu/. Accessed: 2017-04-16.

[3] L. Subramaniam - Ninnu kori (varnam) - 15 speeds. https://www.youtube.com/watch?v=w4oOB8EOe5g. Accessed: 2017-04-16.

[4] Ashwin Bellur, Vignesh Ishwar, and Hema A Murthy. Motivic analysis and its relevance to raga identification in carnatic music. 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. 153-157*. Universitat Pompeu Fabra, 2012.

[5] Subbarama Dikshita. *Sangita Sampradaya Pradarshini*. Web version, http://www.ibiblio.org/guruguha/ssp.htm, 2008.

[6] Shrey Dutta. Analysis of motifs in carnatic music: A computational perspective. Master's thesis, IIT, Madras, Chennai, 2016.

[7] Shrey Dutta, SPV Krishnaraj, and Hema A Murthy. Raga verification in carnatic music using longest common segment set. In *Int. Soc. for Music Information Retrieval Conf.(ISMIR)*, pages 605–611, 2015.

[8] Shrey Dutta and Hema A Murthy. Discovering typical motifs of a raga from one-liners of songs in carnatic music. In *ISMIR*, pages 397–402, 2014.

[9] Shrey Dutta and Hema A Murthy. A modified rough longest common subsequence algorithm for motif spotting in an alapana of carnatic music. In *Communications (NCC), 2014 Twentieth National Conference on*, pages 1–6. IEEE, 2014.

[10] K. K. Ganguli, A. Rastogi, V. Pandit, P. Kantan, and P. Rao. Efficient melodic query based audio search for hindustani vocal compositions. In *Proc. of the International Society for Music Information Retrieval (ISMIR)*, Oct. 2015, pp. 591597, Malaga, Spain.

[11] Kaustuv Kanti Ganguli, Ashwin Lele, Saurabh Pinjani, Preeti Rao, Ajay Srinivasamurthy, and Sankalp Gulati. Melodic shape stylization for robust and efficient motif detection in hindustani vocal music. In *Proc. of the National Conference on Communication (NCC)*, 2017.

[12] Sankalp Gulati, Joan Serrà, Kaustuv Kanti Ganguli, Sertan Sentürk, and Xavier Serra. Time-delayed melody surfaces for rāga recognition. In *ISMIR*, pages 751–757, 2016.

[13] Sankalp Gulati, Joan Serrà, Vignesh Ishwar, and Xavier Serra. Discovering Raga Motifs by Characterizing Communities in Networks of Melodic Patterns. In *41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*, pages 286–290, Shanghai, China, 2016. IEEE.

[14] Vignesh Ishwar, Shrey Dutta, Ashwin Bellur, and Hema A Murthy. Motif spotting in an alapana in carnatic music. In *ISMIR*, pages 499–504, 2013.

[15] T. M. Krishna. *A Southern Music: The Karnatik Story*. Harper Collins India, 1st edition, 2016.

[16] Arvindh Krishnaswamy. Melodic atoms for transcribing carnatic music. In *International Symposium on Music Information Retrieval*, number 228. ISMIR, 2004.

[17] P V Krishna Raj, Venkata Subramanian Viraraghavan, Sridharan Sankaran, and Hema A Murthy. An approach to transcription of varnams in carnatic music using hidden markov models. In *Proc. of the National Conference on Communication (NCC)*, 2017.

[18] Justin Salamon and Emilia Gómez. Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20:1759–1770, 2012.

[19] Justin Salamon, Sankalp Gulati, and Xavier Serra. A multipitch approach to tonic identification in indian classical music. In *Gouyon F, Herrera P, Martins LG, Müller M. ISMIR 2012: Proceedings of the 13th International Society for Music Information Retrieval Conference; 2012 Oct 8-12; Porto, Portugal. Porto: FEUP Ediçoes; 2012.* International Society for Music Information Retrieval (ISMIR), 2012.

[20] Srikumar Subramanian, Lonce Wyse, and Kevin McGee. Modeling speed doubling in carnatic music. In *ICMC*, 2011.

[21] Srikumar K Subramanian, Lonce Wyse, and Kevin McGee. A two-component representation for modeling gamakas of carnatic music. In *Proc. of 2nd CompMusic Workshop*, 2012.

[22] Venkatasubramanian V, K R Ramakrishnan, and H V Sahasrabuddhe. Music information retrieval using continuity. In *Proceedings of the Symposium on the Frontiers of Research on Speech and Music (FRSM 2004)*, number 228, pages 74–83. Annamalai University, Chidambaram, 2004.

ACCURATE AUDIO-TO-SCORE ALIGNMENT FOR EXPRESSIVE VIOLIN RECORDINGS

Jia-Ling Syue[1*] **Li Su**[2*] **Yi-Ju Lin**[1] **Pei-Ching Li**[1]
Yen-Kuang Lu[1] **Yu-Lin Wang**[1] **Alvin W. Y. Su**[1]
[1] SCREAM Lab., Department of CSIE, National Cheng-Kung University, Taiwan
[2] Music and Culture Technology Lab., IIS, Academia Sinica, Taiwan
`P76044457@mail.ncku.edu.tw, lisu@iis.sinica.edu.tw`

ABSTRACT

An audio-to-score alignment system adaptive to various playing styles and techniques, and also with high accuracy for onset/offset annotation is the key step toward advanced research on automatic music expression analysis. Technical barriers include the processing of overlapped notes, repeated note sequences, and silence. Most of these characteristics vary with expressions. In this paper, the audio-to-score alignment problem of expressive violin performance is addressed. We propose a two-stage alignment system composed of the dynamic time warping (DTW) algorithm, simulation of overlapped sustain notes, background noise model, silence detection, and refinement process, to better capture the onset. More importantly, we utilize the non-negative matrix factorization (NMF) method for synthesis of the reference signal in order to deal with highly diverse timbre in real-world performance. A dataset of annotated expressive violin recordings in which each piece is played with various expressive musical terms is used. The optimal choice of basic parameters considered in conventional alignment systems, such as features, distance functions in DTW, synthesis methods for the reference signal, and energy ratios, is analyzed. Different settings on different expressions are compared and discussed. Results show that the proposed methods notably improve the conventional DTW-based alignment method.

1. INTRODUCTION

An audio-to-score alignment algorithm captures note-level information of a music performance with the aid of symbolic data such as MIDI. It has numbers of applications in the field of music information retrieval (MIR), such as automatic accompaniment [4], and music retrieval through matching a MIDI file to a polyphonic audio recording [7]. Besides, an effective audio-to-score alignment algorithm is also critical in computational music analysis, specifically in the case of extracting the note-level information in expressive music recordings for music expression analysis. For example, Li *et al.* [10] used an audio-to-score alignment algorithm [19] to annotate the onset and offset positions of each note in a dataset, including violin solo pieces interpreted by professional violinists with 10 expressive musical terms. [1] However, such annotation still needs to be checked and corrected manually as it suffers from low quality when there are overlapped sustain segments and unexpected silence between successive notes in expressive violin performance. Therefore, an improved audio-to-score alignment algorithm for expressive violin performance would be of great help to avoid such a tedious and labor-intensive process.

There have been numbers of audio-to-score alignment algorithms proposed in the past few decades, based on graphical models [2, 15, 16], hidden Markov models (HMM) [3, 6, 13, 18], and DTW [5, 12, 14]. Among these models, an HMM [13] is of better potential in modeling how the states of attack, decay or sustain evolve in a note, but to train the model parameters one needs amounts of correctly annotated data which are, as mentioned, hard to get without an improved audio-to-score alignment algorithm. Therefore, as an attempt to with low-resource data, we opt to use DTW with further processing steps, which can be implemented without the need of a large dataset.

Another challenge of matching a score to a musical recording is the diversity of timbre of the input signals, depending on the performer, instrument, recording environment, etc. Mismatch of spectral features between the MIDI-synthesized reference and the real violin sounds of the same event leads to errors in alignment. Such an issue was addressed by an HMM-based model exploiting the spectral templates adaptive to different recordings [9]. In this paper, we adopt a more straightforward approach,

[1] The SCREAM-MAC-EMT dataset compiled by Li *et al.* contains recordings of 10 different classical music pieces, each of which is interpreted with 5 selected expressions by 11 musicians [10]. It considers in total 10 expressive musical terms, including *Scherzando* (playful), *Tranquillo* (calm), *Con Brio* (bright), *Maestoso* (majestic), *Risoluto* (rigid), *Affettuoso* (affectionate), *Agitato* (agitated), *Cantabile* (like singing), *Grazioso* (graceful), and *Espressivo* (expressive). The experiments in this paper are performed on a subset of this dataset, which contains 50 recordings from randomly selected 3 musicians' performance.

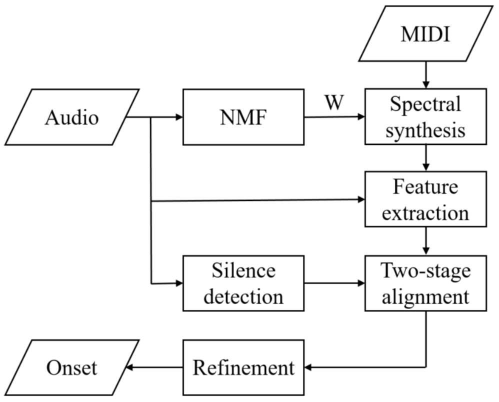


Figure 1. Flowchart of the proposed audio-to-score alignment system for expressive violin performance.

which utilizes the NMF method to directly learn the spectral template for synthesis under low-resource data.

By inspecting the data, we raise four issues which could be seen in aligning an expressive violin recording with its corresponding MIDI. First, the strong sustain of one note could be overlapped with the weak onset of its next note, and this makes the algorithm fail to accurately capture the onset time of the next note. Second, in a repeated note sequence, an algorithm is prone to erroneous onset detection since all notes have the same pitch. Third, the *staccato* technique usually causes unexpected silent segments which are not compatible to the ground-truth MIDI. Lastly, the background noise in a real-world environment may also cause mismatching. These issues are commonly seen; for instance, an inspection shows that in the dataset, 37% of the notes have overlapped sustain with their successive notes, 35% are in a repeated note sequence, 6% have unexpected silence, and 2% follow a rest symbol.

To this end, we add solutions to deal with the four problems: First, we simulate the note overlap in two-stage alignment (cf. Section 2.3). Second, we perform the duration ratio of repeated notes (cf. Section 2.4) and silence detection (cf. Section 2.2) to refine onset positions. Besides, we add a background noise template to model the rest parts in a recording (cf. Section 2.1). These processes are evaluated after a systematic experiment which finds the optimal parameters such as features, timbres for synthesis, distance measures, and energy measures in an audio-to-score algorithm (cf. Section 3.1). Moreover, we also discuss the precision of proposed alignment system within different levels of error tolerance, and draw an insight from analyzing the expression-wise performance (cf. Section 3.2).

2. THIS WORK

Figure 1 shows the diagram of the proposed alignment system, whose goal is to find the onset position accurately in expressive violin performance. The system takes an audio signal and its corresponding MIDI file as input. We adopt the NMF to learn the spectral patterns of the audio input of violin solo for MIDI-to-audio synthesis. Then, either chroma or log-frequency spectral features, which are the basic parameters considered in conventional alignment systems (cf. Section 3.1), are extracted from both the audios. Incorporated with a silence detection process, a two-stage DTW-based audio-to-score alignment and a refinement process are conducted for resolving the discrepancies between audio and MIDI in expressive recordings including overlapped sustain segments, unexpected silence segments, and repeated note sequences.

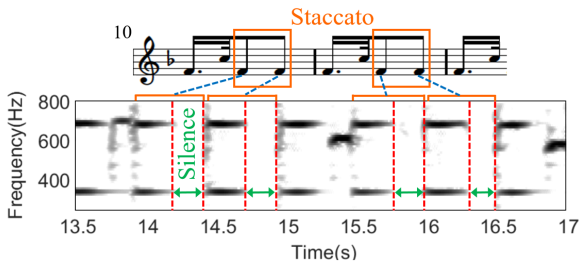


Figure 2. The audio played with *staccato* has extra silence segments (green) that could not be found from the score.

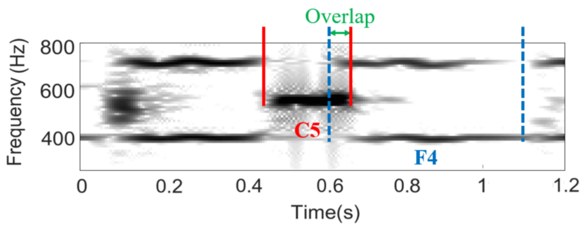


Figure 3. An example of a note with strong energy in sustain segment overlaps with its successive note, which has weak energy in attack segment.

2.1 NMF and Spectral Synthesis

For MIDI-to-audio conversion, the NMF with the Kullback-Leibler divergence [17] is adopted to train the spectral patterns of each pitch in a recording. The NMF decomposes an audio spectrogram V into two matrices W and H, i.e. $V \approx \hat{V} = WH$, where W is a spectral template represented in column, and H is a time-varying energy activation represented in row. Following Joder *et al.* [8], we adopt a Gaussian mixture model to initialize the template matrix W for each pitch with k Gaussian functions centered at the fundamental frequency and the first k-th harmonics of the pitch. Due to the weak energy in the high-frequency range, we take k to be 4, the weight of each Gaussian function to be k^{-2}, and the variance to be 30 cents. Besides, we consider the frequency range from 65 Hz to 4 kHz, which removes the high- and low-frequency noise. The activation matrix H is initialized by a normal distribution with zero mean and unit standard deviation. Moreover, we add an additional *noise template* (NT) with random numbers in [0, 1] to simulate the noise of silence parts in the recorded audio signal. Furthermore, we adopt a preprocessing step of stretching or shortening the reference signal by insertion or deletion of frames so as to make

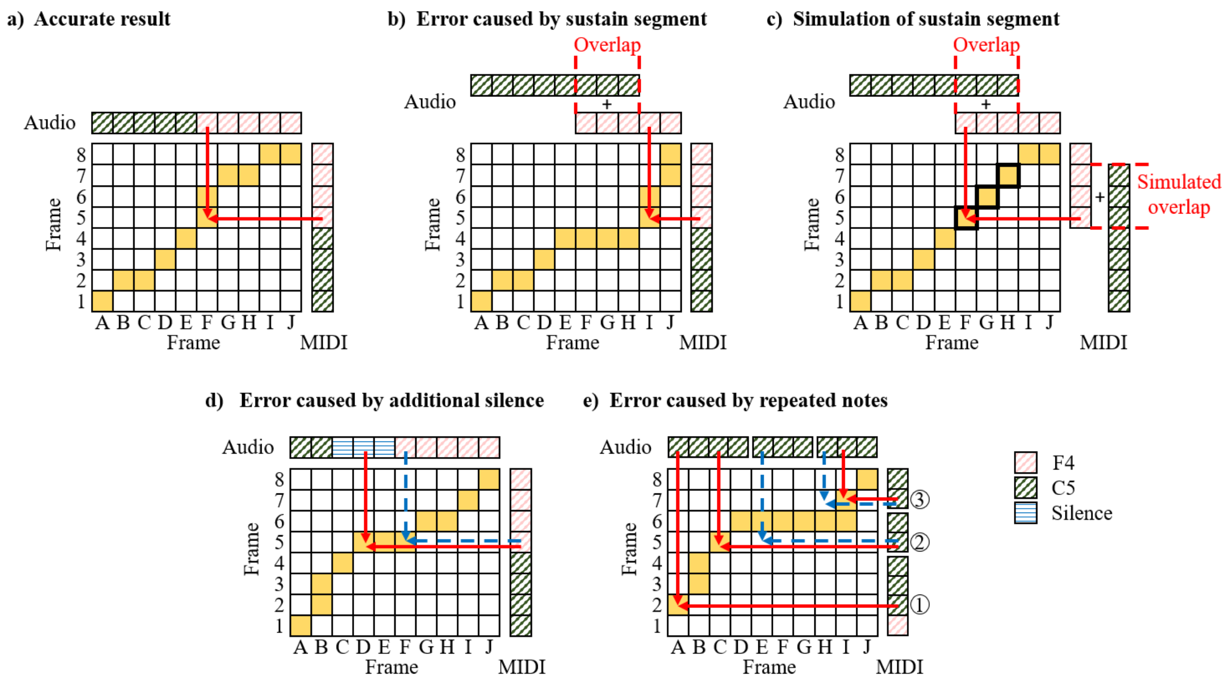

a) Accurate result
b) Error caused by sustain segment
c) Simulation of sustain segment
Overlap
Audio
Frame
MIDI
A B C D E F G H I J
1 2 3 4 5 6 7 8
Simulated overlap
d) Error caused by additional silence
e) Error caused by repeated notes
①
②
③
F4
C5
Silence

Figure 4. Examples of accurate and erroneous alignment paths: (a) accurate result, (b) error caused by overlapped sustain segment, (c) simulation of overlapped sustain segment, and error caused by (d) additional silence and (e) repeated notes.

its length similar to the input, in order to reduce the effect of tempo changes in expressive violin performance.

2.2 Silence Detection

Violinists use different playing techniques to interpret distinct expressions. The *staccato* technique might be the one which is most likely to cause errors in DTW alignment among others due to the silence segments caused by articulation of successive notes. Such a silent segment do not have any information in the reference signal but could be found in the audio recording as shown in Figure 2, it results in deterioration in the DTW alignment path. This issue is addressed by introducing an extra silence detection process with energy measurement. From the fast Fourier transform (FFT) [2], the energy curve is computed by summing the spectrum over all the frequency bins and is expressed in dB scale. A silence segment is one which is longer than 100ms and whose energy is less than 12 dB.

2.3 Two-stage Alignment

The main purpose of the proposed two-stage audio-to-score alignment process is to capture the accurate onset for overlapped successive notes where the former note has a *long sustain* and the latter one has a *soft onset*. Such a specific energy characteristic of violin is likely to cause wrong alignment paths. As illustrated in Figure 3, the first note `C5`, which has strong energy in sustain segment, overlaps with the second note `F4`, which has weak energy in the attack, and leads to a distorted alignment path. Figure 4(a) shows the ideal accurate result of this example: the onset of `F4` is at position $(\mathrm{F}, 5)$, while Figure 4(b) shows the actual erroneous result of `F4`, where the onset of `F4` locates at $(\mathrm{I}, 5)$. This is because the first 3 frames of `F4` are submerged in the sustain segment of `C5`.

The two-stage alignment process is proposed to solve this problem. In the first stage, we adopt the conventional DTW-based alignment as our *baseline*, and obtain a rough estimation of the onset of each note. Next, we add the information of the silence segments mentioned in Section 2.2. If there is no silence detected between two successive notes, then the two notes are considered overlapped. For every pair of the overlapped notes, we lengthen the first note of the reference MIDI with a duration of 3 frames, in order to simulate the behavior of overlapped notes. [3] Then, we perform DTW again to align the audio with the modified MIDI. This is the *second-stage alignment* (SA) process. The SA process of the overlapped notes `C5`-`F4` is shown in Figure 4(c). The two-stage alignment is therefore a combination of the baseline process and the SA process.

2.4 Refinement

The refinement process contains two tasks. The first task is to fix the case of *staccato* and *rest*, where the errors of alignment are usually caused by *silence* rather than overlapping between two consecutive notes. Figure 4(d) shows the alignment result of an audio played with *staccato*. The ideal accurate onset of `F4` is at position $(\mathrm{F}, 5)$ as pointed by the dashed-line arrow; however, the actual result of `F4`

[2] This paper uses a 2,048-point Hamming-windowed FFT with each frame staggered by 882 samples (20ms) throughout all the experiments.

[3] We observe that for most of the overlapped notes, the overlapping durations are between 2 and 4 frames (40ms and 80ms). Such an observation gives to an estimation 3 frames.

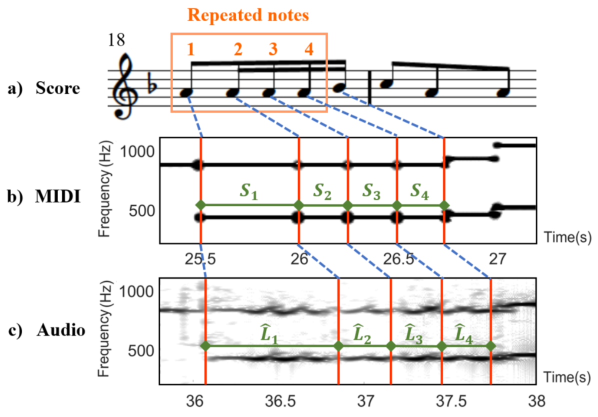


Figure 5. An example of repeated notes, whose spectrum is highly similar to each other. Their onsets are calibrated via the duration ratio estimated from the reference signal.

locates at $(\mathrm{D}, 5)$, a position within the silence segment, as pointed by the solid-line arrow. To address this issue, the *refinement of silence* (RS) is implemented as follows: if the onset of an aligned note locates in a silence segment of the audio, then it will be shifted to the correct position, that is, the frame right after such a silence segment.

The second task is to adjust the duration of repeated notes, where the alignment path is hard to estimate because of high similarity among the note spectra. This issue is illustrated in Figure 4(e), where one can see that except for the first C5, the onset results of the other notes are at the wrong positions, $(\mathrm{C}, 5)$ and $(\mathrm{I}, 7)$, respectively, as pointed by the solid-line arrows. The correct onsets of the second and the third C5 are the positions pointed by the dashed-line arrows. Figure 5 shows a real-world example of repeated notes in a more detailed manner: the spectra of the four repeated notes A4 are highly similar, especially for the last three notes with the same note type. It is hard to figure out the accurate onset of each repeated note when performing DTW alignment.

We therefore come up with a strategy to deal with this problem. Our assumption is that both audio and its corresponding MIDI file have approximately similar duration ratio of repeated notes. In other words, we can modify the onsets of repeated notes by referring to the duration ratio of such notes in the reference signal. Given a sequence of repeated notes, $r = (r_1, r_2, ..., r_m)$, the *refinement of repeated notes* (RRN) is realized as follows:

STEP 1 Computing the duration of each repeated note, $S = (S_1, S_2, ..., S_m)$ and $L = (L_1, L_2, ..., L_m)$, according to the reference signal and the onset results of the two-stage alignment, respectively.

STEP 2 Calculating the duration ratio for each repeated note, i.e. $\text{ratio}_k = S_k / \sum_{k=1}^{m} S_k$.

STEP 3 Estimating the predicted duration for such notes in the audio recording, $\hat{L} = (\hat{L}_1, \hat{L}_2, ..., \hat{L}_m)$, via the calculation of $\hat{L}_k = L_k \times \text{ratio}_k$.

Timbre	MIDI synthesizer		NMF
Feature	12-D chroma	84-D linear-log spectrum	
Precision	72.68	81.37	95.12

Table 1. Comparison of precision (in %) using chroma and the 84-D spectrum feature. Since the 84-D spectrum performs better, the timbre synthesis via MIDI synthesizer or NMF is considered.

STEP 4 Adjusting the duration of the first $m-1$ repeated notes according to the criterion of $|\hat{L}_k - L_k| > \theta$. If $\hat{L}_k > L_k$ then the onset of the $(k+1)$-th note is shifted backward by $|\hat{L}_k - L_k| - \theta$ ms. Besides, L_{k+1} is also updated by the shifted value. On the contrary, it is updated by shifting forward $|\hat{L}_k - L_k| - \theta$ ms, and so does the L_{k+1}.

Our pilot study shows that choosing $\theta = 60$ ms gives better performance.

3. EVALUATION

The experiments are separated into two parts. The first part is to find the optimal setting used in the conventional DTW-based alignment. The second part is the results of the proposed system, including the performance of baseline process, proposed processes, and expression-wise. The test dataset contains 10 expressions, each with 5 classical music pieces, totaling 50 excerpts (2,925 notes). We use precision as our evaluation method, which is the percentage of the number of correct onsets among all the excerpts. A correct onset is defined as the difference between aligned onset time and its corresponding ground-truth onset time, being less than 100ms.

3.1 Factors Experiment

We consider four types of factors: feature, timbre, distance function, and energy. The brief introduction and results of each factor are described one by one as follows.

Two features are considered: chroma [1] and linear-log frequency spectrum [5]. The chroma is a 12-dimensional vector representing the energy of the 12 pitch classes (i.e. C, C#, ..., B). The linear-log frequency spectrum is a spectral feature with reduced dimension, performed by a 84-D filterbank, which is in linear scale in the low frequencies and in logarithm scale in the high frequencies [5]. Such a feature simulates the linear-log frequency sensitivity in human auditory systems. Table 1 compares the two features according to the averaged precision (in %) of all the 50 excerpts. The results indicate that the linear-log spectrum is better than the chroma. We therefore select the 84-D spectrum for the feature factor.

Then, we compare two methods of synthesizing the reference signal from MIDI, where the one is directly through a MIDI synthesizer, and the other uses the NMF to learn the spectral features from the original audio recording, a similar strategy of the HMM-based timbre learning method

E_a/E_m	13 dB	10 dB	0 dB	-10 dB	-13 dB
Cosine	96.10	96.10	**96.14**	96.00	96.00
Euclidean	77.81	88.62	95.12	78.39	60.62
SKL	95.73	95.86	95.41	96.85	96.75

Table 2. Comparison of precision (in %) using three types of distance functions for DTW with five different levels of energy ratios of audio recording to reference signal.

Process	Precision
Baseline	96.14
Baseline+NT	97.03
Baseline+NT+SA	97.64
Baseline+NT+SA+RS	97.88
Baseline+NT+SA+RS+RRN ('Proposed')	**98.43**

Table 3. Performance (in %) of the baseline and the proposed system. NT: noise template; SA: second stage alignment; RS: refinement of silence; RRN: refinement of repeated notes.

Process		NT	SA	RS	RRN	Other
# Notes		45	1094	173	1015	598
# Errors	Baseline	15	42	6	40	10
	Proposed	2	26	1	10	7

Table 4. Comparison of the number of error notes between the baseline and the proposed system based on the four raised issues.

by Joder *et al.* [9]. To simulate the silence parts, we simply apply zero values for silence segments, which is the same means used in the MIDI synthesizer. Results in Table 1 also shows that using NMF for timbre synthesis yields much better precision (95.12%) than using a MIDI synthesizer (81.37%), since a common MIDI synthesizer can not well resemble the wide variety of timbre in expressive violin performance. We therefore take the NMF-based synthesis method for the following experiments.

Moreover, since the dynamics of notes vary largely in expressive violin performance, the distance functions in DTW and the frame-level energy are also essential factors in the alignment process. We compare three types of distance functions in the DTW algorithm: cosine similarity [11], Euclidean distance [5], and symmetric Kullback-Leibler (SKL) divergence [9]. Cosine similarity is the normalized inner product of two non-zero vectors. Since we would like to find the minimal value of the cost function, the inner product is subtracted by 1. Besides, Euclidean distance calculates the straight-line distance of two feature vectors. Further, SKL divergence is defined as: $d_{SKL}(i,j) = d_{KL}(i \parallel j) + d_{KL}(j \parallel i)$, where i and j are two n-dimensional vectors. The performance of a distance function is highly related to the effect of *energy ratios*, i.e. the ratio of the energy levels of the audio recording (E_a) to the reference signal (E_m). Table 2 presents the averaged precision values using the three distance functions for DTW with five different levels of energy ratio from -13 dB to 13 dB. All the three distance functions have similar performance when audio and reference signals have similar levels of energy. However, when the energy ratio exceeds 10 dB, performance degrades significantly for Euclidean distance, while the cosine similarity turns out to be the most stable distance function among all levels of energy ratio (STD=0.06%). Therefore, we opt to use cosine similarity in the following experiments.

In short, the optimal setting of the conventional DTW-based alignment algorithm (i.e. the *baseline*) encompasses linear-log spectral features, the reference signal synthesized with NMF on the input signal, and cosine similarity as a distance function. We will use these settings in the following experiments if not mentioned.

3.2 System Experiment

3.2.1 Overview

Table 3 lists the performance of the proposed system and a comparison to the individual building blocks mentioned in Section 2, i.e. one-stage DTW only (Baseline), noise template (NT), second-stage alignment (SA), refinement of silence (RS), and refinement of repeated notes (RRN). Results show that the baseline achieves a precision of 96.14%. Its performance is then increased by 0.89% after adding NT. A further improvement of 0.61% is seen after adding SA and addressing the issue of overlapped sustain notes. Finally, the RS and RRN also give a slight improvement of 0.24% and 0.55% subsequently. As a result, the averaged precision of the proposed system comes to 98.43%, showing a significant improvement from the baseline system as validated by a two-tailed t-test ($p <$ 0.05, d.f.=98).

Table 4 gives a more in-depth comparison of the number of error notes between the baseline and the proposed system according to the four raised issues.[4] We find that every process reduces the number of error notes of their corresponding types, and the RRN process leads to the greatest improvement: 40 error notes within repeated note sequences are reduced to 10 notes.

Table 5 shows the precision of the proposed system within different levels of error tolerance values not only at 100ms but ranged from 20ms (1 frame) to 700ms (35 frames). We find that the performance is over 90% when using the tolerance with 60ms (3 frames). In addition, the maximal erroneous time of onset is within 700ms.

3.2.2 Expression-wise Performance

Table 6 presents the averaged precision for the 10 violin expressions based on the Baseline process with two distinct timbre synthesis methods, MIDI synthesizer and NMF, and the proposed system, respectively. Comparing the MIDI

[4] An NT refers to a note which follows a rest symbol; SA counts the note that overlaps its successive notes over 60ms; RS includes the notes played with *staccato*; RRN contains the notes belonging to a sequence of repeated notes; the remaining ones are marked as'Other'.

Error $\leq$		Proposed
Frames	Seconds	
1	0.02	58.60
2	0.04	85.61
3	0.06	94.84
5	0.1	98.43
10	0.2	99.62
15	0.3	99.79
20	0.4	99.83
25	0.5	99.93
30	0.6	99.97
35	0.7	100.00

Table 5. Performance (in %) of the proposed system within different levels of error tolerance values.

Expression	Baseline		Proposed
	MIDI	NMF	
Scherzando	78.25	96.37	96.98
Tranquillo	69.65	93.06	98.55
Con Brio	87.19	98.44	98.75
Maestoso	82.73	97.48	98.56
Risoluto	76.86	96.92	99.49
Affettuoso	87.41	96.67	100.00
Agitato	88.78	97.96	96.94
Cantabile	86.43	93.97	95.98
Grazioso	86.44	95.25	99.66
Espressivo	78.07	95.35	98.00

Table 6. Performance (in %) of the ten violin expressions via the baseline process with two distinct timbre synthesis methods and the proposed system.

synthesizer (i.e. the second column) to NMF-based synthesis from audio recording (i.e. the third column), a significant improvement can be observed ($p < 0.005$, d.f.=18), especially for *Tranquillo* and *Risoluto*, where the improvement is over 20% for both cases. Besides, the proposed system (i.e. the fourth column) has significant improvement from both the Baseline cases ($p < 0.05$, d.f.=18). Particularly, *Tranquillo*, *Grazioso*, and *Affettuoso* are improved the most; this implies that the proposed system can enhance the onset precision for such violin performance with plentiful expression and intense vibrato. For *Risoluto*, the expression played with *staccato* technique mostly, the proposed system also gives excellent result. Furthermore, we see that the improvement of *Con Brio* and *Scherzando* is limited, probably due to their intense characteristics of performance such as clear attack of energy envelope.

3.3 Discussion

According to the expression-wise performance as illustrated in Table 6, we find that *Agitato* is the only one expression which has degraded precision via the proposed system. The reason is possibly that the energy of sustain segment might be weak such that the simulation of sustain perhaps cause additional errors. Except for *Agitato*, the proposed system has improvement for other expressions.

Although we use a refinement process to deal with the unexpected silence segments caused by the *staccato* technique, this process actually could be merged into the two-stage alignment. For example, we can adopt similar means which is used in the simulation of overlapped sustain notes, by inserting additional frames in a reference signal based on the information of silence segments. Thereby, the system will be made more succinct.

In this paper, we only consider a subset of the violin expression dataset, which includes 50 solo recordings from randomly selected 3 musicians' performance. In order to obtain more reliable performance and to develop a robust alignment system, the test data needs to be expanded such as using the recordings from other musicians in the SCREAM-MAC-EMT dataset as well as data of polyphonic recordings, where the latter suggest a future work of constructing a new dataset for expression analysis of violin solo in polyphonic music.

Moreover, this study only considers the accuracy of onset-only alignment. Another important task for music expression analysis of notes is offset alignment, which is still a challenging problem. An extension of the proposed alignment system such as to cover the offset alignment issue is also left as future work.

4. CONCLUSION

To have better expression analysis of violin recordings, it is desired to have the precise onset information of each note. The conventional DTW algorithm is modified for accurate audio-to-score alignment for the violin dataset, including the simulation of sustain notes, silence detection, refinement of duration ratio of repeated notes, and background noise model, which are used to deal with the four common issues usually seen in violin recordings. Experiments show that high precision is achieved if instrumental timbre and the 84-dimensional spectral feature vector are used. Cosine similarity is adopted as our distance formula for its robustness to various violin playing techniques. The proposed two-stage alignment system obtains significant improvement, not only for the addressed issues but also for the distinct expressions, from the baseline process.

5. ACKNOWLEDGEMENTS

The authors would like to thank the Ministry of Science and Technology of Taiwan for its financial support of this work, under contract MOST 105-2221-E-006-148.

6. REFERENCES

[1] M. A. Bartsch and G. H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on multimedia*, pages 96–104, 2005.

[2] B. Catteau, J. P. Martens, and M. Leman. A probabilistic framework for audio-based tonal key and chord

recognition. *Advances in Data Analysis*, pages 637–644, 2007.

[3] A. Cont. Realtime audio to score alignment for polyphonic music instruments, using sparse non-negative constraints and hierarchical HMMs. In *ICASSP*, pages 245–248, 2006.

[4] R. B. Dannenberg and C. Raphael. Music score alignment and computer accompaniment. *Communications of the ACM*, pages 38–43, 2006.

[5] S. Dixon. Live tracking of musical performances using on-line time warping. In *DAFx*, pages 92–97, 2005.

[6] Z. Duan and B. Pardo. A state space model for online polyphonic audio-score alignment. In *ICASSP*, pages 197–200, 2011.

[7] N. Hu, R. B. Dannenberg, and G. Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *WASPAA*, pages 185–188, 2003.

[8] C. Joder, S. Essid, and G. Richard. Optimizing the mapping from a symbolic to an audio representation for music-to-score alignment. In *WASPAA*, pages 121–124, 2011.

[9] C. Joder and B. Schuller. Off-line refinement of audio-to-score alignment by observation template adaptation. In *ICASSP*, pages 206–210, 2013.

[10] P.-C. Li, L. Su, Y.-H. Yang, and A. W. Y. Su. Analysis of expressive musical terms in violin using score-informed and expression-based audio features. In *ISMIR*, pages 809–815, 2015.

[11] R. Macrae and S. Dixon. Accurate real-time windowed time warping. In *ISMIR*, pages 423–428, 2010.

[12] B. Niedermayer and G. Widmer. A multi-pass algorithm for accurate audio-to-score alignment. In *ISMIR*, pages 417–422, 2010.

[13] N. Orio and F. Déchelle. Score following using spectral analysis and hidden markov models. In *ICMC*, pages 151–154, 2001.

[14] C. Raffel and D. P. W. Ellis. Optimizing DTW-based audio-to-midi alignment and matching. In *ICASSP*, pages 81–85, 2016.

[15] C. Raphael. A hybrid graphical model for aligning polyphonic audio with musical scores. In *ISMIR*, pages 387–394, 2004.

[16] C. Raphael. Aligning music audio with symbolic scores using a hybrid graphical model. *Machine learning*, pages 389–409, 2006.

[17] T. Virtanen. Monaural sound source separation by non-negative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, pages 1066–1074, 2007.

[18] S. Wang, S. Ewert, and S. Dixon. Robust and efficient joint alignment of multiple musical performances. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pages 2132–2145, 2016.

[19] T.-M. Wang, P.-Y. Tsai, and A. W. Y. Su. Note-based alignment using score-driven non-negative matrix factorisation for audio recordings. *IET Signal Processing*, pages 1–9, 2014.

ACOUSTIC FEATURES FOR DETERMINING GOODNESS OF TABLA STROKES

Krish Narang **Preeti Rao**
Department of Electrical Engineering,
Indian Institute of Technology Bombay, Mumbai, India.
krishn@google.com, prao@ee.iitb.ac.in

ABSTRACT

The tabla is an essential component of the Hindustani classical music ensemble and therefore a popular choice with musical instrument learners. Early lessons typically target the mastering of individual strokes from the inventory of bols (spoken syllables corresponding to the distinct strokes) via training in the required articulatory gestures on the right and left drums. Exploiting the close links between the articulation, acoustics and perception of tabla strokes, this paper presents a study of the different timbral qualities that correspond to the correct articulation and to identified common misarticulations of the different bols. We present a dataset created out of correctly articulated and distinct categories of misarticulated strokes, all perceptually verified by an expert. We obtain a system that automatically labels a recording as a good or bad sound, and additionally identifies the precise nature of the misarticulation with a view to providing corrective feedback to the player. We find that acoustic features that are sensitive to the relatively small deviations from the good sound due to poorly articulated strokes are not necessarily the features that have proved successful in the recognition of strokes corresponding to distinct tabla bols as required for music transcription.

1. INTRODUCTION

Traditionally the art of playing the tabla (Indian hand drums) has been passed down by word of mouth, and documentation of the same is rare. Moreover, recent years have seen a decline in the popularity of Indian classical music, possibly due to the relatively limited accessibility options in todays digital age. While tuners are commonly utilized with melodic instruments, a digital tool that assesses the timbre of the produced sound can prove invaluable for learners and players of percussion instruments such as the tabla, in avoiding deep-seated deficiencies that arise from erroneous practice.

Based on the fact that there is an overall consensus among experts when it comes to the quality of sound (in terms of intonation, dynamics and tone quality) produced by an instrumentalist [2], Picas et al. [3] proposed an automatic system for measuring perceptual goodness in instrumental sounds, which was later developed into a community driven framework called good-sounds.org [4]. The website worked with a host of string and wind instruments, whose goodness broadly depended on similar acoustic attributes. We follow the motivation of good-sounds, extending it to a percussive instrument, the tabla, which has a sophisticated palette of basic sounds, each characterized by a distinct vocalized syllable known as a “bol”. Further, in the interest of creating a system that provides meaningful feedback to a learner, we explicitly take into account the link between the manner of playing, or articulatory aspects, and the corresponding acoustic attributes.

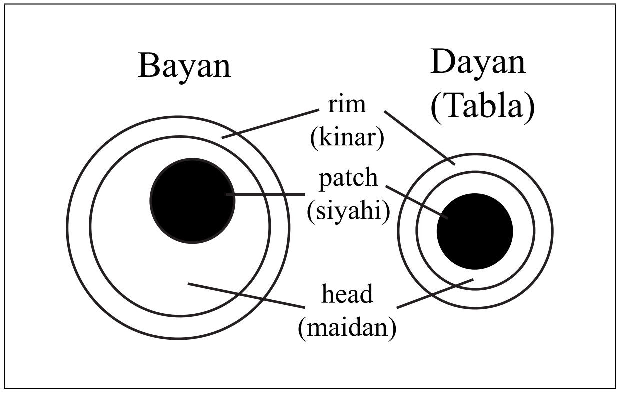


Figure 1. Regions of the left (bayan) and right (dayan) tabla surfaces, Patel and Iversen [1].

The tabla consists of two sealed membranophones with leather heads: the smaller, wooden-shell “dayan’ (treble drum) is played with the right hand, and the larger, metal-shell “bayan’ (bass drum) is played with the left. Each drum surface is divided into regions as shown in Figure 1. Unlike typical percussion instruments that are played with sticks or mallets hit at the fixed place on the drum surface, a tabla stroke is specified by the precise hand gesture to be employed (we term this the “manner of articulation”, borrowing on terminology from speech production) and the particular region of the drum surface to be struck (“place of articulation”). Previous work has addressed the recognition of tabla bols for transcription via the distinct acous-

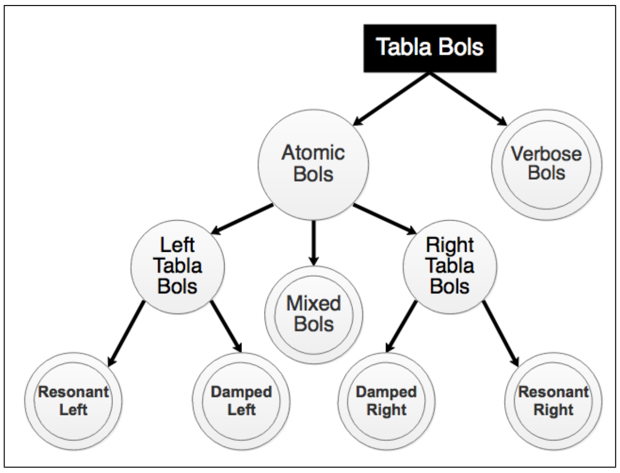


Figure 2. Articulation based classification of tabla bols.

tic characteristics associated with each of the strokes [5,6]. Temporal and spectral features commonly applied to musical instrument identification were used to achieve the classification of segmented strokes corresponding to different bols. Gillet and Richard [5] performed classification of individual bols by fitting Gaussian distributions to the energies in each of four different frequency bands. Chordia [6] used descriptors comprised of generic temporal as well as spectral features commonly used in the field of Music Information Retrieval for bol classification. More recently, Gupta et al. [15] used traditional spectral features, the mel-frequency cepstral coefficients, for the transcription of strokes in a ryhythm pattern extraction task on audio recordings. While the recognition of well-played strokes can benefit from the contrasting sounds corresponding to the different bols, the difference between a well-played and badly-played version of a bol is likely to be more nuanced and require developing bol-specific acoustic features. In fact, Herrera et al. [8] use spectral features for percussion classification based on a taxonomy of shape/material of the beaten object, specifically omitting instruments that drastically change timbre depending on how they are struck.

In this work, we consider the stroke classification problem where we wish to distinguish improperly articulated strokes from correct strokes by the analysis of the audio recording, and further provide feedback on the nature of the misarticulation. Based on a training dataset, that consists of strokes representing various kinds of playing errors typical of learners, as simulated by tabla teachers, we carry out a study of acoustic characteristics in relation to articulation aspects for each stroke. This is used to propose acoustic features that are sensitive to the articulation errors. Traditional features used in tabla bol recognition are used as baseline features and eventually we develop and evaluate a stroke classification system based on the combination of proposed and baseline features in a random forest classifier.

Type	Bol	Label	Position	Manner	Pressure
Resonant Left	Ge	Good	Maidan	Bounce	Variable
		Bad1	Siyahi	Bounce	Medium
		Bad2	Maidan	Press	Medium
		Bad3	Kinar	Bounce	Medium
Damped Left	Ke	Good	Siyahi	Press	Medium
		Bad1	Maidan	Press	Medium
		Bad2	Siyahi	Bounce	Light
Resonant Right	Ta/ Na	Good	Kinar	Press	Medium
		Bad1	Kinar(e)	Press	Heavy
		Bad2	Maidan	Press	Medium
		Bad3	Kinar	Press	Heavy
	Tun	Good	Siyahi	Bounce	None
		Bad1	Siyahi	Press	Light
		Bad2	Maidan	Bounce	None
	Tin	Good	Maidan	Bounce	Light
		Bad1	Siyahi	Bounce	Light
		Bad2	Maidan	Bounce	Heavy
Damped Right	Ti/ Ra	Good	Siyahi	Press	Medium
		Bad1	Siyahi	Bounce	Light
		Bad2	Siyahi(e)	Press	Medium
	Tak	Good	Maidan	Press	Medium
		Bad1	Maidan	Bounce	Light
		Bad2	Kinar(e)	Press	Medium
		Bad3	Siyahi(e)	Press	Medium

Table 1. Common articulations of bols in terms of position of articulation, manner of articulation, and hand pressure.

2. ARTICULATION BASED CLASSIFICATION

The tabla is a set of two drums, the left bass drum (bayan) and the right, higher pitched drum (dayan). Each tabla drum surface is composed of three major regions- siyahi, maidan, and kinar as depicted in Figure 1. Each tabla stroke (bol) is characterized by a very specific combination of the hand orientation with respect to the the position on drum surface, manner of striking, and pressure applied to the drum head, and has a very distinctive sound. Due to the heavy dependence of perceived quality of tabla bols on articulation accuracy of the player, it is instructive to understand the articulatory configurations of bols via the taxonomy visualized in Figure 2. Mixed bols are bols where both tablas are struck simultaneously (e.g. Dha, Dhin Dhit). Verbose bols (e.g. TiNaKeNa) consist of a sequence of strokes played in quick succession, whereas atomic bols are single stroke bols. A resonant bol is one where the skin of the drum is allowed to freely vibrate after it is struck, and a damped bol is one where the skin is muted in some way after it is struck.

Bols of each type (leaf nodes of Figure 2) can further be classified based on the place of articulation, manner of articulation and amount of hand pressure applied on the skin of the tabla. For example, for the bol tun, the index finger strikes the siyahi of the right tabla (dayan), with no damping (hand does not touch the tabla, finger is lifted after striking) (Patel and Iversen [1]). These are the three major attributes that distinguish bols within a type, and are also what decide the perceptual goodness of a tabla stroke. For the same hand orientation, the drum can be struck sharply followed by immediately lifting the finger (we call this the 'bounce' manner of articulation) or it can be struck followed by leaving the finger or palm pressed against the drum head (we call this the 'pressed' manner of articulation). For the rest of the study we focus on atomic bols, which are sufficient for coverage of all beginner tabla rhythms, as listed on raganet, an educational magazine on Indian music [7]. For simplicity, mixed bols are not covered, since they are combinations of simultaneous left and right tabla strokes.

Two tabla teachers were consulted on the common mistakes made by beginners while playing a particular bol. Based on these, multiple classes were defined for each bol using the aforementioned three attributes governing goodness of a bol. One of these classes represents the well-played version of that bol, whereas the others represent the most common deviations that are perceptually distinct from the expected good sound. These are listed for all bols in Table 1, which explicitly shows the position, manner and hand pressure for different articulations of each bol, where "(e)" refers to the edge of the specified region. For example, a resonant right bol played on the maidan, while applying light hand pressure and lifting the finger after striking, constitutes a well-played Tin bol. However the same played while applying medium to heavy hand pressure is a badly-played Tin bol.

3. DATABASE AND ACOUSTIC CHARACTERISTICS

A dataset composed of 626 isolated strokes of 7 different bols was recorded (sampling rate of 44.1 kHz) by two experienced tabla players on a fixed tabla set that was tuned to D4 (294 Hz). The players were asked to play several instances of each stroke while also simulating typical errors that a new learner is likely to make in realizing a given stroke. Thus our dataset consists of recordings of each of bols realized in different ways as listed in Table 1, which also provides an articulation based description of the different realizations as executed by the tabla players. All the recordings were perceptually validated by one of the players who listened to each stroke and labeled it as "good" or "bad". In order to develop a system that provides specific feedback on the quality of a stroke, we required badly played instances of the bols as well. This made it impossible to use a publicly available dataset, as most archived recordings are from professional performances. Also, since our dataset is generated with reference to controlled variations in articulation as typical of a learner, it is likely to be more complete than the randomly sampled acoustic space of all possible productions.

A number of recordings was made per bol as seen in the Count column of Table 3, but with a roughly equal distribution of strokes across the classes corresponding to each bol in order to facilitate the construction of balanced training and test datasets for the classification task. The only exception to this is the bol Ge where a relatively large number of instances of the good stroke were produced since it is the only bol with pitch that can be modulated by changing the amount of pressure applied on the drum surface while striking. A number of such hand pressure based variations were recorded for the correct articulatory settings of the Ge stroke in order to get a reasonably representative dataset for the good quality bol Ge (124 out of the total of 187 Ge strokes in Table 3). This was important to ensure that the classifier we build is robust to pitch variations and other irrelevant changes caused by an increase or decrease in hand pressure.

Since each stroke presented in Table 1 is characterized by specific articulation (in terms of place of articulation, manner of articulation and amount of hand pressure), the acoustic variability is likely to cover more than one dimension. By studying the short-time magitude spectra (i.e. spectrograms) of the recorded bols, we were able to isolate the acoustic characteristics that distinguished the various classes of each bol. Time-domain waveforms and short-time magnitude spectra for two bols, Tin (a resonant right bol) and Ke (a damped left bol) are shown in Figure 3 and Figure 4 respectively. We observe that the rate of decay of the time-domain waveforms clearly discriminate the good from bad strokes. Further, the saliency as well as rate of decay of the individual harmonics (horizontal dark bands in the spectrograms) are seen to differ between the differently realised versions of each of the strokes. The resonant bol Tin is characterised by strong sustained harmonic components for good quality. In contrast, the damped bol Ke has a diffuse spectrum and rapidly decaying temporal envelope when realised correctly in Figure 4 top. A bounce in the hand gesture, on the other hand, degrades the stroke quality, contributing the prominent harmonics seen in the low frequency region of the bottom most bad stroke in Figure 4.

4. DEVISING FEATURES

From acoustic observations similar to those outlined in the previous section, across bols and goodness classes, we hypothesize that the strength, concentration and sustain of particular harmonics is critical to the quality of realization of a bol, especially for the resonant bols. Based on this, we propose and evaluate a harmonics based feature set which we call Feature set A. The features are designed to capture per-harmonic strength, concentration and decay rates. Harmonic based features are computed for each of the first 15 harmonics by extracting the corresponding spectral region by passing the signal through a narrow bandpass filter centered around that harmonic. These are important for resonant bols. The energy, spectral variance, and decay rate of each of the bandpass-filtered signals are computed. The decay rate is obtained as a parameter corresponding to an exponential envelope fitted to the signal. The energy and variance together constitute the strength of the harmonic, whereas decay rate represents how quickly that particular harmonic dies out. Spectral shaping features include variance, skewness, kurtosis and high frequency content. These features are extracted using Essentia [9], an open-

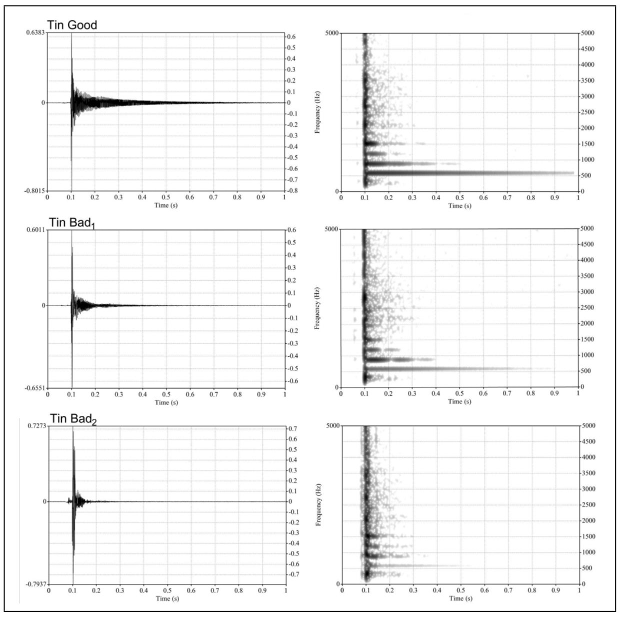


Figure 3. Waveform (left) and spectrogram (right) for good and selected bad recordings of Tin. Bad_1 is played in the wrong position, on the siyahi. Bad_2 is played with excess hand pressure.

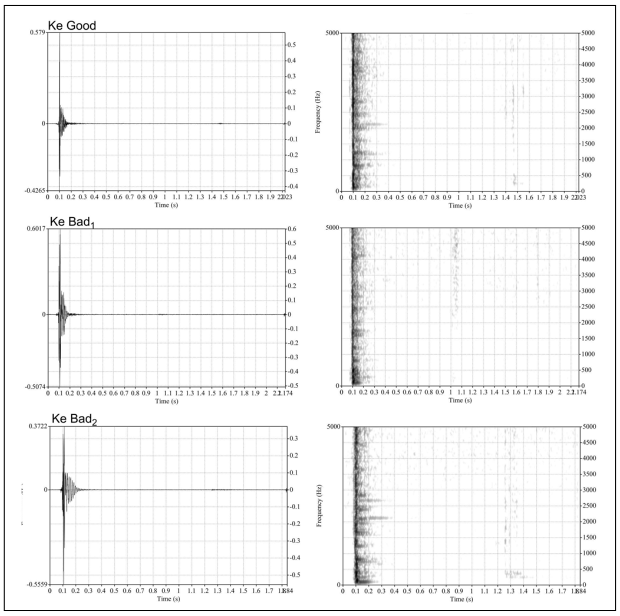


Figure 4. Waveform (left) and spectrogram (right) for good and selected bad recordings of Ke. Bad_1 is played in the wrong position, on the maidan. Bad_2 is played loosely- by bouncing the palm instead of pressing it.

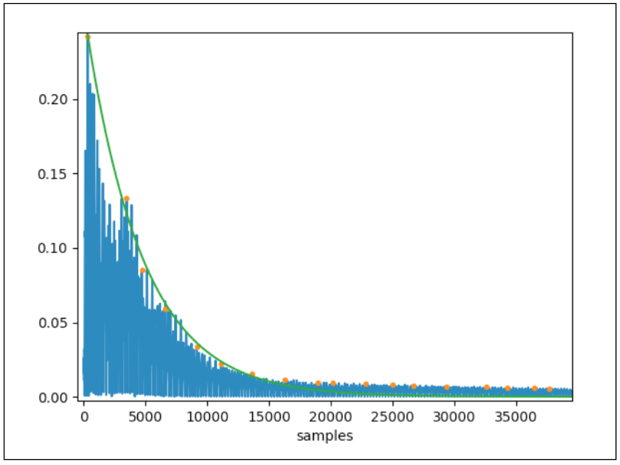


Figure 5. Exponential envelope fitted to rectified waveform for a Ge stroke. Dots mark the retained samples for curve fitting.

source library for audio analysis and audio-based music information retrieval. The temporal features include the energy and decay rate of the signal, and are useful for determining goodness of both damped and resonant bols. We also evaluate a baseline feature set (termed Feature Set B) which is essentially the same as the features employed by Chordia [6] in a tabla bol recognition task.

4.1 Harmonic Based Features

For each resonant bol that is correctly rendered, clear harmonics are visible in the spectrogram at multiples of a fundamental frequency. For resonant bols on the right tabla, the fundamental frequency is equal to the tonic of the tabla, except for Tun, for which the fundamental frequency is two semitones higher than the tonic [10]. However, these are not always precise, and a pitch detection algorithm should be used for determining the fundamental frequency of the recorded bol, e.g. the YinFFT algorithm [11]. For our dataset, the fundamental frequencies were manually estimated by viewing the spectrogram. For the tabla set used in our experiments, the tonic was determined to be 294 Hz, and fundamental frequency for Tun to be 330 Hz. For the left tabla stroke Ge the fundamental frequency was estimated to be 125 Hz.

For extracting harmonic based features, the signal is first passed through fifteen second-order IIR band pass filters with a bandwidths of 100 Hz and center frequencies at multiples of the fundamental frequency for that bol. Then an exponential envelope is fitted to the resulting time domain waveform. The waveform is full-wave rectified ($A'(t) = |A(t)|$), and only the maximum amplitude sample in every 50 millisecond interval is retained (as marked in Figure 5). The onset sample of the signal (assumed to be maximum amplitude sample over all time) is kept at $t = 0$. Next, SciPy's curve_fit function [12] is used to fit an exponential (ae^{-bt}) to the obtained samples, and both parameters a and b are considered as features. a represents the estimated maximum amplitude (referred in our

Bol	Selected Features
Ge	Energy(overall, 250, 500, 750, 1000, 1125, 1625), Decay(overall, 125, 250, 375, 625, 875), Impulse(125), Variance(125, 1500), MFCC(5, 6, 8, 10), Attack Time, Temporal Centroid, ZCR, Spectral Centroid
Ke	Energy(overall, 1764, 2352, 3528, 4116), Decay(overall, 294, 588, 2646, 3822), Impulse(294, 588, 882, 2352, 3234), MFCC(0, 1, 7, 12), Attack Time
Ta/Na	Energy(overall, 294, 1176, 1470), Decay(882), Impulse(2058), Variance(882, 1470, 2352), MFCC(1), Temporal Centroid
Tak	Energy(588, 882, 1176, 1470, 2646, 4116), Decay(294), Impulse(588), Variance(294), MFCC(1, 3), Attack Time, Temporal Centroid
Ti/Ra	Energy(588, 1764), Decay(588, 1176), Impulse(588), Variance(588), MFCC(11, 12), Attack Time, Temporal Centroid, ZCR
Tin	Energy(294, 2352, 3822), Decay(overall, 294, 588, 1470), Impulse(1764), Variance(294, 588), MFCC(2), Temporal Centroid
Tun	Energy(4950), Decay(330, 2310, 3960), Impulse(overall), Spectral Centroid, Temporal Centroid, ZCR

Table 2. Features selected from combination of set A and set B. The numbers in the bracket indicate the harmonic frequencies selected for energy/decay/impulse/variance and the indexes of selected coefficients (0-12) for MFCC.

feature set as 'impulse') of the signal and b represents the estimated decay rate (inversely proportional to the decay time). A similar curve fitting is done to the unfiltered time domain waveform. From the spectrum of the unfiltered signal, we calculate the energy and variance of the spectrum in bands centered around the first 15 harmonics with bandwidth equal to fundamental frequency. Finally the total energy of the signal is also taken as a feature. Finding energy and variance in a particular frequency range and band pass filtering were both done using routines from Essentia [9]. A total of 63 features were extracted in this way.

4.2 Baseline Feature Set

The baseline feature set consists of commonly used temporal and spectral features along with 13 MFCC's. These were used by Chordia [5] for tabla bol classification, and their relevance and effectiveness is also described in detail by Brent [13]. The temporal features are zero crossing rate, temporal centroid (the centroid of the time domain signal) and attack time. The attack time is calculated as time taken for the signal envelope to go from 20% to 90% of its maximum amplitude (default used in Essentia [9]). The spectral features are spectral centroid, skewness, and kurtosis. These are all obtained from the magnitude spectrum computed over the full duration of the recorded stroke. All of these features were computed using Essentia [9] routines.

Bol	Count	Classes	Set A	Set B	Combined Set
Ge	187	4	89.8	89.8	94.1
Ke	67	3	79.1	76.1	85.1
Ta/Na	86	4	89.5	86.1	91.9
Tak	101	4	80.2	82.2	86.1
Ti/Ra	79	3	77.2	96.2	91.1
Tin	48	3	89.6	93.8	97.9
Tun	29	3	81.0	91.4	93.1

Table 3. Percentage classification accuracies (one good class, multiple articulation based bad classes) Accuracies with Harmonic Features (Set A), Baseline Features (Set B), and selected features from a combination of Set A and Set B (Combined Set).

5. TRAINING AND EVALUATION OF BOL ARTICULATION CLASSIFIERS

Given our set of features, engineered as presented in the previous section, and the fact that our dataset is not very large, we employ a random forest classifier for the stroke classification task. A random forest classifier is an ensemble approach based on decision trees. We test for k-way classification accuracy in 10 fold cross validation mode with each of the different feature sets using the Weka [14] data mining software. Here k is the number of classes for a particular bol, consisting of one good class and multiple articulation based bad classes as shown in Table 3 where the number of strokes in the dataset for each bol is provided as well. For each instance the classifier predicts whether a bol is well-played (labeled good) or what mis-articulations were made while playing the bol (labeled as the appropriate bad class). Apart from this, a subset of features is selected from the union of the two feature sets, using the CfsSubsetEval attribute selector with a GreedyStepwise search method from the Weka [14] data mining software. The greedy search picks each succeeding feature based on the classification improvement it brings to the existing set, using a threshold on achieved accuray as a stopping criterion. The set of selected features for each bol is shown in Table 2. Classification accuracies obtained with each of the 3 feature sets are presented in Table 3. We observe that the combination of features performs better than the baseline in nearly all cases. This indicates that the new harmonics based features bring in some useful information, complementary to the baseline features. In the case of the bol Ti/Ra, there is a decrease in classification accuracy with respect to the baseline. This is a damped bol and therefore harmonic features are not as important to it as spectral shaping features; however the issue of decreased accuracy after feature selection needs further investigation. Finally, Table 4 shows the results of two-way classification into good and bad strokes achieved by the combination features.

Bol	Feature Dimension	Accuracy
Ge	25	96.3
Ke	20	95.5
Ta/Na	11	96.5
Tak	13	94.1
Ti/Ra	10	92.4
Tin	12	97.9
Tun	8	93.1

Table 4. Percentage classification accuracies for two-way classification (good/bad stroke) based on features selected from the combined data set (as listed in Table 2).

6. CONCLUSION

Unlike many percussion instruments, the tabla is a musical instrument with a diverse inventory of basic sounds that demand extensive training and skill on the part of a player to elicit correctly. We proposed a taxonomy of strokes in terms of the main dimensions of articulation obtained through discussions with tabla teachers. This allowed us to construct a representative dataset of correct and common incorrectly articulated strokes by systematically modifying the articulatory dimensions. The results of this study show that nuanced changes in articulation are linked to perceptually significant changes in the acoustics of a tabla stroke. We presented acoustic features extracted from the isolated stroke segments to detect the articulation accuracy and therefore the perceptual goodness of a stroke from its audio. The best choice of features was observed to depend on the nature of the bol.

The present dataset was restricted to a single tabla set. For future work we would like to continue this research using a larger database from more sources, and to include coverage of mixed bols. The latter would further require measurements of relative timing between the atomic strokes that make up the mixed bol. This study can also easily be extended to evaluate sequences of bols (talas) for beginners- by a combination of rhythm scoring and evaluation of segmented bols of the sequence individually. The concept of expression and emotion in the playing of the tabla, which is vital to intermediate and expert players, is however a much more open ended question, and further research will hopefully lead to a characterization of that problem as well.

7. ACKNOWLEDGEMENT

The authors would like to thank Digant Patil and Abhisekh Sankaran for their help in recording and evaluating the Tabla strokes dataset. We are also indebted to Kaustuv Kanti Ganguli for lending his expertise in Indian Classical music, to Hitesh Tulsiani for his help with feature selection algorithms and classifiers, and to Shreya Arora for editing and rendering of graphs and images.

8. REFERENCES

[1] A. Patel and J. Iversen. "Acoustic and Perceptual Comparison of Speech and Drum Sounds in the North Indian Tabla Tradition: An Empirical Study of Sound Symbolism," *Journal of Research in Music Education, vol. 46*, pp. 522–534, 1998.

[2] J. Geringe and C. Madsen. "Musicians ratings of good versus bad vocal and string performances," *Proc. of the 15th international congress of phonetic sciences (ICPhS)*, pp. 925–928, 2003.

[3] O. Roman Picas, H. Parra Rodriguez, D. Dabiri, H. Tokuda, W. Hariya, K. Oishi, and X. Serra. "A real-time system for measuring sound goodness in instrumental sounds," *Audio Engineering Society Convention 138*, Audio Engineering Society, 2015.

[4] G. Bandiera, O. Roman Picas, H. Tokuda, W. Hariya, K. Oishi, and X. Serra. "good-sounds. org: a Framework To Explore Goodness in Instrumental Sounds," *Proc. 17th International Society for Music Information Retrieval Conference*, pp. 414–419, 2016.

[5] O. Gillet, and G. Richard. "Automatic labelling of tabla signals," *Johns Hopkins University*, 2003.

[6] P. Chordia. "Segmentation and Recognition of Tabla Strokes," *ISMIR*, pp. 107–114, 2005.

[7] A Batish. "Tabla Lesson 8 - Some Popular Tabla Thekas," *Batish Institute of Indian Music and Fine Arts*, `http://raganet.com/Issues/8/tabla8.html`, 2003.

[8] P. Herrera, A. Yeterian, and F. Gouyon. "Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques," *Music and Artificial Intelligence*, pp. 69–80, 2002.

[9] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra. "Essentia: An Audio Analysis Library for Music Information Retrieval," *ISMIR*, pp. 493–498, 2013.

[10] C. V. Raman. "The Indian musical drums," *Proc. of the Indian Academy of Sciences - Section A*, pp. 179–188, 1934.

[11] P. M. Brossier. "Automatic annotation of musical audio for interactive applications," *Diss. Queen Mary, University of London*, 2006.

[12] E. Jones, T. Oliphant, P. Peterson and others. "SciPy: Open Source Scientific Tools for Python," `http://www.scipy.org/`, 2001.

[13] W. Brent. "Physical and perceptual aspects of percussive timbre," *UC San Diego Electronic Theses and Dissertations*, 2010.

[14] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. "Data Mining: Practical machine learning tools and techniques," *Morgan Kaufmann*, 2016.

[15] S. Gupta, A. Srinivasamurthy, M. Kumar, H. A. Murthy, X. Serra. ”Discovery of Syllabic Percussion Patterns in Tabla Solo Recordings,” *Proc. of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

AUTOMATIC SAMPLE DETECTION IN POLYPHONIC MUSIC

Siddharth Gururani, Alexander Lerch
Georgia Institute of Technology, Center for Music Technology
{siddgururani, alexander.lerch}@gatech.edu

ABSTRACT

The term 'sampling' refers to the usage of snippets or loops from existing songs or sample libraries in new songs, mashups, or other music productions. The ability to automatically detect sampling in music is, for instance, beneficial for studies tracking artist influences geographically and temporally. We present a method based on Non-negative Matrix Factorization (NMF) and Dynamic Time Warping (DTW) for the automatic detection of a sample in a pool of songs. The method comprises of two processing steps: first, the DTW alignment path between NMF activations of a song and query sample is computed. Second, features are extracted from this path and used to train a Random Forest classifier to detect the presence of the sample. The method is able to identify samples that are pitch shifted and/or time stretched with approximately 63% F-measure. We evaluate this method against a new publicly available dataset of real-world sample and song pairs.

1. INTRODUCTION

In the context of music composition and production, *sampling* stands for the concept of reusing pre-existing digital recordings in new compositions in a way that it fits the musical context. In digital sampling, an artist records a segment of a song or sound that they wish to sample, possibly modifies it, and then reuses it (and possibly other samples) by incorporating it into a new composition [11]. Sampling of audio has become popular in mainstream pop, hip-hop, and rap music.

A Sample Detection (SD) system automatically detects samples from a pool of songs and thus enables musicological studies of the influence of older artists over newer generation artists by observing sampling patterns over the years and geographically. Another possible use case of a SD system could be to detect plagiarism or copyright infringement. Sampling is legally controversial and determining fair use is largely left to the judicial system. A system that gives an objective measure of the likelihood of a sample being present in an audio file could add weight to either party's argument in a lawsuit.

The algorithm discussed in this paper focuses on solving the problem of detecting the presence of a given sample in a set of songs as well as its time location in the song.

2. RELATED WORK

The task of SD has been addressed in only few previous publications. There exist, however, several parallels that may be drawn from other areas of research that are relevant to SD such as cover song detection, audio fingerprinting, and remix recognition.

2.1 Audio Fingerprinting

Audio Fingerprinting (AFP) refers to the method of extracting content-based signatures from audio [1, 5, 10]. It is commonly used in content-based music identification systems such as Shazam. [1] Van Balen proposed the use of AFP for sample detection [24,25], using a popular fingerprinting algorithm by Wang [27] in an implementation by Ellis [8].

Fingerprinting systems are robust against noise injection and, with appropriate modifications, pitch shifting and time stretching [31]. As long as the level difference between the sample and other sources is within the noise level expectations, a modified system could be a good choice for sample detection. Given that a sample might be mixed at a low level, it is questionable if this assumption really holds true for the majority of cases.

2.2 Cover Song Detection

Cover Song Detection (CSD) is the task of recognizing whether a given reference track has a cover song in a set of test tracks [2,9,19]. Covers may, for example, be transposed and deviate from the original song in terms of tempo and other properties. Dynamic Time Warping (DTW) [20] is often used to make the comparison tempo-invariant. The difference from SD is that covers are renditions of a musical piece, while samples are snippets of audio which are usually a part of the mix overlaid with multiple other instruments and sounds unrelated to the sample. The evaluation of SD systems, however, is similar to that of CSD. Both have a test/reference pair which is then categorized as a positive or negative match with or without a confidence measure.

2.3 Remix Recognition

Work done in remix recognition by Casey and Slaney [6] draws inspiration from a method for web crawling called

[1] https://www.shazam.com, last accessed: 4/26/2017

'shingling' which utilizes a stream of text position-based features to detect if a document has already been crawled before. In their work, they compute audio features such as pitch-class profiles and Log-Frequency Cepstral Coefficients using 0.1 second frames. They concatenate these feature vectors into 4 second 'shingles' and model the distribution of pair-wise distances between shingles from remixes and shingles that are not from remixes. A nearest neighbor classifier is used to identify which distribution a query 'shingle' belongs to. Such a system is not appropriate for SD since this method relies on long-term similarity between the query and the reference, however, a sample may be a short, one-shot sample (triggered samples without looping). A working remix recognition system, however, would be helpful in ways similar to SD in that it helps with tracing influences across artists. In addition, some remixing cases might also involve instances of sampling.

2.4 Sample Detection with Non-negative Matrix Factorization

Dittmar et al. listed sampling as one of three kinds of plagiarism in music [7]. They utilize Non-negative Matrix Factorization (NMF) to learn the spectral templates from the sample and detect the presence of these templates in the suspect audio. Correlating the activations from the sample and the song then gives the likelihood of plagiarism. While the authors provide a general outline of a sample detection system, they neither offer a detailed algorithmic description nor a formal evaluation of their proposed system. In [28], Whitney uses NMF in a similar manner except that instead of factorizing entire spectrograms, NMF is applied to short texture windows in the sample and the song. The detection is done using pairwise 2-dimensional cross-correlation of the two activation matrices obtained. To account for pitch shifting and time stretching, the audio files are resampled using factors computed by taking the ratio of the sample and song BPM and multiple NMFs are performed. The issue with this approach is that the time stretching and pitch shift factor are not necessarily inverse when sampling. Nonetheless, NMF appears to be a good choice for a SD system as fixed templates allow to obtain activations for only the common components between the song and the sample.

The task of sample detection has been identified in music information retrieval literature, and various approaches have been proposed. However, it has not yet been well defined in terms of evaluation methodology or metrics. Reasonably sized datasets are also non-existent or proprietary. Therefore, there is no formal evaluation that can be performed to compare different sample detection methods. This paper aims to bridge this gap by providing a dataset and propose a common evaluation framework.

3. METHOD

The algorithm presented in this paper is inspired by the proposal of Dittmar et al. [7]. Since the task of sample detection is similar to a source identification problem where the sample is one of the sources present in the mix, an NMF-based approach is fitting due to its prevalence in audio source separation tasks [18, 21]. The block diagram in Figure 1 shows the processing steps of the algorithm.

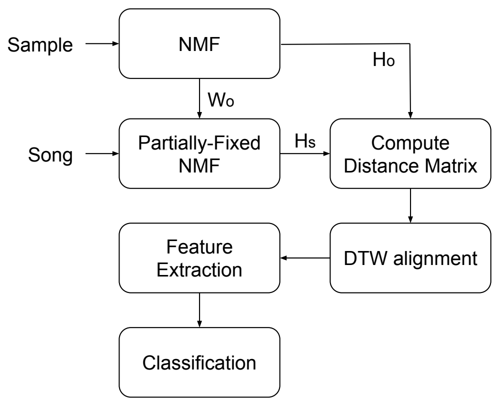


Figure 1. High Level Block Diagram

3.1 Non-Negative Matrix Factorization

NMF is a widely popular algorithm in unsupervised learning with applications in recommendation systems [12, 15, 16] and signal processing tasks, specifically source separation [13, 23, 26]. NMF factorizes a signal matrix $V \in \mathbb{R}^{M\times N}$ into a template matrix $W \in \mathbb{R}^{M\times K}$ and an activation matrix $H \in \mathbb{R}^{K\times N}$ such that:

$$V \approx W \cdot H. \tag{1}$$

If V is the magnitude spectrogram with M frequency bins and N blocks, W contains the K spectral or harmonic component templates in V while H contains temporal information about each corresponding spectral component in the template matrix [22].

In a pre-processing step, both the original sample and the paired song are RMS-normalized, downmixed, and downsampled to 22.05 kHz; then, their magnitude spectrogram is computed (block size: 4096, hop size: 1024 samples). Using NMF, the sample spectrogram will be factorized into K templates W_o and the activation matrix H_o, o indicating 'original'. A sample, used in a song, may be thought of as one source in a mixture of multiple sources in this song. The factorization of the song will then be performed using partially fixed NMF (PFNMF) [29, 30]. In this case, the template matrix W consists of a fixed, not updated, part containing the extracted templates W_o and a randomly initialized part W_m with L templates that is iteratively learned and represents what we refer to as the mixture templates. The dimension of the complete template matrix is thus $M \times (K + L)$. All activations are iteratively updated as well.

3.1.1 NMF Rank Selection

The rank K of the sample spectrogram has to be chosen based on how many spectral templates can approximate the

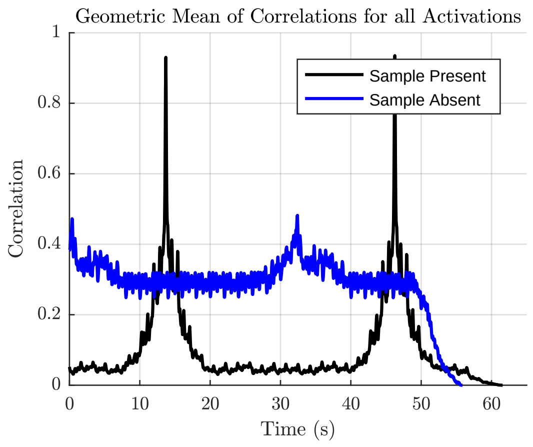


Figure 2. Geometric mean of cross-correlation functions. Black function shows sample occurrence, while blue does not have any samples.

specific sample. Similarly, while computing the PFNMF, the rank L for approximating the templates representing the remaining mixture in the song has to be selected in a way that minimizes the impact on the fixed sample template activations in order to robustly detect the sample.

Different songs will usually require different ranks for accurately approximating and factorizing their magnitude spectrograms with a low reconstruction error. In the current algorithm, however, fixed ranks are chosen: $K = 10$ and $L = 20$. The rationale behind using fixed low ranks is that for this task a perfect reconstruction is not required and that the following processing steps should be robust enough to detect the sample regardless of whether the templates are able to combine linearly to an accurate reconstruction of the original spectrogram. If the sample is used in a song, the same fixed templates should produce a roughly similar set of activations independent of the mixture rank. Larger fixed ranks were tested but no gain in performance was observed while resulting in increased computational cost.

Still, a future extension might be to analyze the audio separately as a pre-processing step to obtain a ‘complexity’ measure that could be used to adapt the ranks K and L based on the signals to be modeled.

3.2 Activation Function Processing

In the subsequent analysis, only the activations H_s are of interest. These are the activations corresponding to the original templates W_o after PFNMF and indicate the presence of the sample in the song if they match the pattern of the original activations H_o. If the sample were neither pitch shifted nor time stretched, cross-correlation functions between each corresponding activation in H_o and H_s could be computed. Peaks in the aggregated (across the K dimensions) cross-correlation function would then indicate the presence of the sample. Figure 2 shows two example functions after using the geometric mean for aggregation: a sample occurs twice in the black song (as indicated by the two local maxima), while it is not present in the blue song.

3.2.1 Activation Normalization

Proper normalization of the activation matrix is essential for accurate sample detection, however, there is no “correct” way to do this as the level (or even the presence) of the sample in the paired song is unknown. The relative activation levels between the K templates of one matrix, however, should remain identical. Thus, each activation matrix H is normalized by the absolute maximum across all the K activations across time, preserving the relative activation strengths for the spectral templates of the sample: $H_{normalized} = \frac{H}{\max(H)}$.

3.2.2 Pitch Shifting & Time Stretching

The assumption that the sample is neither time stretched nor pitch shifted is false for the majority of cases. In the dataset used for this study, for example, 57.5% of samples are pitch shifted. Pitch shifting is often required for the sample to match the tonality of the song, and time stretching is often required to adjust for tempo differences between the two.

In case of pitch shifting, the original templates W_o will no longer be valid templates since the frequency axis of the spectral content will be scaled by the pitch shift factor. In order to account for pitch shifting, we construct new spectral templates W_o^p by scaling the frequency axis of the original templates with a set of hypothetical pitch shift factors. Now, a partially fixed NMF will be able to extract activations corresponding to the pitch shifted templates and these may be compared to the activations from the original sample. Ideally, we would perform factorizations for a set of hypothetical pitch shifts usually ranging from one octave lower to one octave higher in semi-tone or quarter-tone steps. For this paper, however, we allow the dataset to inform our pitch shift steps. The used pitch shift set in semi-tone steps from the original sample templates W_o is:

$$P = \{p \mid p \in \{-5, -4, -3, -2, -1, 0, 0.5, 1, 2, 3, 4, 5\}\}.$$

Partially-fixed NMF is then performed individually for each pitch shift in P, i.e., 12 times. The activations H_s^p correspond to the pitch shifted templates W_o^p.

When the sample is time stretched, the activations from the song will be stretched or compressed accordingly; therefore, cross-correlation cannot be used. In such a scenario, DTW can align the activations H_o with the activations H_s^p in the song for each pitch shift p. A distance matrix D is constructed using the pair-wise correlation distance between the K-dimensional activations. The size of D is $N_\mathrm{o} \times N_\mathrm{s}$, where N_o is the number of frames in the original sample activations and N_s is the number of frames in the song activations. Correlation is used as the distance measure because it is scale independent. Preliminary tests showed that other frequently used distance measures perform either similarly or worse. The resulting distance matrix shows how similar a set of activations is to the extracted activations at each time instant. The distance matrix D and properties of the extracted path can be used to indicate the presence of a sample in a song. Figure 3 shows an example of a distance matrix with a looped sample. The low cost parallel blue paths indicate the presence of the sample.

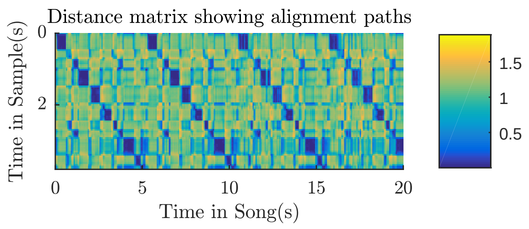


Figure 3. Distance matrix computed between activations in the case where a sample is looped

The problem is now a subsequence search for the sample activations H_o within the series of activations corresponding to the sample templates in the song, H_s^p. A standard DTW implementation would initialize the cost matrix C by accumulating the first row and first column of the distance matrix D. Such a scheme can be applied when only one global optimal path that aligns the two entire time series is required. In the case of sample detection, a sample could be present in multiple locations within the song; hence, the detection of multiple alignment paths at multiple locations in D is required. Each of these paths should align the entire sample with segments of the song. Therefore, the initialization of C is modified to only accumulate the distances along the dimension of the sample, which in our case is the column. This subsequence DTW scheme [17] initializes the cost matrix C as follows:

$$\begin{aligned} C(1,j) &= D(1,j) \\ C(i,1) &= \sum_{k=1}^{i} D(k,i) \end{aligned} \tag{2}$$

Initializing in this fashion and proceeding to compute the cost matrix allows us to obtain backtracking paths from every index in the last row of D (as opposed to one path from the last element of D, which is the case for "standard" DTW). The last row of the cost matrix C corresponds to the cost of aligning the sample backtracking from every frame of the song. The backtracking paths now satisfy the requirement of aligning the entire sample with a section in the song. To summarize, every alignment path obtained is the path that aligns the sample activations H_o backtracking from every frame f in the song activations H_s^p. Note that the subsequence DTW is performed for each $p \in P$.

3.2.3 Pitch Candidate Selection

From the set of pitch shifts, the most likely pitch shift is inferred before feature extraction. Of the 12 cost matrices, the one corresponding to the most likely pitch candidate will be the one with the global minimum cost (minimum of the last row of the cost matrix),i.e., the one with a minimum cost lower than the minimum cost of all other matrices. All subsequent computations are based on the activation matrix of the selected candidate; the results for the 11 remaining pitch shifted templates are discarded.

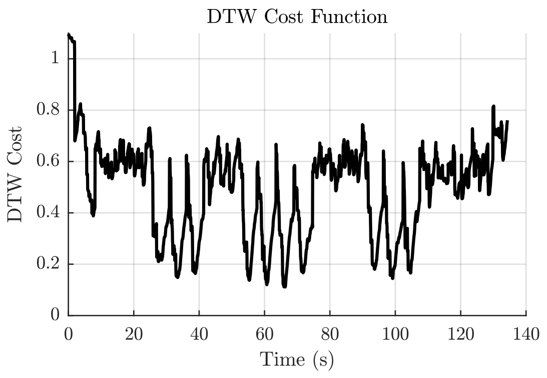


Figure 4. DTW cost function; minima indicate the end of the sample

3.3 Feature Extraction

The last row of the cost matrix C, containing the alignment path costs normalized by the length of the path, will be referred to as *DTW cost function*. Local minima in this DTW cost function indicate potential sample end points.

Using an absolute threshold on the DTW cost function to detect a sample is not meaningful because the absolute cost level depends on both sample and song characteristics. The mixing ratio of different samples in different songs will be different. Some samples might occur in a section with no other sources while others might be heavily overlaid with other sounds, leading to interference and varying strength in activations across different (song, sample) pairs. The DTW backtracking paths from all possible end points to their corresponding start location results in a set of unique start locations in the song for each (song, sample) pair. We refer to this mapping from every end point to a start point in the song as *DTW path start function*. Every unique start location is a candidate for a sample being present. Note that each unique start location can have multiple paths/end points. Figure 5 shows one example of this function. We choose to derive features from the two functions we defined above because we expect that the properties of the DTW alignment paths vary depending on whether a sample is present or not. The features extracted from this data are explained in the following sections.

3.3.1 Cost-based Features

From the multiple path end points of each start location, the following three features are extracted: (i) the minimum DTW cost across all end points, (ii) the average DTW cost across all end points, and (iii) the standard deviation of the DTW cost across all end points.

3.3.2 Path-based Features

The properties of the DTW alignment path should be meaningful for detecting the presence of the sample. For example, the tempo of the sample should stay roughly constant, meaning that the slope of the path stays constant as well. Thus, the idealized path would connect start point and end point with a straight line. Note that when we refer to end

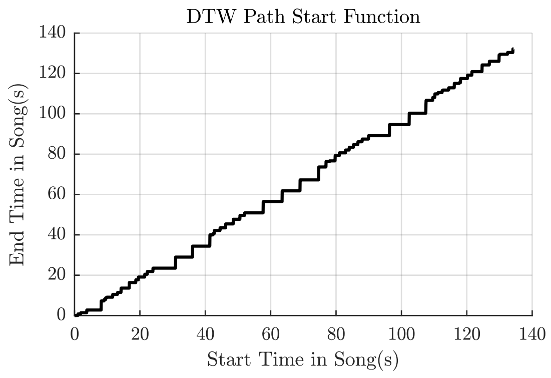


Figure 5. DTW path start function; Longer steps indicate sample; Long steps indicate that several DTW paths back-tracked to the same start point.

points here, we refer to the group of end points that map to one unique start location. Overall, the following features are extracted for every unique start location: (i) the absolute length of the minimum cost path normalized by the sample length, (ii) the slope of the minimum cost path, (iii) the average perpendicular deviation of the minimum cost path from the idealized path, normalized by the length of the path, (iv) the average slope across all end point paths, (v) the standard deviation of the slope across all end point paths, (vi) the average absolute length of all end point paths normalized by the sample length, (vii) the standard deviation of the absolute length of all end point paths normalized by the sample length, (viii) the average perpendicular deviation from the idealized path across all end point paths, normalized by the length of the paths, (ix) the standard deviation of the perpendicular deviation from the idealized path across all end point paths, normalized by the length of the path, and (x) the number of end points mapping to this unique start location.

3.4 Classification

Each unique start location is represented by a 13-dimensional feature vector, which is a data point that can be used as the input to a binary classifier for detecting whether a sample is present or not. A random forest classifier with an ensemble of 200 decision trees was chosen [3]. The number of features chosen for each decision split is 4 and has been decided based on the convention of choosing $round(\sqrt{\mathrm{n}})$ where n is the number of features, 13. The output is a probability of the data point belonging to each class.

4. EVALUATION

4.1 Dataset

A dataset for Sample Detection was compiled using data from whosampled.com[2] which aggregates information about songs that sample or cover other songs. The audio was downloaded using web services from streaming websites such as Youtube or Dailymotion.

[2] www.whosampled.com, last accessed: 1/22/2017

Eighty popular samples (according to the users of Whosampled) were selected from the catalog of songs by popular and frequently sampled artists such as James Brown, Stevie Wonder, Michael Jackson, and Queen. The resulting set has a balanced distribution among the genres Pop, Rock, Funk and Hip-Hop.

The samples cover several variations of sampling such as: one-shot samples of musical snippets or voice samples, looped drums, and looped melodies. The length of the longest sample is 25 s, the shortest is half a second and the average length of the samples is 4.5 s seconds. The total number of sampling instances is 876. The overall dataset contains 80 pairs of original song and sampling song.

The following annotations were added manually with the software Sonic Visualizer [4]: (i) start and end time in seconds of the sample in the original song, (ii) start time in seconds of the sample in the sampling song, and (iii) pitch shift (in semi-tones) of the sample in the sampling song. 57.5% of the samples are pitch shifted. Pitch shift was annotated by a human listener by ear. In cases where the pitch shift was difficult to ascertain, the sample and song snippet was compared in a DAW and different pitch shifts were tested until a match was found. These annotations plus additional meta-data including the song names, identifiers, and URLs for obtaining the audio have been made available publicly in an online repository.[3] The repository also contains the MATLAB source code for the algorithm.

4.2 Experiments

For each of the 80 samples in our dataset, there are 79 songs in which the sample does not occur. We randomly pick 9 songs from these 79 songs. This, in combination with the one song that includes the sample, results in a pool of 10 songs to be paired with each sample, resulting in an overall number of queries: $80 \cdot 10 = 800$ for the 80 samples. Performing experiments with all possible pairs in the entire dataset would be impractical without more compute power. This overall set is split into a training set of 50 samples/500 queries and a test set of 30 samples/300 queries.

In order to use the ground truth data for training, one additional step of interaction is necessary: all unique start points have to be labeled as '0' or '1' based on whether a sample is present. More specifically, the start points associated with minimum DTW cost within a 1 s window of the ground truth annotation are labeled '1' and the rest are labeled '0'. Multiple start points within the tolerance range are merged so that the minimum cost path remains.

For the test set, any positive detection of the sample within a 1 s tolerance window of the ground truth annotation will be regarded as True Positive. Every positive detection outside of this tolerance window and for queries not containing the sample will be regarded as False Positive.

With respect to the metrics, the False Positive Rate, Precision, Recall, and the F-measure are reported for the sample location detection. We refer to these as *Micro*-accuracy measures as they take into account the sample location and the number of occurrences.

[3] https://github.com/SiddGururani/sample_detection

	Precision	**Recall**	**F-measure**	**Fp-rate**
Micro	79.37%	34.60%	48.19%	0.04%
Macro	83.33%	50.00%	62.50%	1.11%

Table 1. Results for song-level macro accuracy and sample location-level micro accuracy measures

Macro-accuracy measures, on the other hand, report the song-level sample detection results and indicate whether a sample is present in a song or not regardless of position and number of occurrences. The same metrics are reported, but the sample detection is evaluated per song rather than per sample instance. In summary, using both the Macro and Micro-level accuracy metrics we are able to report the performance of the method in two usage scenarios: First, detecting whether a sample is present in a song (Macro), and second, detecting where in a song and how often a given sample is present (Micro).

5. RESULTS AND DISCUSSION

Table 1 reports the test accuracy of the classifier. The results show that the presented method is somewhat effective at detecting the presence of sampling in a set of songs with a low false-positive rate and a reasonably high precision. The low recall of the method can be attributed to the highly imbalanced nature of the problem as depicted in the confusion matrix in Table 5: the testing dataset has 289 instances of sampling against around 74,000 instances that are not locations of sampling. The training set is similarly skewed in its distribution of positive and negative classes. We observe an area under receiver operating characteristic curve (AUROC) of 72.54%, a result strongly impacted by the low false positive rate due to the imbalanced classes.

A possible reason for the low recall is also that the method is not always accurate when it comes to picking candidates. More specifically, it already misses approximately 7% of sampling instances during the candidate selection stage. This number is computed by classifying all unique start locations as sample instances and calculating the number of false negatives. A closer investigation of some training samples showed that sometimes the DTW cost function did not have a minimum at respective sample locations. In these cases, the distance matrix would not contain clear alignment paths like the one shown in Figure 3. The absence of clear alignment paths might stem from incorrect modeling of the fixed sample templates in the song NMF step or pitch shifts not considered in our algorithm. The choice of the distance measure might also impact the results: while the use of the correlation distance makes sense because it is scale invariant, custom distance measures might outperform it in this particular use-case.

It is worth pointing out that most problematic cases that we came across were hard to detect for humans as well. These cases included sparse drum loops, very short samples, samples that were mixed at a very low level, and samples with excessive use of audio effects applied to them. The high precision of our method, especially in the macro-accuracy use case, enables utilizing this system as a preprocessing step to a manual detection of sampling instances, e.g., for studies that trace artist influences. Given a database of songs and a set of samples to look for in the database, this method can be used to pre-label data as cases of sampling with high confidence on songs where samples are detected, allowing the human operator to focus on the remaining database. In the context of plagiarism detection, a high precision enables such a system to be used with high confidence in case of a positive detection of plagiarism. However, a low recall system "favors" the sampling artists so the involvement of human experts remains a requirement.

Micro	Not Sample	Sample
Not Sample	74126	26
Sample	189	100

Macro	No Sample	Has Sample
No Sample	267	3
Has Sample	15	15

Table 2. Confusion Matrices for Micro & Macro Accuracy

6. CONCLUSION AND FUTURE WORK

We introduced a method to detect the presence of a sample in a set of songs robust against common sampling modifications such as pitch shifting and time stretching. PFNMF is used to extract sample activations from the song, and DTW is used to align the set of activations obtained from the song and the sample. We also present a new publicly available dataset of real-world samples and songs containing fine-grained annotations for exact time locations of sample occurrences within the song. The presented method is evaluated against this dataset and we obtain 79.37% precision in detecting the exact location of the sample and 83.33% precision in song-level detection of a given sample.

Further research is required in order to improve the usability of this method. Since the algorithm works for many cases, a systematic way to improve it would be to more closely investigate the problematic cases in order to design modifications and algorithmic extensions to increase recall.

As this problem is inherently unbalanced, a possible direction is to observe best practices for machine learning on imbalanced datasets. In addition to undersampling, other techniques such as algorithmic modifications and cost-sensitive learning that may be employed to solve imbalanced classification problems [14].

Investigating custom distance measures for the DTW is another possible avenue to explore. Applying a non-linearity to the NMF activations may help in increasing the sparsity, possibly improving the differentiation between an instance containing a sample and one that does not.

Sample detection is an intriguing and challenging, yet largely untouched MIR task and it is our hope that the dataset and this paper will encourage future work on this topic in the MIR community.

7. REFERENCES

[1] Shumeet Baluja and Michele Covell. Audio fingerprinting: Combining computer vision & data stream processing. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 213–216, Honolulu, HI, USA, 2007.

[2] Thierry Bertin-Mahieux and Daniel P. W. Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 117–120, New Paltz, NY, USA, 2011.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Chris Cannam, Christian Landone, and Mark Sandler. Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files. In *Proc. of the ACM International Conference on Multimedia*, pages 1467–1468, Firenze, Italy, October 2010.

[5] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of audio fingerprinting. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 41(3):271–284, 2005.

[6] Michael Casey and Malcolm Slaney. Fast recognition of remixed music audio. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1425–1428, Honolulu, HI, USA, 2007.

[7] Christian Dittmar, Kay F Hildebrand, Daniel Gaertner, Manuel Winges, Florian Müller, and Patrick Aichroth. Audio forensics meets music information retrieval — A toolbox for inspection of music plagiarism. In *Proc. of the European Signal Processing Conference*, pages 1249–1253, Bucharest, Romania, 2012.

[8] Daniel P. W. Ellis. Robust landmark-based audio fingerprinting. `http://labrosa.ee.columbia.edu/matlab/fingerprint/`. Online; accessed: 28-April-2017.

[9] Daniel P. W. Ellis and Graham E. Poliner. Identifying 'cover songs' with chroma features and dynamic programming beat tracking. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1429–1432, Honolulu, HI, USA, 2007.

[10] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 107–115, Paris, France, 2002.

[11] Mark Katz. *Capturing Sound: How Technology has Changed Music*. University of California Press, Berkeley and Los Angeles, 2004.

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[13] Daniel D Lee and H Sebastian Seung. Learning the parts of sbjects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[14] Victoria López, Alberto Fernández, Salvador García, Vasile Palade, and Francisco Herrera. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

[15] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284, 2014.

[16] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: Social recommendation using probabilistic matrix factorization. In *Proc. of the ACM Conference on Information and Knowledge Management*, pages 931–940, Napa Valley, CA, USA, 2008.

[17] Meinard Müller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[18] Alexey Ozerov and Cédric Févotte. Multichannel non-negative matrix factorization in convolutive mixtures for audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):550–563, 2010.

[19] Suman Ravuri and Daniel P.W. Ellis. Cover song detection: From high scores to general classification. In *Proc. of the IEEE International Conference on Acoustics Speech and Signal Processing*, pages 65–68, Dallas, TX, USA, 2010.

[20] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[21] Mikkel N Schmidt and Morten Mørup. Nonnegative matrix factor 2-d deconvolution for blind single channel source separation. In *Proc. of the International Conference on Independent Component Analysis and Blind Source Separation*, pages 700–707, Chareston, SC, USA, 2006. Springer.

[22] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, New Paltz, NY, USA, 2003.

[23] Paris Smaragdis, Cedric Fevotte, Gautham J Mysore, Nasser Mohammadiha, and Matthew Hoffman. Static and dynamic source separation using nonnegative factorizations: A unified view. *IEEE Signal Processing Magazine*, 31(3):66–75, 2014.

[24] Jan Van Balen. Automatic recognition of samples in musical audio. Master's thesis, Universitat Pompeu Fabra, 2011.

[25] Jan van Balen, Martín Haro, and Joan Serrà. Automatic identification of samples in hip hop music. In *Proc. of the 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, pages 544–551, London, UK, 2012.

[26] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE transactions on Audio, Speech, and Language Processing*, 15(3):1066–1074, 2007.

[27] Avery Wang. An industrial-strength audio search algorithm. In *Proc. of the International Conference on Music Information Retrieval*, pages 7–13, Baltimore, MD, USA, 2003.

[28] Jordan L Whitney. *Automatic Recognition of Samples in Hip-Hop Music Through Non-Negative Matrix Factorization*. PhD thesis, University of Miami, 2013.

[29] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization. In *Proc. of the European Signal Processing Conference*, pages 1281–1285, Nice, France, 2015. EURASIP.

[30] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, Malaga, Spain, 2015.

[31] Bilei Zhu, Wei Li, Zhurong Wang, and Xiangyang Xue. A novel audio fingerprinting method robust to time scale modification and pitch shifting. In *Proc. of the ACM International Conference on Multimedia*, pages 987–990, Firenze, Italy, 2010. ACM.

CHORD RECOGNITION IN SYMBOLIC MUSIC USING SEMI-MARKOV CONDITIONAL RANDOM FIELDS

Kristen Masada
School of EECS
Ohio University, Athens, OH
km942412@ohio.edu

Razvan Bunescu
School of EECS
Ohio University, Athens, OH
bunescu@ohio.edu

ABSTRACT

Chord recognition is a fundamental task in the harmonic analysis of tonal music, in which music is processed into a sequence of segments such that the notes in each segment are consistent with a corresponding chord label. We propose a machine learning model for chord recognition that uses semi-Markov Conditional Random Fields (semi-CRFs) to perform a joint segmentation and labeling of symbolic music. One benefit of using a semi-Markov model is that it enables the utilization of segment-level features, such as segment purity and chord coverage, that capture the extent to which the events in an entire segment of music are compatible with a candidate chord label. Correspondingly, we develop a rich set of segment-level features for a semi-CRF model that also incorporates the likelihood of a large number of chord-to-chord transitions. Evaluations on a dataset of Bach chorales and a corpus of theme and variations for piano by Beethoven and Mozart show that the proposed semi-CRF model outperforms a discriminatively trained Hidden Markov Model (HMM) that does sequential labeling of sounding events, thus demonstrating the suitability of semi-Markov models for joint segmentation and labeling of music.

1. INTRODUCTION AND MOTIVATION

Harmonic analysis is an important step towards creating high level representations of tonal music. High level structural relationships form an essential component of music analysis, whose aim is to achieve a deep understanding of how music works. At its most basic level, harmonic analysis requires the partitioning of music into segments along the time dimension, such that the notes in each segment correspond to a musical chord. This *chord recognition* task can often be time consuming and cognitively demanding, hence the utility of computer-based implementations. Reflecting historical trends in artificial intelligence, automatic approaches to harmonic analysis have evolved from purely grammar-based and rule-based systems [6, 15], to systems employing weighted rules and optimization algorithms [8, 11, 13, 14], to data driven approaches based on supervised machine learning (ML) [9, 10]. Due to their requirements for annotated data, ML approaches have also led to the development of music analysis datasets containing a large number of manually annotated harmonic structures, such as the 60 Bach Chorales introduced in [9], and the 27 theme and variations of TAVERN [2].

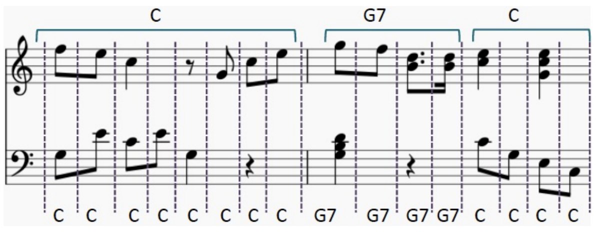


Figure 1. Segment-based recognition (top) vs. event-based recognition (bottom), on measures 11 and 12 from Beethoven WoO68.

A relatively common strategy in ML approaches to chord recognition is to break the musical input into a sequence of short duration spans and then train sequence tagging algorithms such as HMMs to assign a chord label to each span in the sequence (at the bottom in Figure 1). The spans can result from quantization using a fixed musical period such as half a measure [10] or constructed from consecutive note onsets and offsets [9]. Variable-length chord segments are then created by joining consecutive spans labeled with the same chord symbol (at the top in Figure 1). A significant drawback of these short-span tagging approaches is that segments are not known during training and inference, therefore the ML model cannot use features that capture properties of segments that are known to be relevant with respect to their harmonic content. The chordal analysis system of Pardo and Birmingham [8] is an example where the assignment of chords to segments takes into account segment-based features, however it uses a processing pipeline where segmentation is done independently of the subsequent chord labeling.

In this paper, we propose a machine learning approach to chord recognition in which a semi-Markov CRF model is trained to do joint segmentation and labeling of symbolic music. Also called segmental CRFs, this class of models can exploit features that look at all the notes inside a segment. Correspondingly, we define a rich set of features that capture the extent to which the events in an entire segment of music are compatible with a candidate chord label. When evaluated on a dataset of Bach chorales, the semi-CRF approach obtains a 15% error reduction over an event-tagging system. Substantially larger improvements in event-level and segment-level performance are observed on a more difficult corpus of theme and variations by Beethoven and Mozart, thus validating empirically the modeling advantage of joint segmentation and labeling.

2. SEMI-MARKOV CRF MODEL FOR CHORD RECOGNITION

Since harmonic changes may occur only when notes begin or end, we first create a sorted list of all the note onsets and offsets in the input music, i.e. the list of *partition points* [8]. A basic music *event* [9] is then defined as the set of notes sounding in the time interval between two consecutive partition points. Let $\mathbf{s} = \langle s_1, s_2, ..., s_K \rangle$ denote a segmentation of the musical input $\mathbf{x}$, where a segment $s_k = \langle s_k.f, s_k.l \rangle$ is identified by the indices $s_k.f$ and $s_k.l$ of its first and last events, respectively.

Let $\mathbf{y} = \langle y_1, y_2, ..., y_K \rangle$ be the vector of chord labels corresponding to the segmentation $\mathbf{s}$. The set of labels can range from coarse grained labels that indicate only the chord root [14] to fine grained labels that capture mode, inversions, added and missing notes [3], and even chord function [2]. Here we follow the middle ground proposed by Radicioni and Esposito [9] and define a set of labels that encode the chord root (12 pitch classes), the mode (major, minor, diminished), and the added note (none, fourth, sixth, seventh), for a total of 144 different labels. Since the labels do not encode for function, the model does not require knowing the key in which the input was written.

A semi-Markov CRF [12] defines a probability distribution over segmentations and their labels as shown in Equations 1 and 2. Here, the global segmentation feature vector $\mathbf{F}$ decomposes as a sum of local segment feature vectors $\mathbf{f}(s_k, y_k, y_{k-1}, \mathbf{x})$, with label y_0 set to a constant "no chord" value. The ensuing factorization of the distribution enables an efficient computation of the most likely segmentation $\arg\max_{\mathbf{s},\mathbf{y}} P(\mathbf{s}, \mathbf{y}|\mathbf{x}, \mathbf{w})$ using a semi-Markov analogue of the Viterbi algorithm [12].

$$P(\mathbf{s}, \mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x})}}{Z(\mathbf{x})} \quad (1)$$

$$Z(\mathbf{x}) = \sum_{\mathbf{s},\mathbf{y}} e^{\mathbf{w}^T \mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x})}$$

$$\mathbf{F}(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \sum_{k=1}^{K} \mathbf{f}(s_k, y_k, y_{k-1}, \mathbf{x}) \quad (2)$$

Following Muis and Lu [7], for faster inference, we further restrict the local segment features to two types: *segment-label features* $\mathbf{f}(s_k, y_k, \mathbf{x})$ that depend on the segment and its label, and label *transition features* $\mathbf{g}(y_k, y_{k-1}, \mathbf{x})$ that depend on the labels of the current and previous segments. The corresponding probability distribution over segmentations is shown in Equations 3 to 5 below.

Given an arbitrary segment s and a label y, the vector of segment-label features can be written as $\mathbf{f}(s, y, \mathbf{x}) = [f_1(s, y), ..., f_{|\mathbf{f}|}(s, y)]$, where the input $\mathbf{x}$ is left implicit in order to compress the notation. Similarly, given arbitrary labels y and y', the vector of label transition features can be written as $\mathbf{g}(y, y', \mathbf{x}) = [g_1(y, y'), ..., g_{|\mathbf{g}|}(y, y')]$. In Section 3 we describe the set of segment-label features $f_i(s, y)$ and label transition features $g_j(y, y')$ that are used in our semi-CRF chord recognition system.

$$P(\mathbf{s}, \mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{e^{\mathbf{w}^T \mathbf{F}(\mathbf{s},\mathbf{y},\mathbf{x}) + \mathbf{u}^T \mathbf{G}(\mathbf{s},\mathbf{y},\mathbf{x})}}{Z(\mathbf{x})} \quad (3)$$

$$\mathbf{F}(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \sum_{k=1}^{K} \mathbf{f}(s_k, y_k, \mathbf{x}) \quad (4)$$

$$\mathbf{G}(\mathbf{s}, \mathbf{y}, \mathbf{x}) = \sum_{k=1}^{K} \mathbf{g}(y_k, y_{k-1}, \mathbf{x}) \quad (5)$$

3. CHORD RECOGNITION FEATURES

Given a segment s and chord y, we will use the following notation:

- $s.Notes$, $s.N$ = the set of notes in the segment s.
- $s.Events$, $s.E$ = the sequence of events in s.
- $e.len$, $n.len$ = the length of event e or note n, in quarters.
- $e.acc$, $n.acc$ = the accent value of event e or note n, as computed by the `beatStrength()` function in Music21 [1]. This is a value that is determined based on the metrical position of n, e.g. in a song written in a 4/4 time signature, the first beat position would have a value of 1.0, the third beat 0.5, and the second and fourth beats 0.25. Any other eighth note position within a beat would have a value of 0.125, any sixteenth note position strictly within the beat would have a value of 0.0625, and so on.
- $y.root$, $y.third$, and $y.fifth$ = the triad tones of the chord y.
- $y.added$ = the added note of chord y, if y is an added tone chord.

We use the following heuristics to determine whether a note n from a segment s is a figuration note with respect to a candidate chord label y:

1. **Passing**: There are two anchor notes n_1 and n_2 such that: n_1's offset coincides with n's onset; n_2's onset coincides with n's offset; n_1 is one scale step below n and n_2 is one step above n, or n_1 is one step above n and n_2 one step below; n is not longer than either n_1 or n_2; the accent value of n is strictly smaller than the accent value of n_1; at least one of the two anchor notes belongs to segment s; n is non-harmonic with respect to chord y, i.e. n is not equivalent to the root,

[1] http://web.mit.edu/music21

third, fifth, or added note of y; both n_1 and n_2 are harmonic with respect to the segments they belong to.

2. **Neighbor**: There are two anchor notes n_1 and n_2 such that: n_1's offset coincides with n's onset; n_2's onset coincides with n's offset; n_1 and n_2 are both either one step below or one step above n; n is not longer than either n_1 or n_2; the accent value of n is strictly smaller than the accent value of n_1; at least one of the two anchor notes belongs to segment s; n is non-harmonic with respect to chord y; both anchor notes are harmonic with respect to the segments they belong to.

3. **Suspension**: Note n belongs to the first event of segment s. There is an anchor note m in the previous event (last event in the previous segment) such that: m and n have the same pitch; n is either tied with m (i.e. held over) or m's offset coincides with n's onset (i.e. restruck); n is not longer than m; n is non-harmonic with respect to chord y, while m is harmonic with respect to the previous chord.

4. **Anticipation**: Note n belongs to the last event of segment s. There is an anchor note m in the next event (first event in the next segment) such that: n and m have the same pitch; m is either tied with n (i.e. held over) or n's offset coincides with m's onset (i.e. restruck); n is not longer than m; n is non-harmonic with respect to chord y, while m is harmonic relative to all other notes in its event.

Futhermore, because we are using the weak semi-CRF features shown in Equation 4, we need a heuristic to determine whether an anchor note is harmonic whenever the anchor note precedes the current segment. The heuristic simply looks at the other notes in the event containing the anchor note: if the event contains 2 or more other notes, at least 2 of them need to be consonant with the anchor, i.e. intervals of octaves, fifths, thirds, and their inversions; if the event contains just one other note, it has to be consonant with the anchor.

We emphasize that the rules mentioned above for detecting figuration notes are only approximations. We recognize that correctly identifying figuration notes can also depend on subtler stylistic and contextual cues, thus allowing for exceptions to each of these rules.

Equipped with this heuristic definition of figuration notes, we augment the notation as follows:

- $s.Fig(y)$ = the set of notes in s that are figuration with respect to chord y.
- $s.NonFig(y) = s.Notes - s.Fig(y)$ = the set of notes in s that are not figuration with respect to y.

Some of the segment-label features introduced in this section have real values. Given a real-valued feature $f(s, y)$ that takes values in $[0, 1]$, we discretize it into $K+2$ Boolean features by partitioning the $[0, 1]$ interval into a set of K subinterval bins $\mathcal{B} = \{(b_{k-1}, b_k] | 1 \leq k \leq K\}$. For each bin, the corresponding Boolean feature determines whether $f(s, y) \in (b_{k-1}, b_k]$. Additionally, two Boolean features are defined for the boundary cases $f(s, y) = 0$ and $f(s, y) = 1$. For each real-valued feature, unless specified otherwise, we use the bin set $\mathcal{B} = [0, 0.1, ..., 0.9, 1.0]$.

3.1 Segment Purity

The segment purity feature $f_1(s, y)$ computes the fraction of the notes in segment s that are harmonic, i.e. belong to chord y:

$$f_1(s, y) = \frac{\sum\limits_{n \in s.Notes} \mathbf{1}[n \in y]}{|s.Notes|}$$

The duration-weighted version $f_2(s, y)$ of the purity feature weighs each note n by its length $n.len$:

$$f_2(s, y) = \frac{\sum\limits_{n \in s.Notes} \mathbf{1}[n \in y] * n.len}{\sum\limits_{n \in s.Notes} n.len}$$

The accent-weighted version $f_3(s, y)$ of the purity feature weighs each note n by its accent weight $n.acc$:

$$f_3(s, y) = \frac{\sum\limits_{n \in s.Notes} \mathbf{1}[n \in y] * n.acc}{\sum\limits_{n \in s.Notes} n.acc}$$

The 3 real-valued features are discretized using the default bin set $\mathcal{B}$.

3.1.1 Figuration-Controlled Segment Purity

For each segment purity feature, we create a figuration-controlled version that ignores notes that were heuristically detected as figuration, i.e. replace $s.Notes$ with $s.NonFig(y)$ in each feature formula.

3.2 Chord Coverage

The chord coverage features determine which of the chord notes belong to the segment. The first 3 features refer to the triad notes:

$$\begin{aligned} f_4(s, y) &= \mathbf{1}[y.root \in s.Notes] \\ f_5(s, y) &= \mathbf{1}[y.third \in s.Notes] \\ f_6(s, y) &= \mathbf{1}[y.fifth \in s.Notes] \end{aligned}$$

A separate feature determines if the segment contains all the notes in the chord:

$$f_7(s, y) = \prod_{n \in y} \mathbf{1}[n \in s.Notes]$$

A chord may have an added tone $y.added$, such as a 4th, a 6th, or a 7th. If a chord has an added tone, we define two features that determine whether the segment contains the added note:

$$\begin{aligned} f_8(s, y) &= \mathbf{1}[\exists y.added \wedge y.added \in s.Notes] \\ f_9(s, y) &= \mathbf{1}[\exists y.added \wedge y.added \notin s.Notes] \end{aligned}$$

Through the first feature, the system can learn to prefer the added tone version of the chord when the segment contains it, while the second feature enables the system to learn to prefer the triad-only version if no added tone is in the segment. To prevent the system from recognizing added chords too liberally, we add a feature that is triggered whenever the total length of the added note in the segment is greater than the total length of the root:

$$alen(s,y) = \sum_{n \in s.Notes} \mathbf{1}[n = y.added] * n.len$$
$$rlen(s,y) = \sum_{n \in s.Notes} \mathbf{1}[n = y.root] * n.len$$
$$f_{10}(s,y) = \mathbf{1}[\exists y.added] * \mathbf{1}[alen(s,y) > rlen(s,y)]$$

The duration-weighted versions of the chord coverage features weigh each chord tone by its total duration in the segment. For the root, the feature would be computed as shown below:

$$f_{11}(s,y) = \frac{\sum_{n \in s.Notes} \mathbf{1}[n = y.root] * n.len}{\sum_{n \in s.Notes} n.len}$$

Similar features f_{12} and f_{13} are computed for the third and the fifth. The corresponding accent-weighted features f_{14}, f_{15}, and f_{16} are computed in a similar way, by replacing the note duration $n.len$ in the duration-weighted formulas with the note accent value $n.acc$.

The duration-weighted feature for the added tone is computed similarly:

$$f_{17}(s,y) = \frac{\mathbf{1}[\exists y.added] * \sum_{n \in s.Notes} \mathbf{1}[n = y.added] * n.len}{\sum_{n \in s.Notes} n.len}$$

Furthermore, by replacing $n.len$ with $n.acc$, we also obtain the accent-weighted version f_{18}.

An alternative definition of duration-weighted features is based on the proportion of the segment time that is covered by a particular chord note. The corresponding duration-weighted feature for the chord root is shown below:

$$f_{19}(s,y) = \frac{\sum_{e \in s.Events} \mathbf{1}[y.root \in e] * e.len}{\sum_{e \in s.Events} e.len}$$

Similar duration-weighted features normalized by the segment length are defined for thirds, fifths, and added notes.

All duration-weighted and accent-weighted features are discretized using the default bin set $\mathcal{B}$.

3.2.1 *Figuration-Controlled Chord Coverage*

For each chord coverage feature, we create a figuration-controlled version that ignores notes that were heuristically detected as figuration, i.e. replace $s.Notes$ with $s.NonFig(y)$ in each feature formula.

3.3 Bass

The bass note provides the foundation for the harmony of a musical segment. For a correct segment, its bass note often matches the root of its chord label. If the bass note instead matches the chord's third, fifth, or added dissonance, this may indicate that the chord is inverted. Thus, comparing the bass note with the chord tones can provide useful features for determining whether a segment is compatible with a chord label.

There are multiple ways to define the bass note of a segment s. One possible definition is the lowest note of the first event in the segment, i.e. $s.e_1.bass$. Comparing it with the root, third, fifth, and added tones of a chord results in the following features:

$$\begin{aligned}
f_{20}(s,y) &= \mathbf{1}[s.e_1.bass = y.root] \\
f_{21}(s,y) &= \mathbf{1}[s.e_1.bass = y.third] \\
f_{22}(s,y) &= \mathbf{1}[s.e_1.bass = y.fifth] \\
f_{23}(s,y) &= \mathbf{1}[\exists y.added \wedge s.e_1.bass = y.added]
\end{aligned}$$

An alternative definition of the bass note of a segment is the lowest note in the entire segment, i.e. $\min_{e \in s.E} e.bass$. The corresponding features will be:

$$\begin{aligned}
f_{24}(s,y) &= \mathbf{1}[y.root = \min_{e \in s.E} e.bass] \\
f_{25}(s,y) &= \mathbf{1}[y.third = \min_{e \in s.E} e.bass] \\
f_{26}(s,y) &= \mathbf{1}[y.fifth = \min_{e \in s.E} e.bass] \\
f_{27}(s,y) &= \mathbf{1}[\exists y.added \wedge y.added = \min_{e \in s.E} e.bass]
\end{aligned}$$

The duration-weighted version of the bass features weigh each chord tone by the time it is used as the lowest note in each segment event, normalized by the duration of the bass notes in all the events. For the root, the feature is computed as shown below:

$$f_{28}(s,y) = \frac{\sum_{e \in s.Events} \mathbf{1}[e.bass = y.root] * e.len}{\sum_{e \in s.Events} e.len}$$

Similar features f_{29} and f_{30} are computed for the third and the fifth. The duration-weighted feature for the added tone is computed as follows:

$$f_{31}(s,y) = \frac{\mathbf{1}[\exists y.added] * \sum_{e \in s.E} \mathbf{1}[e.bass = y.root] * e.len}{\sum_{e \in s.E} e.len}$$

The corresponding accent-weighted features f_{31}, f_{32}, f_{33}, and f_{34} are computed in a similar way, by replacing the bass duration $e.bass.len$ in the duration-weighted formulas with the note accent value $e.bass.acc$.

All duration-weighted and accent-weighted features are discretized using the default bin set $\mathcal{B}$.

3.3.1 Figuration-Controlled Bass

For each bass feature, we create a figuration-controlled version that ignores event bass notes that were heuristically detected as figuration, i.e. replace $e \in s.Events$ with $e \in s.Events \wedge e.bass \notin s.Fig(y)$ in each feature formula.

3.4 Chord Bigrams

The arrangement of chords in chord progressions is an important component of *harmonic syntax* [1]. A first order semi-Markov CRF model can capture chord sequencing information only through the chord labels y and y' of the current and previous segment. To obtain features that generalize to unseen chord sequences, we follow Radicioni and Esposito [9] and create chord bigram features using only a) the *mode*: major (M), minor (m), or diminished (d); b) the *added* note: none ($\emptyset$), fourth (4), sixth (6), or seventh (7); and c) the interval in semitones between the roots of the two chords.

$$\begin{aligned} g_1(y, y') = \mathbf{1}[&y.mode = \{M, m, d\} \wedge y.added = \{\emptyset, 4, 6, 7\} \wedge \\ &y'.mode = \{M, m, d\} \wedge y'.added = \{\emptyset, 4, 6, 7\} \wedge \\ &|y.root - y'.root| = \{0, 1, ..., 11\}] \end{aligned}$$

Note that $g_1(y, y')$ is a feature template that can generate (3 modes $\times$ 4 added)$^2 \times$ 12 intervals = 1,728 distinct features. To reduce the number of features, we use only the *(mode.added)–(mode.added)'–interval* combinations that appear in the manually annotated chord bigrams from the training data.

4. CHORD RECOGNITION DATASETS

For evaluation, we used two chord recognition datasets:

1. BCHD: this is the Bach Choral Harmony Dataset, a corpus of 60 four-part Bach chorales that contains 5,664 events and 3,090 segments in total [9].

2. TAVERN: this is a corpus of 27 complete sets of theme and variations for piano, composed by Mozart and Beethoven. It consists of 63,876 events and 12,802 segments overall [2].

The BCHD corpus has been annotated by a human expert with chord labels, using the set of labels described in Section 2. Of the 144 possible labels, 102 appear in the dataset and of these only 68 appear 5 times or more. Some of the chord labels used in the manual annotation are enharmonic, e.g. C-sharp major and D-flat major, or D-sharp major and E-flat major. Reliably producing one of two enharmonic chords cannot be expected from a system that is agnostic of the key context. Therefore, we normalize the chord labels and for each mode we define a set of 12 canonical roots, one for each scale degree. When two enharmonic chords are available for a given scale degree, we selected the one with the fewest sharps or flats in the corresponding key signature. Consequently, for the major mode we use the canonical root set {C, Db, D, Eb, F, Gb, G, Ab, A, Bb, B}, whereas for the minor and diminished modes we used the root set {C, C#, D, D#, F, F#, G, G#, A, Bb, B}. Thus, if a chord is manually labeled as C-sharp major, the label is automatically changed to the enharmonic D-flat major. The actual chord notes used in the music are left unchanged. Whether they are spelled with sharps or flats is immaterial, as long as they are enharmonic with the root, third, fifth, or added note of the labeled chord.

The TAVERN dataset [2] currently contains 17 works by Beethoven (181 variations) and 10 by Mozart (100 variations). The themes and variations are divided into a total of 1,060 phrases, 939 in major and 121 in minor. The pieces have two levels of segmentations: chords and phrases. The chords are annotated with Roman numerals, using the Humdrum representation for functional harmony [3]. When finished, each phrase will have annotations from two different experts, with a third expert adjudicating cases of disagreement between the two. After adjudication, a unique annotation of each phrase is created and joined with the note data into a combined file encoded in standard **kern format. However, many pieces do not currently have the second annotation or the adjudicated version. Consequently, we only used the first annotation for each of the 27 sets. Furthermore, since our chord recognition approach is key agnostic, we developed a script that automatically translated the Roman numeral notation into the key-independent canonical set of labels used in BCHD. Because the TAVERN annotation does not mark added fourth or sixth notes, the only added chords that were generated by the translation script were those containing sevenths. This results in a set of 72 possible labels, of which 69 appear in the dataset.

Few other annotated chord recognition datasets exist. One of these is the Kostka-Payne corpus [4], a dataset of 46 brief excerpts compiled by David Temperley from Kostka and Payne's music theory textbook [4]. Several chord recognition systems have used this dataset in the past [8,9]. However, it is smaller than both BCHD and TAVERN, with 3,964 events and only 779 segments. Another dataset is Chris Harte's Beatles collection [3], containing annotations for 12 complete albums. Though this dataset is much larger in size, the chord labels are mapped to audio. We considered re-mapping these labels to MIDI files, but had difficulty finding accurate MIDI files for most Beatles songs.

5. EXPERIMENTAL EVALUATION

We implemented the semi-Markov CRF chord recognition system using a multi-threaded package [5] that has been previously used for noun-phrase chunking of informal text [7]. Following the experimental setting from [9], we evaluated the semi-CRF model on BCHD using 10-fold cross validation: the 60 Bach Chorales were split into 10 folds, and each fold was used as test data, with the other nine folds being used for training. We used the same parti-

[2] https://github.com/jcdevaney/TAVERN
[3] http://www.humdrum.org/Humdrum/representations/harm.rep.html
[4] http://www.cs.northwestern.edu/ pardo/kpcorpus.zip
[5] http://statnlp.org/research/ie/

tion into folds as that employed by Radicioni and Esposito [9], to enable comparison with their perceptron-trained HMM system, henceforth referred to as HMPerceptron. We also used their set of labels, consisting of the 102 chords observed in the dataset, which corresponds to 90 canonical chords. For each feature we computed its frequency of occurrence in the training data, using only the true segment boundaries and their labels. To speedup training and reduce overfitting, we only used features whose counts were at least 5. The performance measures were computed by pooling together the results from the 10 test folds. Table 1 shows the event-level and segment-level performance of the semi-CRF model, together with two versions of the HMPerceptron: HMPerceptron$_1$, for which we do enharmonic normalization both on training and test data, similar to the normalization done for semi-CRF; and HMPerceptron$_2$, which is the original system from [9] that does enharmonic normalization only on test data. When computing the segment-level performance, a predicted segment is considered correct only if both its boundaries and its label match those of the true segment.

System	Acc$_E$	P$_S$	R$_S$	F$_S$
semi-CRF	**83.16%**	**77.60%**	**73.48%**	**75.48%**
HMP-tron$_1$	80.30%	74.18%	69.76%	71.90%
HMP-tron$_2$	80.16%	70.24%	65.78%	67.94%

Table 1. Comparative results on the BCHD dataset, using Event-level accuracy (Acc$_E$) and Segment-level precision (P$_S$), recall (R$_S$), and F-measure (F$_S$).

The semi-CRF model achieves a 3% improvement in accuracy over the original HMPerceptron model, which corresponds to a 15% relative error reduction. The improvement in accuracy over HMPerceptron$_1$ is statistically significant at a p-value of 0.01, using a one-tailed Welch's t-test over the sample of 60 chorale results. The improvement in segment-level performance is even more substantial, with a 7.5% absolute improvement in F-measure over the original HMPerceptron model, and a 3.6% improvement in F-measure over the HMPerceptron$_1$ version, which is statistically significant at a p-value of 0.04, using a one-tailed Welch's t-test.

Error analysis revealed wrong predictions being made on chords that contained dissonances that spanned the duration of the entire segment (e.g. a second above the root of the annotated chord), likely due to an insufficient number of such examples during training. Manual inspection also revealed a non-trivial number of cases in which the authors disagreed with the manually annotated chords, e.g. some chord labels were clear mistakes, as they did not contain any of the notes in the chord. This further illustrates the necessity of building music analysis datasets that are annotated by multiple experts, with adjudication steps akin to the ones followed by TAVERN.

To evaluate on the TAVERN corpus, we created a test dataset from 6 Beethoven sets (*B063*, *B064*, *B065*, *B066*, *B068*, *B069*) and 4 Mozart sets (*K025*, *K179*, *K265*, *K353*). The remaining 11 Beethoven sets and 6 Mozart sets were used for training. All sets were normalized enharmonically before being used for training or testing. Table 2 shows the event-level and segment-level performance of the semi-CRF and HMPerceptron model on the TAVERN dataset. Despite the smaller number of chord labels (69 in TAVERN vs. 90 in BCHD), the results in Tables 1 and 2 show that chord recognition is substantially more difficult in the TAVERN dataset. The comparatively lower performance on TAVERN is likely due to the substantially larger number of figurations and higher rhythmic diversity of the variations compared to the easier, mostly note-for-note texture of the chorales. Error analysis on TAVERN revealed many segments where the first event did not contain the root of the chord. For such segments, HMPerceptron incorrectly assigned chord labels whose root matched the bass of this first event. Since a single wrongly labeled event invalidates the entire segment, this can explain the larger discrepancy between the event-level accuracy and the segment-level performance. In contrast, semi-CRF assigned the correct labels in these cases, likely due to its ability to exploit context through segment-level features, such as the chord root coverage feature f_4 and its duration-weighted version f_{11}.

System	Acc$_E$	P$_S$	R$_S$	F$_S$
semi-CRF	**77.47%**	**66.86%**	**60.35%**	**63.44%**
HMP-tron	60.55%	27.83%	23.21%	25.31%

Table 2. Comparative results on the TAVERN dataset, using Event-level accuracy (Acc$_E$) and Segment-level precision (P$_S$), recall (R$_S$), and F-measure (F$_S$).

6. CONCLUSION AND FUTURE WORK

We presented a semi-Markov CRF approach to chord recognition that does joint segmentation and labeling of tonal music in symbolic form. Compared to event-level tagging approaches based on HMMs or linear CRFs, the segmental CRF approach has the advantage that it can accommodate features that consider all the notes in a candidate segment. This capability was shown to be especially useful for music with complex textures that diverge from the simpler note-for-note structures of the Bach chorales. On the more difficult TAVERN corpus, the semi-CRF substantially outperformed a previous system based on event-level tagging, thus validating empirically the suitability of joint segmentation and labeling for chord recognition.

Manually engineering good features for chord recognition is a cognitively demanding and time consuming process that requires music theoretical knowledge and that is unlikely to lead to optimal sets of features, especially when complex features are involved. In future work we plan to investigate automatic feature extraction using recurrent neural networks (RNN) that preserve the semi-Markov property, such as the recently proposed segmental RNNs [5].

7. ACKNOWLEDGMENTS

We would like to thank Patrick Gray for his help with preprocessing the TAVERN corpus. We also greatly appreciate the prompt help provided by Daniele P. Radicioni and Roberto Esposito. Their detailed responses to our questions and willingness to share the original BCHD dataset and the HMPerceptron code enabled us to conduct a more thorough experimental comparison with their system. Finally, we would like to thank the anonymous reviewers for their insightful comments on the initial draft of this paper.

8. REFERENCES

[1] E. Aldwell, C. Schachter, and A. Cadwallader. *Harmony and Voice Leading*. Schirmer, 4th edition, 2011.

[2] Johanna Devaney, Claire Arthur, Nathaniel Condit-Schultz, and Kirsten Nisula. Theme and variation encodings with roman numerals (TAVERN): A new data set for symbolic music analysis. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[3] Christopher Harte. *Towards Automatic Extraction of Harmony Information from Music Signals*. PhD thesis, Queen Mary University of London, August 2010.

[4] Stefan Kostka and Dorothy Payne. *Tonal Harmony*. McGraw-Hill, 1984.

[5] Liang Lu, Lingpeng Kong, Chris Dyer, Noah A. Smith, and Steve Renals. Segmental recurrent neural networks for end-to-end speech recognition. In *INTERSPEECH*, San Francisco, CA, September 2016.

[6] H. John Maxwell. An expert system for harmonizing analysis of tonal music. In Mira Balaban, Kermal Ebcioğlu, and Otto Laske, editors, *Understanding Music with AI*, pages 334–353. MIT Press, Cambridge, MA, USA, 1992.

[7] Aldrian Obaja Muis and Wei Lu. Weak semi-Markov CRFs for noun phrase chunking in informal text. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 714–719, San Diego, California, June 2016. Association for Computational Linguistics.

[8] Bryan Pardo and William P. Birmingham. Algorithms for chordal analysis. *Computer Music Journal*, 26(2):27–49, July 2002.

[9] Daniele P. Radicioni and Roberto Esposito. BREVE: an HMPerceptron-based chord recognition system. In *Advances in Music Information Retrieval*, pages 143–164. Springer Berlin Heidelberg, 2010.

[10] Christopher Raphael and Josh Stoddard. Harmonic analysis with probabilistic graphical models. In *International Society for Music Information Retrieval Conference (ISMIR)*, 2003.

[11] Thomas Rocher, Matthias Robine, Pierre Hanna, and Robert Strandh. Dynamic chord analysis for symbolic music. In *International Computer Music Conference, ICMC*, 2009.

[12] Sunita Sarawagi and William W. Cohen. Semi-Markov Conditional Random Fields for information extraction. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 1185–1192, Cambridge, MA, USA, 2004. MIT Press.

[13] Ricardo Scholz and Geber Ramalho. Cochonut: Recognizing complex chords from midi guitar sequences. In *International Conference on Music Information Retrieval (ISMIR)*, pages 27–32, Philadelphia, USA, September 14-18 2008.

[14] David Temperley and Daniel Sleator. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23(1):10–27, January 1999.

[15] Terry Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12(1):2–49, 1968.

CONFIDENCE MEASURES AND THEIR APPLICATIONS IN MUSIC LABELLING SYSTEMS BASED ON HIDDEN MARKOV MODELS

Johan Pauwels **Ken O'Hanlon** **György Fazekas** **Mark B. Sandler**
Centre for Digital Music, Queen Mary University of London, UK
{j.pauwels, k.o.ohanlon, g.fazekas, mark.sandler}@qmul.ac.uk

ABSTRACT

Inspired by previous work on confidence measures for tempo estimation in loops, we explore ways to add confidence measures to other music labelling tasks. We start by reflecting on the reasons why the work on loops was successful and argue that it is an example of the ideal scenario in which it is possible to define a confidence measure independently of the estimation algorithm. This requires additional domain knowledge not used by the estimation algorithm, which is rarely available. Therefore we move our focus to defining confidence measures for hidden Markov models, a technique used in multiple music information retrieval systems and beyond. We propose two measures that are oblivious to the specific labelling task, trading off performance for computational requirements. They are experimentally validated by means of a chord estimation task. Finally, we have a look at alternative uses of confidence measures, besides those applications that require a high precision rather than a high recall, such as most query retrievals.

1. INTRODUCTION

Most of the efforts in music information retrieval research are directed towards improving the performance of various automatic labelling tasks. This consists of developing algorithms that are increasingly good at approximating some reference labels, often produced by human annotators, based on an input audio file. These labels represent different musical concepts, such as genre, tempo, instrumentation or musical key.

When such algorithms are deployed in real world scenarios, however, no explicit comparison is made between the generated labels and a reference. An example is the retrieval of audio based on musically meaningful search terms. The only relevant measure of performance here is the degree of satisfaction of the user with the returned audio files. The user will subconsciously verify if the returned audio corresponds somewhat to the query term, and be dissatisfied if it doesn't, but this implicit and informal evaluation is nowhere as rigorous as the numerical evaluation performed to demonstrate algorithmic improvements. This gap between algorithmic evaluation and user evaluation makes that increases in algorithmic performance do not necessary lead to increases in user satisfaction.

Crucially, in many retrieval tasks the precision is more important than the recall. The users only judge the quality of the returned audio, the amount of potentially useful audio files that are not returned to them are unknown and irrelevant (once the amount of returned files reaches a minimally acceptable number of course). A relatively easy way to improve the perceived quality of the returned audio (and thereby user satisfaction) would be to only return those files for which the generated labels are known to be of a high quality. This necessitates a reliable measure of confidence for the generated output labels, which must be calculated without relying on a known reference output.

Despite its obvious use-case, not much work has been performed on confidence measures for music labelling. For tempo estimation in music loops specifically, a thorough study has been recently performed by Font and Serra [5]. They propose a new confidence measure and compare it to earlier efforts of Zapata et al. [18]. In this paper, we devise new methods for confidence estimation that are not specific to a single task, but work with all systems based on hidden Markov models (HMMs). To this end, we start by analysing the reasons why the work on loops was successful and what we can and cannot reuse from it in Section 2. Then we look at the general framework of HMMs, propose some candidate confidence measures and evaluate them for chord estimation in Section 3. Next, a novel application for confidence measures is discussed in Section 4. We end by formulating some conclusions and directions for future work in Section 5.

2. DOMAIN-BASED VERSUS ALGORITHM-BASED CONFIDENCES MEASURES

To aid coming up with confidence measures for a wider range of tasks, it is useful to first reflect on the underlying conditions that made Font and Serra's work [5] successful. They managed to define a confidence measure that can be calculated from just the generated output. It is therefore oblivious to the algorithm that was used to calculate the output. The advantage is that knowledge of and access to the inner workings of the algorithm are not required in order to use the confidence measure. However, this type of confidence measure relies on extra prior knowledge about the application domain, which is used to verify the output against in the absence of internal states of the algorithm and (obviously) the target labels. Therefore we call these confidence measures *domain-specific*, as opposed to *algorithm-specific* measures. For example, the domain knowledge used for loops is that they are cut in such a way that each loop contains an exact number of beats.

It is of the utmost importance that this domain information hasn't been exploited yet by the estimation algorithm. All tested software in [5] fulfils this criterion, as they were developed for music pieces in general, not just loops. If algorithms would already rely on this prior knowledge, the output would be internally adjusted by keeping only those tempo candidates that lead to an exact multiple of beats for the duration of the loop. The confidence measure would then always be maximal and therefore useless.

Finding such unexploited knowledge for a specific application is not always possible, and if there is one, it also needs to be distinctive enough. Take for instance the case of key estimation in loops. A reasonable prior would be to assume that there are no key changes for the duration of a loop. Even if a key estimation algorithm is capable of producing key changes, it is unlikely that multiple keys will be generated over the short duration of a typical loop. No reliable confidence measure can then be derived from this additional information.

We argue that having sufficient unexploited and distinctive domain knowledge for a particular task is a rare event. In practice, it is therefore more likely that we need recourse to algorithm-specific confidence measures. These are defined using the intermediate states of the algorithm, which unfortunately means that separate measures need to be formulated for each algorithm and that the resulting confidence cannot easily be compared between algorithms. The upside is that they are not tied to a particular domain.

In the remainder of this paper, we propose some candidate algorithm-based confidence measures. To mitigate their algorithm-specificity, we will look at the framework of hidden Markov Models [15], which is commonly used in a variety of estimation tasks. Our hope is that the proposed solutions will therefore be task-independent and widely applicable.

3. HIDDEN MARKOV MODEL-BASED CONFIDENCE MEASURES

3.1 Hidden Markov Model Basics

According to Ghahramani [7], "a hidden Markov model (HMM) is a tool for representing probability distributions over sequences of observations." It is widely used to model sequences in applications as diverse as speech recognition [15] and bioinformatics [4]. In music information retrieval, it is commonly used to take temporal dependencies into account when observations are localised.

Formally, an HMM is a doubly stochastic process that consists of a first-order Markov chain of hidden states that can only be observed through another, visible stochastic process. Both processes are sampled at discrete intervals, so they can be represented by an index variable t. This sequence index often represents time. The observed variables can be discrete or continuous, finite or infinite, univariate or multivariate, or any combination thereof, as long as they have a probability distribution associated with them. The state variables, on the other hand, are always discrete and there's a finite number N of them. The values the state variable can possibly take are therefore enumerated as $S_n, \forall 1 \leq n \leq N$. The value of the specific hidden state at index t is represented as Y_t, so $Y_t \in \mathbf{S} = \{S_1, \ldots, S_N\}$. The observed variable at index t is represented as X_t. It can potentially take an uncountable number of values, so we can't enumerate them, only represent their space by $\mathbf{O}$. Furthermore, an observed variable X_t is assumed to depend only on the hidden state Y_t at the corresponding position t in the sequence, not on the hidden or observed variables at any other positions.

Hidden Markov models are entirely described by specifying three probability distributions: (1) the initial state distribution; (2) the state transition distribution, which is time-invariant in a standard HMM; (3) the observation distribution, which is also time-invariant in a standard HMM

$$P\left(Y_1 = S_i\right) \equiv \pi_i, \forall 1 \leq i \leq N \tag{1}$$

$$P\left(Y_{t+1} = S_j | Y_t = S_i\right) \equiv a_{ij}, \forall 1 \leq i, j \leq N \tag{2}$$

$$p\left(X_t = O | Y_t = S_j\right) \equiv b_j\left(O\right), \forall 1 \leq j \leq N, \forall O \in \mathbf{O} \tag{3}$$

The set of parameters of an HMM can therefore be summarised as $\lambda = \{\Pi, A, B\}$, where $\Pi = \{\pi_i\}_i$, $A = \{a_{ij}\}_{i,j}$, $B = \{b_j\left(O\right)\}_{j,\mathbf{O}}$

The context in which confidence measures are useful assumes that the HMM parameters λ are already determined. Given a particular sequence of observations $x^{1:T} = x_1, \ldots, x_T$, we want to determine the underlying state sequence (called path) $y^{1:T} = y_1, \ldots, y_T$ that produced these observations and a value c that indicates how much confidence we have in the generated hidden state sequence. The process that determines the optimal hidden state sequence is called "decoding" the HMM and is well established in the literature [15]. Our goal is to find out which of the internal states of the decoding process could be repurposed as a confidence measure.

The most common way to decode an HMM is to find the single path $\hat{y}^{1:T}$ that is the *maximum a posteriori* (MAP) estimate:

$$\hat{y}^{1:T} = \underset{y^{1:T} \in \mathbf{S}^T}{\operatorname{argmax}}\, p\left(y^{1:T} | x^{1:T}, \lambda\right) \tag{4}$$

$$= \underset{y^{1:T} \in \mathbf{S}^T}{\operatorname{argmax}}\, p\left(y^{1:T}, x^{1:T} | \lambda\right) \tag{5}$$

This path can be found by following the Viterbi-algorithm [17]. For more details about its implementation, we refer to Rabiner's well-known tutorial [15].

3.2 Experimental Setup

In order to experimentally validate the theoretical confidence measures we're about to propose, we need a concrete labelling system based on an HMM. In this section, we describe the chord estimation system that will be used to this end.

Our system first converts an audio file into mono and extracts a time-chroma [6] representation from it that will be used as observations in the HMM. We use two different chroma extractors, such that we can verify that the confidence measure works regardless of features. The first variant we use is recently developed by Korzeniowski et al. [9]. They trained a three layer dense neural network to map quarter-tone log-frequency magnitude spectra to chromas. The second variant is the Compressed Log Pitch (CLP) chroma [12], which applies logarithmic compression before summing the semi-tone log-frequency power spectra into one octave. Both implementations are taken from the *madmom* library [2], version *0.15.1*. The parameters were set to the values proposed in their original papers.

Because the features that are fed into the HMM are chromas, this means the observation space consists of twelve-dimensional positive real numbers $\mathbf{O} = \mathbb{R}^{12}_{\geq 0}$. The observation probability distribution $b_j\left(O\right)$ over $\mathbf{O}$ is calculated using template matching. This requires that each chord state S_j has a template M_j associated with it and a similarity measure that maps observation-template pairs $\left(O, M_j\right)$ to probabilities. We use the normalised cosine similarity, defined as

$$b_j\left(O\right) = \frac{\langle O\,, M_j\rangle}{\|O\|_{L2}\,\|M_j\|_{L2}} \tag{6}$$

where $\langle O, M_j\rangle$ represents the inner product between O and M_j.

In our experiments, we set the number of chord states to 48. We discern four chord types (maj-min-dim-aug triads) for each of the twelve possible roots. The associated chord templates are binary, with the chromas that are theoretically present in the chord set to one and the other chromas set to zero.

The last parts of the HMM that need to be configured are the initialisation and the transition probabilities. Both systems are initialised uniformly, i.e. the probability to start in a specific state is $^1/_{48}$ for all chords. The transition probabilities are kept deliberately simple. The state self-transition probabilities are all assigned the same value

$$a_{ii} = \tau, \forall 1 \leq i \leq N \tag{7}$$

whereas the state-changing probabilities are distributed uniformly

$$a_{ij} = \frac{1-\tau}{N-1}, \forall 1 \leq i, j \leq N, i \neq j \tag{8}$$

This reduces the number of parameters considerably, thereby reducing the potential that our experiments don't translate to other datasets, but it has also been demonstrated that such a simple transition matrix is enough to get most of the benefits of applying an HMM to the observations. In [13], it is shown that the state-changing probabilities in an HMM where states represent relative chords in a key can at most improve the estimation performance by 2–3 %-points, whereas [3] show that this is even less when states represent absolute chords, without the context of a key (as is the case here too). The HMM then effectively acts as a probabilistic temporal smoother, and does not take into account the specific values of surrounding states, only their duration. The only remaining parameter τ is determined through an exhaustive search on the test data.

The score that would ideally be predicted by the confidence measure is calculated by the open-source MusOOEvaluator [1] tool [14]. This is the same software that is used for MIREX. We use the "MirexMajMin" preset for chord evaluation.

Finally, we use two datasets for testing the chord estimation systems and their confidence measures, in order to investigate data-specific behaviour. The first is the "Isophonics" [2] dataset [11]. Specifically, we use the subset that is used for the MIREX chord estimation task. It contains 217 songs and is comprised of 12 Beatles albums (180 songs), a Queen compilation (19 songs) and one Zweieck album (18 songs). The second is the "RWC Popular" [3] dataset [8]. The latter contains 100 Japanese pop songs purposefully recorded for music information retrieval research.

3.3 Sequence Probability as Confidence Measure

As part of the MAP decoding, the probability of the optimal label sequence gets returned:

$$p\left(\hat{y}^{1:T}\right) = \max_{y^{1:T}\in\mathbf{S}^T} p\left(y^{1:T}|x^{1:T}, \lambda\right) \tag{9}$$

$$= \max_{y^{1:T}\in\mathbf{S}^T} p\left(y^{1:T}, x^{1:T}|\lambda\right) \tag{10}$$

Although this seems like an obvious candidate for a confidence measure, as far as we know, nobody has ever examined the correlation between optimal path probability and labelling score. We know from the definition that the optimal path probability has the highest value relative to the probabilities of any other paths, but in order for it to be useful as a confidence measure, its absolute value matters.

Since the joint probability $p\left(y^{1:T}, x^{1:T}\right)$ can be decomposed as

$$p\left(y^{1:T}, x^{1:T}\right) = P\left(y_1\right)p\left(x_1|y_1\right)\prod_{t=2}^{T} P\left(y_t|y_{t-1}\right)p\left(x_t|y_t\right) \tag{11}$$

a first step that needs to be taken is to normalise the path probability with respect to the song duration T, as $p\left(y^{1:T}, x^{1:T}\right)$ gets progressively smaller with T [4]. This

[1] `https://github.com/jpauwels/MusOOEvaluator`

[2] annotations available at `http://isophonics.net/content/reference-annotations`

[3] annotations available at `https://github.com/tmc323/Chord-Annotations`

[4] In practice, we're working in the logarithmic domain precisely to mitigate this vanishing probability problem

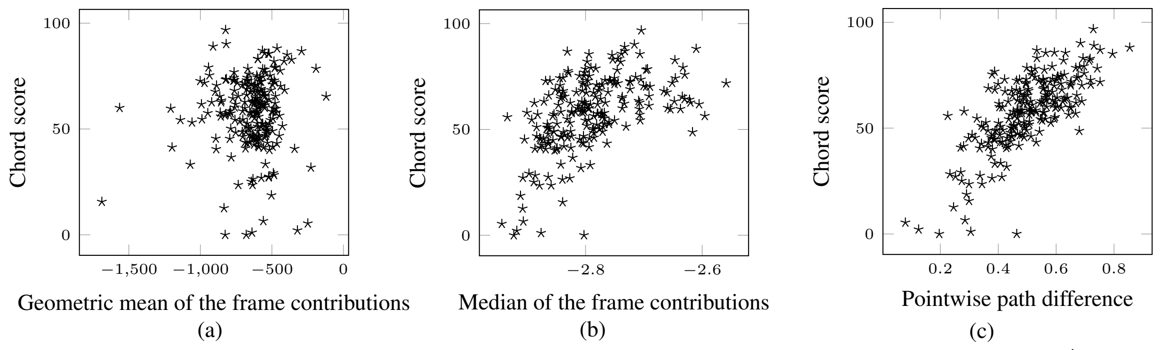


Figure 1: Chord estimation scores for CLP features on the Isophonics dataset: as a function of (a) $\log\left(\sqrt[T]{p\left(\hat{y}^{1:T}\right)}\right)$, (b) $\text{median}_t\left(\log\left[P\left(y_t|y_{t-1}\right)p\left(x_t|y_t\right)\right]\right)$, (c) the pointwise path difference

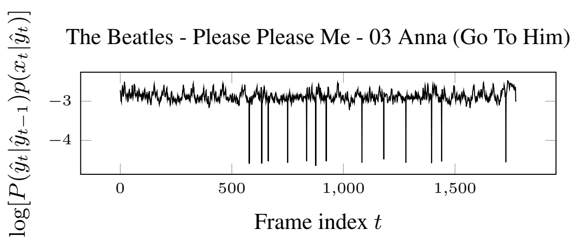


Figure 2: Separate contributions to the optimal path probability $\log\left[P\left(y_t|y_{t-1}\right)p\left(x_t|y_t\right)\right]$ for every frame t

way, we can compare songs of different length. Because the probabilities per frame get accumulated through a product, it makes sense to take the T-th root of the path probability. Then we are effectively calculating the geometric mean of the contributions per frame to the overall path probability.

We've plotted in Figure 1a an example of the scores as a function of the resulting geometric mean for the system with CLP features run on the Isophonics dataset. It is immediately clear from the figure that there is little correlation between the two axes and therefore the optimal path probability is not suitable as a confidence measure.

To find out why this is the case, we take a look at the individual contributions per frame $P\left(\hat{y}_t|\hat{y}_{t-1}\right)p\left(x_t|\hat{y}_t\right), \forall t$ separately. In Figure 2, we show these probabilities for an example song. We can see that there are some strong outliers in these probabilities, and this is the case for every song, not just this example. Because the contributions per frame are multiplied[5] to form the overall probability, we postulate that the limited number of outliers dominate the overall probability regardless of the performance on other frames, such that there is no longer a relation between the overall probability and how well $\hat{y}^{1:T}$ explains $x^{1:T}$. Note that the presence of such an outlier at frame t does not necessarily mean that the global optimal path strays from the locally optimal path at that frame. It could also mean that none of the states can explain that particular observation well $b_j\left(x_t\right) \approx 0, \forall j$. Cases like this are not problematic for the determination of the globally optimal path, since only the difference between observation distributions per state is relevant for the Viterbi algorithm, but the overall probability will be affected.

Since the outliers of $\left(P\left(y_t|y_{t-1}\right)p\left(x_t|y_t\right)\right)$ prove to be so problematic, it makes sense to aggregate the frames differently than through a geometric mean. Instead, we take the median, such that the exact magnitude of the lowest probabilities doesn't matter. We plotted the results in Figure 1b for the same chord estimator configuration as Figure 1a. The figure shows a marked improvement with respect to the geometric mean based confidence measure.

We do believe that it should be possible to achieve a clearer linear relationship between score and confidence measure though. However, as far as repurposing the internal variables of the standard Viterbi algorithm go, we feel we have reached a limit. The advantage of the median-based confidence measure is that it requires practically no more computation time than standard MAP decoding. It only requires the so-called lattice of intermediate path probabilities $p\left(x^{1:t}, y^{1:t}, y_t = S_i|\lambda\right), \forall i, t$ to be kept in its entirety, as opposed to only needing to keep the previous and current frame (t and $t-1$), which obviously leads to an increase in memory. In the next sections, we will compare this measure with a more computationally expensive one and report on more thorough experiments.

3.4 Combining Decoders as Confidence Measure

While the Viterbi algorithm returns the globally optimal label sequence in the MAP sense of the word, the definition of "optimal" is inherently ambiguous. Another criterion of optimality leads to another decoding method. A common alternative to MAP decoding is *pointwise maximum a posteriori (PMAP)* decoding[6]. If we represent the path estimate returned through PMAP decoding from now on as $\tilde{y}^{1:T}$, then we find

$$\tilde{y}_t = \underset{y_t \in \mathbf{S}}{\text{argmax}}\, p\left(y_t|x^{1:T}, \lambda\right) \tag{12}$$

As the name implies, the optimal path is determined point-by-point in such a way that the expected number

[5] The values depicted in Figure 2 are actually summed, because we work in the logarithmic domain.

[6] Confusingly, this decoding algorithm is known under many names, posterior decoding or max-gamma decoding just a few of them. More alternative names can be found in [10, p. 4].

of correct individual states is maximised. The probability to be in a given state s at frame t is found by means of helper forward $\alpha_t(s)$ and backward $\beta_t(s)$ variables obtained through a process known as the forward-backward algorithm [1]:

$$p\left(s|x^{1:T}\right) = \frac{\alpha_t(s)\,\beta_t(s)}{\sum_{r\in\mathbf{S}}\alpha_t(r)\,\beta_t(r)} \tag{13}$$

For further implementation details, we again refer to [15].

It depends strongly on the application whether MAP or PMAP decoding will lead to the best results [7] and to which extent they produce different paths. However, it is known that MAP decoding is most effective when a single path through the HMM strongly dominates all other ones, whereas PMAP decoding gives better results when multiple competing paths have similar overall probabilities [4].

We postulate that when the two methods of decoding yield the same path, this gives a good indication that the path will have a good score. Thus we derive a new confidence measure PPD $\left(y^{1:T}\right)$ based on the pointwise path differences (PPD) as

$$\mathrm{PPD}\left(y^{1:T}\right) = \frac{1}{T}\sum_{t=1}^{T}\delta\left(\hat{y}_t, \tilde{y}_t\right) \tag{14}$$

where δ is the Kronecker-delta, and $\hat{y}_t/\tilde{y}_t$ the MAP/PMAP path estimates at time index t.

To illustrate the PPD, we take once more the same HMM configuration as in Figure 1a and plot the score as a function of the PPD in Figure 1c. Here the relation between the two is more clearly linear. The drawback of this confidence measure is obviously that it requires an additional PMAP decoding step, which requires extra computational power. Some steps, such as the calculation of the observation probabilities, are the same for both decoding algorithms though, so there's potential for reuse. Note that we still return the same MAP path as before, because it gives better results for our HMM configurations than the PMAP path, but the latter is available too.

3.5 Evaluation

As we now have two candidate confidence measures, we can systematically test them on the four combinations of features and datasets. Therefore we perform a similar experiment as in [5]. We start by taking the duration-weighted average score over the complete dataset and then progressively filter the files by first excluding those for which the confidence measure is the lowest. A good confidence measure will then lead to a monotonic increase in score as the filtering threshold increases. The results for the Isophonics dataset and the RWC Popular dataset can be found in Figure 3a, respectively Figure 3b.

In all cases, we observe that the PPD is working well as a confidence measure. The score of the filtered dataset increases monotonically, save for a few exceptions when the number of remaining songs in the dataset becomes so low that the average scores become noisy. The curves of the filtered dataset size as a function of confidence cutoff are also close to straight, which means the PPD is nearly linearly distributed between its extrema. As expected, the median of the per frame contributions to the optimal path is less suitable as a confidence measure. The filtered score initially increases in all situations though, so it can still be used to remove the files with the lowest confidence from the dataset. Doing so will increase the precision when looking for a particular chord sequence in a dataset, for example, at the expense of decreasing the recall. Particularly for the CLP features, the median-based confidence measure seems to work less well. A possible explanation is that the neural network based chromas take context into account. The observations derived from the CLP features are therefore noisier, which will affect the median more.

4. OPTIMAL CHANNEL SELECTION BASED ON CONFIDENCE MEASURE

In this section, we explore an alternative usage for confidence measures. Traditionally, labels in music information retrieval are estimated from mixed down mono audio. Using the mono mix ensures that all information present in the audio is used for the label estimation. For certain types of labels however, it might be beneficial to selectively ignore some of the information. For example, ignoring percussion while estimating chords can be helpful, which has led to percussion separation techniques [16].

If we have multi-channel audio at our disposal, it is therefore possible that analysing a specific channel or combination of channels leads to higher quality labels than when the mono mixdown is used. The problem is then how to determine this (combination of) channel(s). Next, we'll verify if a confidence measure can be used for this.

Ideally, we'd use multi-channel or multi-track audio for this experiment, but since there is no such dataset with annotated chords, we propose an alternative. Starting from stereo Isophonics audio files, we demix the left (L) and right (R) channel according to their panning position into centre (C), hard left (HL) and hard right (HR). We employ the technique used by the "center cut" [8] audio filter of the open-source video editor VirtualDub. It operates in the complex spectral domain and relies on the fact that HL and HR are perpendicular to each other, such that L = C + HL and R = C + HR. In addition to these channels, we calculate the mono (L + R) and sides (HL + HR), such that we end up with seven virtual channels per song.

For each channel, we estimate the chord sequences from CLP features and their confidences. We first aim to determine the theoretical limits of optimal channel selection by performing an oracle-style experiment where we select the channels that lead to the biggest increase and biggest decrease in chord score when compared to the reference mono channel. Then we check how well we can retrieve the optimal channel by selecting the one that returns the chord sequence with the highest confidence.

[7] We tried both and verified that MAP decoding generally gives the best result for our proposed chord estimation system

[8] `http://www.virtualdub.org/blog/pivot/entry.php?id=102`

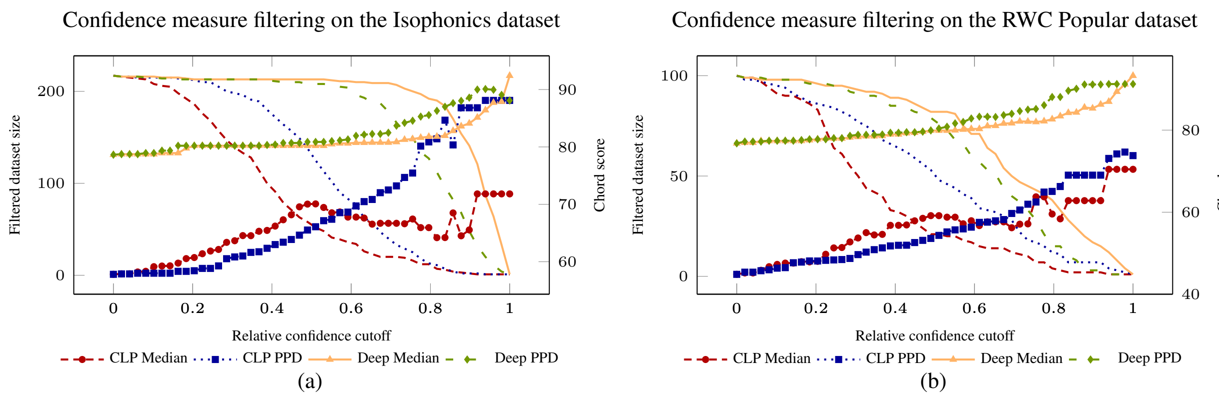


Figure 3: Confidence filtered chord scores for two datasets. The marks indicate the average score over the filtered dataset. The lines represent the number of remaining files in the database as a function of the confidence cutoff.

Album title	Mono score	Oracle best Δ	Oracle worst Δ	Median conf. Δ	PPD conf. Δ
The Beatles - Please Please Me	52.86	6.01	-13.93	-7.23	1.17
The Beatles - With the Beatles	56.22	0.60	-26.81	-18.72	-2.11
The Beatles - A Hard Day's Night	56.21	3.05	-31.91	-21.88	2.24
The Beatles - Beatles for Sale	63.98	9.33	-3.26	5.91	7.33
The Beatles - Help!	52.54	11.82	-13.23	7.90	10.03
The Beatles - Rubber Soul	59.25	5.44	-18.89	-3.00	2.77
The Beatles - Revolver	66.06	5.64	-17.63	-0.70	3.71
The Beatles - Sgt. Pepper's Lonely Hearts Club Band	51.33	4.93	-19.81	-4.83	-1.74
The Beatles - Magical Mystery Tour	66.77	3.52	-18.33	-3.10	0.54
The Beatles - The Beatles (CD1)	62.45	6.32	-17.82	1.76	1.68
The Beatles - The Beatles (CD2)	52.05	6.16	-18.91	1.38	1.67
The Beatles - Abbey Road	63.86	8.39	-17.47	4.33	4.11
The Beatles - Let It Be	61.16	11.96	-8.09	9.24	8.21
Queen - Greatest Hits I	47.50	7.86	-3.54	6.53	6.66
Queen - Greatest Hits II	66.16	4.02	-5.49	-1.45	-1.50
Zweieck - Zwielicht	54.73	7.73	-8.10	5.35	6.26
Overall	57.81	6.60	-14.97	-0.39	3.42

Table 1: Channel selection results grouped per album, using CLP features. The reference mono channel score is reported along with the absolute score differences for the best and worst oracle-style and the confidence-based channel selection.

The channel selection results overall and per album are reported in Table 1. From the oracle experiments we learn that a sizeable improvement in chord score can potentially be achieved by selecting the optimal channel, but also that the consequences of choosing the wrong channel can be severe. The PPD measure can be used successfully to determine a better channel than the mono reference, and manages to get a bit more than half the theoretically maximal improvement. The median-based confidence measure, on the other hand, is not suitable to select the optimal channel.

Based on the individual results per album, no relation with mixing style can be established. The mixing practices range from the mono-like early Beatles albums to the hard-panned late Beatles albums, with more modern Queen and Zweieck in between, but no trends in the (potential) score increase can be identified. Note that when we repeated the experiments with the DeepChroma chord estimation system, the oracle-based maximal increase was barely over 2%-points, and the PPD increase proportionate. A reason might be that the neural network is trained on mono mixes.

5. CONCLUSION AND FUTURE WORK

In this paper, we investigated confidence measures for HMM-based music labelling systems. We formulated two measures, a simple one that doesn't require extra computational power and a better one that is more demanding. They were tested for their ability to filter low-quality output of a chord estimation system. Finally, the capacity of the confidence measures to select the most optimal channel to use for chord estimation has been evaluated.

We hope that the applicability of the proposed confidence measures to other labelling tasks will be verified in the future, by ourselves or by others. To improve the chance of the latter, we've created a new general and modular HMM library [9], usable with C++ and Python, that includes the proposed measures. The code specific to the chord estimation experiments and the presented figures can also be found on-line [10].

Further work will include investigating whether a confidence measure can be used to select the optimal HMM parametrisation. For instance, the self-transition probability τ is currently set to a dataset-wide optimal value, even though it is clearly related to harmonic rhythm and therefore song-dependent. It might be worth investigating if the best value out of a number of candidates can be selected based on confidence.

[9] `https://github.com/jpauwels/Hiddini`
[10] `https://github.com/jpauwels/chord-estimation-confidence`

6. ACKNOWLEDGEMENTS

This work has been partly funded by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/L019981/1 and by the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382.

7. REFERENCES

[1] Leonard E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73(3):360–363, May 1967.

[2] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new python audio and music signal processing library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178. ACM, 2016.

[3] Taemin Cho and Juan P. Bello. On the relative importance of individual components of chord recognition systems. *IEEE Transactions on Audio, Speech and Language Processing*, 22(2):477–492, February 2014.

[4] Piero Fariselli, Pier Luigi Martelli, and Rita Casadio. A new decoding algorithm for hidden markov models improves the prediction of the topology of all-beta membrane proteins. *BMC bioinformatics*, 6(Suppl 4):S12, 2005.

[5] Frederic Font and Xavier Serra. Tempo estimation for music loops and a simple confidence measure. In *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR)*, pages 269–275, 2016.

[6] Takuya Fujishima. Realtime chord recognition of musical sound: a system using Common Lisp Music. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 464–467, 1999.

[7] Zoubin Ghahramani. An introduction to hidden Markov models and Bayesian networks. *International journal of pattern recognition and artificial intelligence*, 15(01):9–42, 2001.

[8] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical and jazz music databases. In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR)*, volume 2, pages 287–288, 2002.

[9] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: the deep chroma extractor. In *Proceedings of the 17th Conference of the International Society for Music Information Retrieval (ISMIR)*, 2016.

[10] Jüri Lember and Alexey A. Koloydenko. Bridging Viterbi and posterior decoding: a generalized risk approach to hidden path inference based on hidden Markov models. *Journal of Machine Learning Research*, 15(1):1–58, 2014.

[11] Matthias Mauch, Chris Cannam, Matthew Davies, Simon Dixon, Chris Harte, Sefki Kolozali, Dan Tidhar, and Mark Sandler. OMRAS2 metadata project 2009. In *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, 2009.

[12] Meinard Müller and Sebastian Ewert. Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR)*, pages 215–220, 2011.

[13] Johan Pauwels and Jean-Pierre Martens. Combining musicological knowledge about chords and keys in a simultaneous chord and local key estimation system. *Journal of New Music Research*, 43(3):318–330, 2014.

[14] Johan Pauwels and Geoffroy Peeters. Evaluating automatically estimated chord sequences. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[15] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[16] Yushi Ueda, Yuki Uchiyama, Takuya Nishimoto, Nobutaka Ono, and Shigeki Sagayama. HMM-based approach for automatic chord detection using refined acoustic features. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5518–5521, March 2010.

[17] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[18] Jose Ricardo Zapata, Matthew E. Davies, Andre Holzapfel, Joao L. Oliveira, and Fabien Gouyon. Assigning a confidence threshold on automatic beat annotation in large datasets. In *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR)*, pages 157–162, 2012.

CONVOLUTIONAL NEURAL NETWORKS FOR REAL-TIME BEAT TRACKING: A DANCING ROBOT APPLICATION

Aggelos Gkiokas
Institute for Language and Speech Processing,
Athena Research and Innovation Center,
Athens, Greece
agkiokas@ilsp.gr

Vassilis Katsouros
Institute for Language and Speech Processing,
Athena Research and Innovation Center,
Athens, Greece
vsk@ilsp.gr

ABSTRACT

In this paper a novel approach that adopts Convolutional Neural Networks (CNN) for the Beat Tracking task is proposed. The proposed architecture involves 2 convolutional layers with the CNN filter dimensions corresponding to time and band frequencies, in order to learn a Beat Activation Function (BAF) from a time-frequency representation. The output of each convolutional layer is computed only over the past values of the previous layer, to enable the computation of the BAF in an online fashion. The output of the CNN is post-processed by a dynamic programming algorithm in combination with a bank of resonators for calculating the salient rhythmic periodicities. The proposed method has been designed to be computational efficient in order to be embedded on a dancing NAO robot application, where the dance moves of the choreography are synchronized with the beat tracking output. The proposed system was submitted to the Signal Processing Cup Challenge 2017 and ranked among the top third algorithms.

1. INTRODUCTION

Rhythm analysis has been one the most important tasks in the Music Information Retrieval (MIR) community. The metrical structure of a music piece, comprising the metrical levels, their relations, the beats and the beat rate (i.e. tempo), fully describe its rhythm content as proposed in the Generative Theory of Tonal Music [1]. The exact beat locations are essential information of a music excerpt which describe various aspects of it, as for example tempo variations, expressive timing, temporal grouping of weak and strong accents within the meter etc.

Automated beat estimation, usually found under the term of Beat Tracking, has been one of the fundamental tasks in the MIR field. Beyond the importance of the information that can be found in the beat locations for a music excerpt as mentioned before, the automatic estimation of beats is important because it can serve as an intermediate step to tackle other MIR tasks, such as chord change detection [2], chord detection [3], the computation of beat-synchronous features for identifying cover songs [4], tempo estimation [5] and downbeat detection [6, 7] to name a few.

A variety of different approaches can be found in the literature to tackle the beat tracking task. In an early work, Scheirer [8] deployed comb resonators on spectral energies to for a tempo and beat estimation system, where the response of the resonators was interpreted as the beat positions. Dixon presented BeatRoot [9], a beat tracking software based on inter-onset interval (IOI) clustering for tempo induction and multiple agents for finding the beats. In [10], the authors extended BeatRoot to a real-time beat tracking software named IBT. In [11] a dynamic programming formulation for the beat-tracking task is presented. In a similar manner in [12] the output of a tempo estimation method was incorporated in a dynamic programming based beat-tracking method. In [13] the authors propose a two-state probabilistic model to handle discontinuities in beats caused by switching metrical levels. Peeters and Papadopoulos [14] propose a probabilistic framework, where beat positions are considered as latent variables in order to extract downbeats and beats. In [15] a unified probabilistic framework in the context of rhythm analysis is proposed, consisting of a time-invariant Bayesian network for modeling the relations of tactus, tatum and meter and the beat locations.

More recent works incorporated Neural Networks (NN) for handling the Beat-Tracking task. A remarkable work was firstly presented in [16], where an onset detection method which was based on Bidirectional Long Short-Term Memory (BLSTM) Neural Networks was adapted to a beat tracking system. The performance of this method outperformed the state-of-the-art. In [17], under the assumption that humans perceive the rhythm in a relative manner with respect to the salient periodicities of a music excerpt, a cepstroid invariant neural network is proposed to estimate the beat positions.

Although Convolutional Neural Networks (CNN) have been successful in many MIR applications such as onset detection [18], structure analysis [19], chord recognition [20] and genre classification [21, 22], to the best of our knowledge there are no works that deploy CNNs for the beat tracking task. CNNs have used recently for downbeat tracking as in [23] and [24]. However, these methods apply the CNN on larger segments of beat synchronous features, i.e. a beat-tracking step (based on another technique) is preceded. In a very recent approach [25] a dance

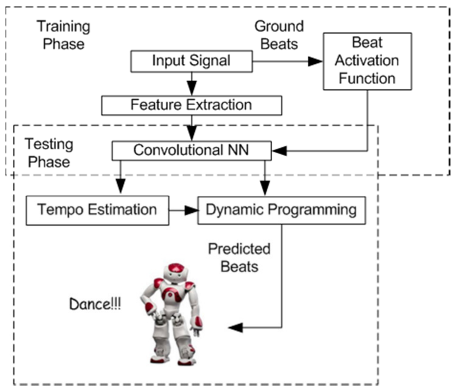


Figure 1. Overview of the proposed method.

genre classifier that is based on CNNs that model temporal features is proposed.

In this paper we propose a novel approach that adopts CNNs to handle the beat-tracking task. The CNN are used to learn a Beat Activation Function (BAF) from a time-frequency input representation. The resulting BAF is subsequently used to infer the beats with a dynamic programming approach. The proposed method was embedded on a dancing NAO[1] robot application. Dancing robots [26] have gained some interest in both robotics and MIR scientific communities and is an essential application for human-robot interaction and entertainment [27]. In [28] the authors focused on the motion of a humanoid dancing robot without any music rhythm analysis. In [29] a simple beat-tracking system was embedded on the RoboNova and Hubo robots. In [30] a singing robot that synchronizes its singing with the estimated beats is presented. In [31] the authors focused on eliminating the ego-motion and beat-synchronous noise caused by a real-time dancing robot.

The rest of the paper is organized as follows. Section 2 will provide a brief overview of the proposed method. The algorithmic details of the beat-tracking method will be presented in Section 3. Technical and implementation details of the dancing robot application are discussed in Section 4. Section 5 is dedicated on the evaluation of the proposed method. Section 6 concludes this paper with discussion and future work directions.

2. METHOD OVERVIEW

An overview of the proposed method is shown in Figure 1. From the input signal the time-varying features that capture the salient musical events are firstly computed. These features along with the Beat Activation Function derived from the ground truth data are used to train a Convolutional Neural Network (CNN). In the testing phase, the output of the CNN is interpreted as a prior probability of a time instant corresponding to a beat. The CNN output is processed further by a filter-bank of comb

[1] https://www.ald.softbankrobotics.com/en/cool-robots/nao

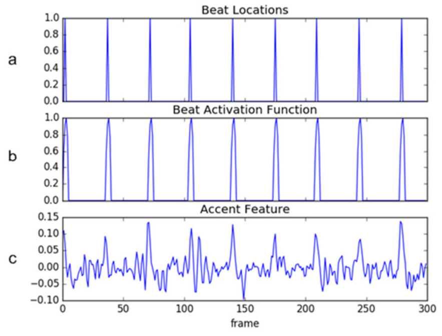


Figure 2. Beat Locations, Beat Activation Function and Accent Features.

resonators to capture the salient periodicities and estimate the music tempo, which along with the BAF are processed by a Dynamic Programming Beat Estimation module. The aim of this module is to find a sequence of beats that are rhythmically consistent, and are dominant on the BAF. The beat tracking algorithm is embedded on a NAO robot. The robot reproduces an audio waveform, while in real-time it computes the beat locations. Consequently, the NAO robot synchronizes its dancing movements of the choreography to the predicted beats.

3. METHOD DETAILS

3.1 Feature Extraction

A conventional front-end schema is used for feature extraction. The input music signal is firstly downsampled to 16 kHz. Downsampling is required in order to conform with NAO's audio recording capabilities, since NAO can only record at 16 and 48 kHz[2]. At next, the amplitude spectrogram denoted by **X** is calculated, using a sliding window of 1024 samples shifted at 160 samples (100 Hz frame rate). Each frame is then processed by a mel-filterbank of M bands to derive the mel-band amplitudes **S.** Next, the time difference of the logarithm of the amplitudes **S** of consecutive frames is computed. Finally, the output features are half-wave rectified to derive the *accent features* **A**.

3.2 Calculating a Beat Activation Function

A Beat Activation Function (BAF) is a function of time that represents the salience or the probability of time instants being beats. Figure 2 (a) shows the beat locations of a music excerpt, Figure 2 (b) shows the corresponding BAF, while in Figure 2 (c) the input features of the corresponding music excerpt are shown. BAF is a smoother version of pulses at the beat locations and is derived by using a Gaussian curve around the beat locations. The aim of the Gaussian blurring is to decrease the sensitivity to less accurate beat annotations. The standard deviation

[2] Although the current implementation does not use the microphone, the use of the recording sampling rate was chosen for reasons of uniformity.

of each Gaussian is set to be inversely proportional to the beat period.

In the proposed method a Convolutional Neural Network is deployed to learn the BAF denoted by $b[n]$ from the accent features $a[n]$. The intuition for this choice is that a CNN can learn temporal patterns of the input feature space and map these features to the BAF space. An example of how a simple Convolutional Unit can learn a rhythmic pattern and outputs a BAF is shown in Figure 3. Figure 3a shows a pattern r_1 of length L and Figure 3b shows a rhythmic repetitive pattern of r_1 , $r = [r_1, r_1, r_1, \ldots r_1]$. We assume that beat positions should occur on the positive peak the r_1. If r is convolved with $\tilde{r}_1[n] = r_1[L-n]$ the resulting vector will exhibit peaks with the same period as r, but in different positions (Figure 3c). However, if r (Figure 3e) is convolved with filter $\tilde{r}_2$ (Figure 3d) which is a circular shift of r_1, then the resulting output (Figure 3f) is in phase, i.e. exhibits peaks at the same locations, with r. This simple example indicates that the CNNs are capable of producing a BAF that is synchronous with its input in a *causal* manner. If a sequence a of length N is convolved with a filter h of length L, the resulting sequence of length N-L+1 can be *synchronous* with a target BAF cropped by L-1 samples from the beginning. In this way, we can compute a BAF in a *causal* manner, i.e. $b[n]$ is derived only from current and past samples of r and r_1:

$$b[n] = \sum_{i=0}^{L-1} \tilde{h}[n-i]a[n-i] \tag{1}$$

where $\tilde{h}[n] = h[L-n]$.

The CNN architecture of the proposed method is shown in Figure 4. It consists of two convolutional layers and three dense layers. No max-pooling or other pooling step is applied anywhere, since this would reduce the frame rate, which is critical in the context of real-time beat tracking. The first convolutional layer consisting of N_1 filters of L_1 length is applied on the accent features **A**. This results to temporal sequences of dimension $N_1 \times M$. Next, the dimension is reduced to dimensions of M and 1 by applying the $N_1 \times 1$ and $M \times 1$ input-output feed forward layers respectively. As a non-linearity function of the CNN, the logistic function is applied to the output of each layer. The above process results to a 1-dimensional temporal sequence of length $N - L_1 + 1$ where N is the length of the input accent features **A**. The output $o_1[n]$ of this subnet is further processed by the 2nd convolutional layer with N_2 filters of L_2 length. The resulting temporal $o_2[n,m]$ sequence of dimension N_2 is then reduced to a single-dimensional sequence $o[n]$ by applying a $N_2 \times 1$ feed forward layer. As before, the logistic function is deployed at each layer. $o[n]$ is considered as the output of the network. For training the net work, the binary cross-entropy is used as the cost function.

After the calculation of the BAF, the beat tracking problem can be seen as a peak selection step, which comprises of two steps. At first a dominant tempo is estimated and then the most salient peaks of the BAF that are rhythmically consistent with that tempo are selected. These two steps are presented in detail in the Sections 3.3-3.5.

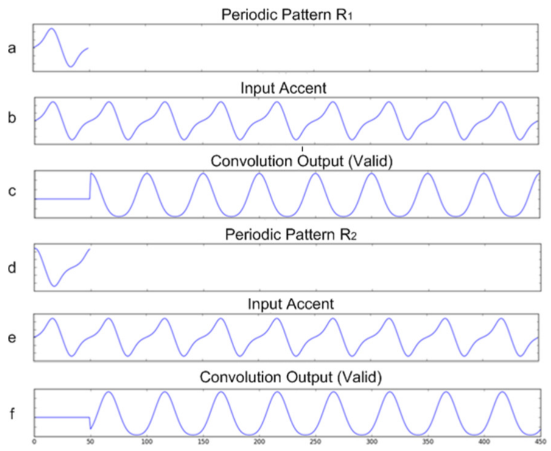


Figure 3. Convolution of periodic sequences with its rhythm elements.

3.3 Tempo Estimation

The next important component of the beat-tracking method is the estimation of the music tempo. A central notion on rhythm analysis is the Periodicity Function (PF) [32] or Periodicity Vector, which is a function or vector that represents the salience of the rhythmic frequencies. In this approach, we compute a PF by processing the BAF by a bank of oscillators, each of which oscillates at a period τ. The output $o_\tau[n]$ of the oscillator with period τ and input $a[n]$ is chosen as:

$$o_\tau[n] = \beta \cdot o_\tau[n-\tau] + (1-\beta) \cdot a[n] \, . \tag{2}$$

The PF for period τ at the frame n is given by the maximum value of o_τ within the past period, i.e.,

$$P[\tau][n] = \max\left\{o_\tau[k], \ \ k = n-\tau \ldots n\right\} . \tag{3}$$

The PF is not calculated for every frame but every T_N frames and for periods in the range $\tau \in [\tau_{min}, \tau_{max}]$. In order to estimate a more reliable PF and cope with slight PF variations, the PFs are averaged for the last K values to get an smoothed PF $\tilde{P}$. The tempo period is then calculated as the maximum value of $\tilde{P}$.

3.4 Beat Tracking as Dynamic Programming

The Beat Estimation method is a modification and adaptation of the method presented in [12]. It is a dynamic programming method that finds an optimal path of a beat sequence that maximizes a cost function. Let $\{b_i\}_i$ denote the candidate beat positions, which are the positive peaks of the BAF. The first part of the beat tracking algorithm is to define a beat similarity of two candidate beats $b_i < b_j$ as

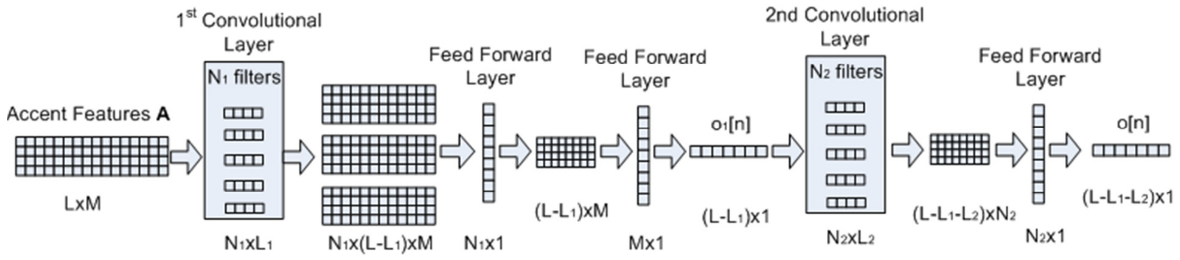


Figure 4. The Convolution Neural Network Architecture.

$$d(b_i,b_j) = c \cdot d_T(b_i,b_j) + (1-c)o[b_j] + d_0, \quad (4)$$

where

$$d_T(b_i,b_j) = \exp\{-\frac{1}{\sigma^2}\ln^2\left((b_j - b_i)/\tau\right)\}, \quad (5)$$

with τ in (5) being the tempo period and $d(b_i, b_j)$ being the weighted sum of two terms plus the bias d_0. The first term in (4) $d_T(b_i, b_j)$ indicates if the beats b_i, b_j are rhythmically consistent, and increases as the distance of b_i, b_j ap proaches the tempo period τ. The second term is related to the beat salience, and is equal to the value of the BAF. Finally, the term d_0 can be seen as a *beat insertion* bias. Given a sequence $\{b_i\}_i$ of possible beats, a beat activation function $o[n]$ and a tempo period τ one can find the optimal path $\{b^*{}_i\}$ that maximizes the objective function

$$O(\{b_l^*, l \in L\}) = \sum_{l \in L} d(b_{l-1}^*, b_l^*) \quad (6)$$

3.5 Real-Time Formulation of Beat Tracking

The beat tracking model described in the previous section is not a straightforward online algorithm and has to be adapted to meet the real-time requirements. For setting the real-time formulation, we define four frame sets. Let us denote with n_0 the current frame. Then we define by $F = \{b_i\}_i$ the beat candidates (peaks of BAF) before n_0, by $B = \{b_i{}^*\}_{n_0}$ the optimal path of beats, by $R = \{r_i{}^*\}_{n_0}$ the output of the real-time beat tracking before n_0 and by $E = \{e_i\}_{n_0}$ the expected beats of the n_0 frame. Note that $\{b_i{}^*\}_{n_0} \neq \{r_i{}^*\}_{n_0}$, since B can be considered as the non-causal output before n_0 , while the set R is the causal output before n_0. A graphical illustration is shown in Figure 5. When a new peak at frame n_1 is detected, then the hypotheses $\{b_i{}^*\}_{n_1}, \{r_i{}^*\}_{n_1}$, and $\{e_i\}_{n_1}$ are updated as follows. Firstly, the optimal path of beats is recalculated $\{b_i{}^*\}_{n_1}$. Then, $E = \{e_i\}_{n_1}$ is updated by adding a new expected beat $\tilde{e}$, which is the last beat $\tilde{b}$ of $B = \{b_i{}^*\}_{n_1}$ plus the tempo period τ. Moreover, the expected beats close to $\tilde{e}$ (based to a threshold) are removed. Next, if the current peak n_1 is close enough (based to the same threshold) to an expected beat (added to $E = \{e_i\}_{n_1}$ on a past frame), the n_1 is considered as a beat, it is added to $R = \{r_i{}^*\}_{n_1}$ and the corresponding expected beat is removed from $E = \{e_i\}_{n_1}$. In other words, a peak is instantly classified as a beat, if it is close enough to an expected

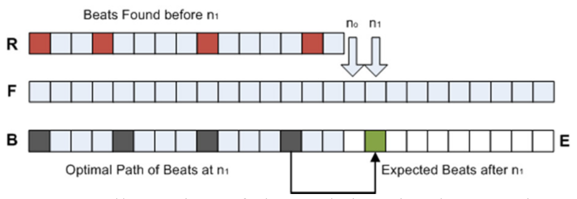


Figure 5. Illustration of the real-time implementation of the beat-tracking method

beat. Based on this formulation, the algorithm is almost online, with a latency of one frame, which is needed to decide whether a frame is peak or not. The algorithm can be summarized as follows:

```
1)  Initialize: $F = \{\}, E = \{\}, B = \{\}, R = \{\}$
2)  Get new frame $n$.
3)  Compute the BAF $o[n]$
4)  If $n$ is not peak GOTO 2.
5)  Compute distance of $n$ to previous peaks.
6)  Get optimal beat sequence $B = \{b_i{}^*\}_n$ before $n$.
7)  Add $\tilde{e} = \tilde{b} + \tau$ ($\tilde{b}$ is last beat of $\{b_i{}^*\}_n$) to $E$.
8)  Remove elements of $E$ very close to $\tilde{e}$.
9)  Get last element $\tilde{r}$ of $R$ (if any).
10) For each $e$ in $E$:
        a.  if ($n$ close to $e$) and ($n - \tilde{r} > \tau/2$)
                i.  add $e$ to $R$.
               ii.  remove $e$ from $E$.
              iii.  This_frame_is_beat = True
11) GOTO 2.
```

The statement x close to y is defined to be true if $|x - y| < 4$ frames. The second condition $n - \tilde{r} > \tau/2$ in the statement 10a ensures that once a past frame is classified as beat, the frames that are close to this frame should not be classified as beats.

4. THE DANCING ROBOT APPLICATION

4.1 The NAO Robot

For deploying the real-time beat tracking method to a dancing robot, a NAO Robot v4 was used as the target hardware. It runs on an Intel Atom Z530 CPU with 2 cores at 1600Mhz, with 1GB RAM. The Operating System is the Linux based NAOqi, version 2.1.2. The NAO's kinematics include 25 motors; 2 motors for controlling the head, 4 motors for each arm, 2 motors for each hand, 5 motors for each leg, plus 1 motor for controlling both hip's yaw pitch (Figure 6). The pose (or "state") of the robot can be uniquely described by the state of the 25 motors, which are used to define NAO's dancing movements that are synchronous with the real-time beat tracking re-

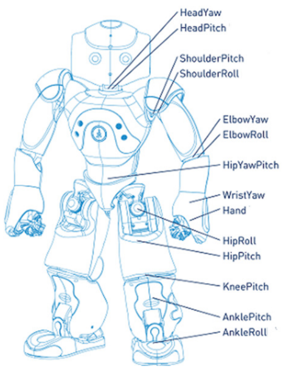


Figure 6. NAO's motor map.

sults. Although NAOqi has built-in functions for controlling the robot's stability, steep movements should be still avoided. Apart from the robot's movements, NAO's eyes are used to show the beat-times that are calculated in real-time. Each eye consists of 8 leds. The color of NAO's eyes is controlled to change on every beat.

4.2 The Choreography Model

NAO's dancing movements have been designed to be easily parameterized. We refer to this parameterization as the "Choreography Model (CM)". The CM consists of a set of poses $p = \{p_1, p_2, \ldots, p_K\}$ and a $K \times K$ transition matrix **T** of these poses. Each pose p_i is represented by the 25 values that correspond to the state of each of the motors mentioned above. The (i, j) element of matrix **T** denotes the probability of moving to pose p_j given that NAO's previous pose is p_i. Thus, each row of **T** sums to one. In this way one can define different choreographies. The elements of **T** can be binary values, thus defining a deterministic sequence of positions, or can take values in the interval (0,1) defining a stochastic sequence, or mixed, i.e. certain groups of deterministic sequences can be mixed stochastically. The CM allows an easy way to define an arbitrary choreography, mixing different choreographies (i.e. reusing p_i) and change the choreography in real-time, as for example in the case where the choreography is linked to a genre classifier. Although the CM poses contain information for the legs, for reasons of simplicity and to maintain the robot's stability, the movement of the legs of the robot are overridden by another simpler CM that considers only leg movements and involves only two poses, which realize the slight hip (or knee) movement of the robot that can be seen in the demonstration videos.

4.3 Robot Dancing in Practice

Since the robot cannot instantly move to the next pose when a beat is found, somehow the next beat-time has to be inferred beforehand. When a beat is found in real-time, then the next beat time is inferred by adding to the current beat time the tempo period (similarly to the expected beat notion of previous Section). Therefore, the predicted next beat that determines the robot's movement, slightly differs from the actual prediction of the real-time beat tracker. However, although such differences might be audible, they are not visible, i.e. they are not evident by human vision, as it can be seen in the demonstration video. Let's consider the case that a beat is found by the real-time algorithm at time instant 0.0 sec with a tempo at 60 BPM (i.e. period of 1 sec). Then the next beat will be inferred to be at 1.0 secs, and therefore the robot will start moving from its current pose to the next at 1.0 sec of a specific choreography. If the beat of the actual real-time algorithm is found a few msecs later of the inferred one (e.g. 50 msecs), this difference will not be visible. Moreover, the robot will adapt its movement in order to complete its next move at 2.10 secs. This can be seen in the demonstration video, since robot's movements look natural and synchronous with the music despite the small variations from the true beat.

However, in order to demonstrate the actual capability of the algorithm and make it clearly visible, apart from dance movement we incorporated an instant change in the color of the eyes when a beat is found. The colors used can also be parameterized as it is shown for the music excerpts in the demonstration video. Moreover, due to motor limitations of the robot, the robot dances on half time, i.e. the movement is planned for every second subsequent beat. Thus, every two beats (two color changes of the eyes) the robot completes one movement of the choreography.

5. EVALUATION

5.1 Algorithm Parameters and Implementation Details

In this section some details of the implementation and parameters of the proposed method will be provided. Regarding the accent features (Eq. 1), the number of bands of the mel-filterbank was set to *M*=8, and the frame rate was set to 100 Hz (see Section 3.1). For the tempo estimation phase, the corresponding periods of the tempo analysis range (Eq. 4) were set to $\tau_{min} = 350, \tau_{max} =$ 700 ms. The size of the CNN is set to $N_1 = 50$, $N_2 =$ 100, $L_1 = 50$ and $L_2 = 200$. The whole implementation is written in Python 2.7. The training functions were written using the Lasagne/Theano[1] libraries and run on a Tesla K40 GPU. The Binary Cross Entropy was used as the loss function of the network, which was trained using Nesterov momentum of 0.9 and a variable learning rate. Due to the robot's software limitations, for the embedded algorithm only the Numpy library was used for calculating dot products. The algorithm runs at ~ 100% CPU single core on the NAO robot and ~3 % CPU single core on an Intel i7.

5.2 Beat-Tracking Performance

The proposed method was submitted to the Real-Time Beat Tracking Challenge of the IEEE Signal Processing

[1] http://lasagne.readthedocs.io/en/latest/

Cup 2017 [33]. It was a prerequisite that the submissions should have been implemented around creative application of a real-time beat tracking algorithm that is embedded on a device. The challenge was run in three phases. Initially, the teams were provided two datasets, an *open* dataset consisting of 25 excerpts that were annotated by the organizers, and a *closed* dataset, where the annotations were not released to the participating teams. In the first phase each team had to submit annotations for three annotated excerpts of their choice (other than the *open* and *closed* datasets). One of these three pieces was selected by each team as a *challenge* piece. The challenge pieces for all teams formed the *challenge* dataset, and the dataset consisting of the other two pieces submitted by each team, formed the *team* datasets. In the second phase each team had to submit the results of their algorithm for the *closed* and *challenge* datasets, while they were provided annotation for the *team* dataset to be used for training or fine tuning. The submissions were evaluated based on three criteria, the annotation quality, the beat-tracking performance and the novelty of the application. The beat-tracking performance was measured by a metric based upon the standard AMLt, but instead of the arbitrary inclusion of double, half-time or off-beat tapping, the "allowed" metrical levels were specified on an excerpt-by-excerpt basis. In this way, the evaluation could cope with excerpts in odd meters in a more robust manner. After the 2nd phase, the three best teams were selected to participate in the final phase of the competition at the ICASSP 2017 conference.

The Convolutional Neural Network of the proposed method was trained on three datasets, the GTZAN dataset [34], the Ballroom Dataset [35] and the SMC Mirex Dataset [36]. The parameters of Eq. (4), and (5) were tuned with a grid search on the *open* and the *team* submitted datasets which were used as validation set. The proposed method achieved a beat-tracking performance of 63.3% and 64.2% on the *closed* and *challenge* datasets respectively, based on the modified AMLt metric and was ranked 6th among 21 submissions regarding the beat-tracking performance. Figure 7 presents the comparative beat tracking results for all submissions. Due to the organizers choice, the comparative results are provided anonymously. The video demonstration of the robot dancing for two excerpts can be downloaded from[1].

6. CONCLUSION AND FUTURE WORK

In this paper a novel method that adopts CNNs for the beat-tracking problem is presented. CNN are proved to be capable to function in an online manner, i.e. the output of the CNN depends only on current and past values of its input, thus allowing a real-time implementation. The proposed method was designed to be embedded on a NAO robot, and its parameters were optimized to meet the real-time requirements rather than to optimize beat-tracking

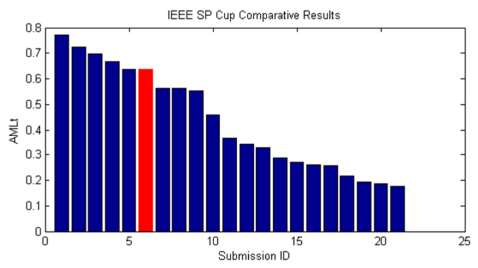


Figure 7. Comparative results on the IEEE SP Cup Challenge. The submissions are ordered by their beat tracking performance.

performance in general. The proposed method was submitted in the IEEE 2017 Signal Processing Cup Beat Tracking Challenge and was ranked 6th among 21 algorithms.

There are a number of challenges and future work imposed by the proposed method. Regarding the use of CNNs, a further investigation and experimentation of various network parameters, such as the network architecture, the number and size of layers, the use of other than the sigmoid non-linearity functions, more relevant to the beat-tracking problem cost functions, or even the smoothing procedure of the target BAF. Moreover, CNNs can be combined with other Neural Network types such as Recurrent Neural Networks, as for example in a setting that CNN will act as a feature preprocessing step to extract a smooth BAF, which will be further processed by an RNN.

Apart from the CNN, other aspects can be further elaborated to increase the performance of the beat-tracking algorithm. At first, the tempo estimation method may be improved, by considering more complex oscillators or other alternative Periodicity Analysis methods that can be found in the literature. Moreover, further processing of the PF can be incorporated to the method, reducing the so called “octave errors”. Regarding the beat tracking, it can be further improved to more sophisticated approaches, as for example with the use of agents [10] allowing smarter selection from the candidate beats.

Regarding the robot itself, we plan to incorporate a genre classifier. This will allow the robot to change choreographies on the fly, with respect to the music being played. Finally, it will be an important extension of the proposed method to handle audio streams recorded from the robot’s microphones instead of audio files. This would require either the training of the CNN to be made with data recorded from microphone, or by deploying denoising techniques.

7. ACKNOWLEDGEMENTS

This work is part of the iMuSciCA project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731861. Students Myrto Damianou, Stelios Stavroulakis, Giannis Karamanolakis and Christos Palivos, participated in the SP Cup team. Special thanks

[1] http://mir.ilsp.gr/dance_robot.html

to Matthew Davies for providing the evaluation SP Cup results and for his helpful feedback.

8. REFERENCES

[1] F. Lerdahl, and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1985.

[2] Goto, Masataka, and Yoichi Muraoka. "Real-time Beat Tracking for Drumless Audio Signals: Chord Change Detection for Musical Decisions." *Speech Communication* 27, no. 3 (1999): 311-335.

[3] Zenz V, Rauber A. "Automatic Chord Detection Incorporating Beat and Key Detection," *in Proc. ICSPC* 2007.

[4] Ellis DP, Poliner GE,, "Identifying Cover Songs with Chroma Features and Dynamic Programming Beat Tracking," *in Proc. ICASSP,* 2007.

[5] Böck S, Krebs F, Widmer G., "Accurate Tempo Estimation Based on Recurrent Neural Networks and Resonating Comb Filters," in *Proc. ISMIR,* 2015.

[6] Durand S, Bello JP, David B, Richard G. Feature Adapted Convolutional Neural Networks for Downbeat Tracking," in *Proc. ICASSP,* 2016.

[7] Krebs F, Böck S, Dorfer M, Widmer G., "Downbeat Tracking using Beat-Synchronous Features and Recurrent Neural Networks," *in Proc. ISMIR,* 2016.

[8] E. D. Scheirer, "Tempo and Beat Analysis of Acoustic Musical Signals," *J. Acoust. Soc. Amer*., 103(1), pp. 588–601, 1998.

[9] Dixon S., "Evaluation of the Audio Beat Tracking System Beatroot," *Journal of New Music Research.* 2007 Mar 1;36(1):39-50.

[10] Oliveira JL, Davies ME, Gouyon F, Reis LP., "Beat Tracking for Multiple Applications: A Multi-Agent System Architecture with State Recovery," *IEEE Trans. Audio, Speech, Lang. Process.*, 20(10), pp. 2696-2706, Dec. 2012.

[11] D.P.W Ellis, "Beat Tracking by Dynamic Programming." *Journal of New Music Research*, 36(1), pp. 51-60, 2007.

[12] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stafylakis, "Music Tempo Estimation and Beat Tracking by Applying Source Separation and Metrical Relations," in *Proc. ICASSP*, 2012.

[13] M. E. P. Davies and M. D. Plumbley, "Context-Dependent Beat Tracking of Musical Audio," *IEEE Trans. Audio, Speech, Lang. Process.*, 15(3), pp. 1009–1020, Mar. 2007.

[14] G. Peeters and H. Papadopoulos, "Simultaneous Beat and Downbeat Tracking Using a Probabilistic Framework: Theory and Large-Scale Evaluation," *IEEE Trans. Audio, Speech, Lang. Process.*, 19(6), pp. 1754–1769, Aug. 2011.

[15] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the Meter of Acoustic Musical Signals," *IEEE Trans. Audio, Speech, Lang. Process.*, 14(1), pp. 342–355, Jan. 2006.

[16] S. Böck and M. Schedl. "Enhanced Beat Tracking with Context-Aware Neural Networks," in *Proc. DAFX,* 2011.

[17] Elowsson A. "Beat Tracking with a Cepstroid Invariant Neural Network," *in Proc. ISMIR*, 2016.

[18] Schluter J, Bock S., "Improved Musical Onset Detection with Convolutional Neural Networks," *in Proc. ICASSP*, 2014.

[19] Ullrich K, Schlüter J, Grill T., "Boundary Detection in Music Structure Analysis using Convolutional Neural Networks," *in Proc. ISMIR*, 2014.

[20] Humphrey EJ, Bello JP. Rethinking Automatic Chord Recognition with Convolutional Neural Networks," *in Proc. ICMLA*, 2012.

[21] Li, T.L., Chan, A.B. and Chun, A., "Automatic Musical Pattern Feature Extraction Using Convolutional Neural Network," in *Proc. Int. Conf. Data Mining and Applications*, 2010.

[22] Dieleman, S., Brakel, P. and Schrauwen, B., "Audio-Based Music Classification with a Pretrained Convolutional Network," in *Proc. ISMIR*, 2011.

[23] Durand S, Bello JP, David B, Richard G., "Robust Downbeat Tracking Using an Ensemble of Convolutional Networks," in *IEEE/ACM Trans. Audio, Speech, Lang. Process.,* 25(1), pp. 76-89, Jan. 2017.

[24] Krebs F, Böck S, Dorfer M, Widmer G., "Downbeat Tracking using Beat-Synchronous Features and Recurrent Neural Networks," in *Proc. ISMIR*, 2016.

[25] Pons J., and Serra X., "Designing Efficient Architectures for Modeling Temporal Features with Convolutional Neural Networks," *in Proc. ICASSP*, 2017.

[26] Aucouturier, J.J., Ikeuchi, K., Hirukawa, H., Nakaoka, S.I., Shiratori, T., Kudoh, S., Kanehiro, F., Ogata, T., Kozima, H., Okuno, H.G. and Michalowski, M.P., 2008. Cheek to Chip: Dancing Robots and AI's Future. *IEEE Intelligent Systems*, *23*(2).

[27] Michalowski, M.P., Sabanovic, S. and Kozima, H., "A Dancing Robot for Rhythmic Social Interaction," in Proc. HRI, 2007.

[28] Shinozaki, K., Iwatani, A. and Nakatsu, R., "Concept and Construction of a Robot Dance System," in *Proc. International Workshop and Conference on Photonics and Nanotechnology,* 2007.

[29] Grunberg, D., Ellenberg, R., Kim, I.H., Oh, J.H., Oh, P.Y. and Kim, Y.E., 2010. Development of an Autonomous Dancing Robot. *International Journal of Hybrid Information Technology*, *3*(2), pp.33-44.

[30] Murata, K., Nakadai, K., Yoshii, K., Takeda, R., Torii, T., Okuno, H.G., Hasegawa, Y. and Tsujino, H., "A Robot Singer with Music Recognition Based on Real-Time Beat Tracking." in Proc. *ISMIR,* 2008.

[31] Oliveira, J.L., Ince, G., Nakamura, K., Nakadai, K., Okuno, H.G., Gouyon, F. and Reis, L.P., "Beat Tracking for Interactive Dancing Robots," in *International Journal of Humanoid Robotics*, *12*(04), p.1550023.

[32] Gkiokas A., Katsouros V., and Carayannis G, "Towards Multi-Purpose Spectral Rhythm Features. An Application to Dance Style, Meter and Tempo,"

IEEE/ACM Trans. Audio, Speech, Lang. Process., 24(11), April 2016, pp. 1885-1896.

[33] C. Jin, M. E. P. Davies and P. Campisi. "Embedded Systems Feel the Beat in New Orleans: Highlights from the IEEE Signal Processing Cup 2017 Student Competition," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 143-170, July 2017

[34] Marchand U., and Peeters G. "Swing Ratio Estimation" *in Proc. DAFX,* 2015.

[35] Krebs F., Böck S. and Widmer G. "Rhythmic Pattern Modeling for Beat and Downbeat Tracking in Musical Audio," *in Proc. ISMIR,* 2013.

[36] Holzapfel A., Davies M. E. P., Zapata J. R., Oliveira J., and Gouyon F. "Selective Sampling for Beat Tracking Evaluation", *IEEE Trans. Audio, Speech, Lang. Process.*, 20(9), pp.2539-2548, 2012.

EARLY MFCC AND HPCP FUSION FOR ROBUST COVER SONG IDENTIFICATION

Christopher J. Tralie
Duke University Department of Electrical and Computer Engineering
`ctralie@alumni.princeton.edu`

ABSTRACT

While most schemes for automatic cover song identification have focused on note-based features such as HPCP and chord profiles, a few recent papers surprisingly showed that local self-similarities of MFCC-based features also have classification power for this task. Since MFCC and HPCP capture complementary information, we design an unsupervised algorithm that combines normalized, beat-synchronous blocks of these features using cross-similarity fusion before attempting to locally align a pair of songs. As an added bonus, our scheme naturally incorporates structural information in each song to fill in alignment gaps where both feature sets fail. We show a striking jump in performance over MFCC and HPCP alone, achieving a state of the art mean reciprocal rank of 0.87 on the Covers80 dataset. We also introduce a new medium-sized hand designed benchmark dataset called "Covers 1000," which consists of 395 cliques of cover songs for a total of 1000 songs, and we show that our algorithm achieves an MRR of 0.9 on this dataset for the first correctly identified song in a clique. We provide the precomputed HPCP and MFCC features, as well as beat intervals, for all songs in the Covers 1000 dataset for use in further research.

1. INTRODUCTION

A "cover song" is a different version of the same song, usually performed by a different artist, and often with different instruments, recording settings, mixing/balance, tempo, and key. To sidestep a rigorous definition, like others, we evaluate algorithms on a set of songs that have been labeled as covers of each other, and we declare success when our algorithm recognizes clusters of songs which have been deemed covers of each other. In fact, this problem is more of a "high level music similarity" task beyond exact recording retrieval, making the problem intrinsically more difficult than traditional audio fingerprinting [16, 31, 30].

Most work on automatic cover song identification to date has focused on estimating and matching note-based features such as chord estimates [1], chroma [8, 11], harmonic pitch class profiles (HPCP) [15, 23, 25], 2D Fourier magnitude coefficients to approximate these features [12, 19]. This is natural, since regardless of all of the transformations that can happen between versions, note sequences should be preserved up to transpositions. Problems occur, however, when note sequences are not the defining characteristic of a musical expression. This is common in hip hop, for example, such as the song "Tricky" in the "Covers 80 Dataset" ([10], Section 4.1) performed by Run D.M.C. and The Beastie Boys. There are also songs which are entirely percussive, such as the 8 covers of Frank Zappa's song "The Black Page" that we present in Section 4.3. Moreover, even for song pairs with strong harmonic content, there may be sections with drum solos, un-pitched spoken words, or other musical statements on which pitch-based features fail. However, the issue with features complementary to Chroma, such as Mel-Frequency Cepstral Coefficients (MFCCs), is that they are highly sensitive to instrument and balance changes. In spite of this, some recent works have shown that MFCC-based features can also be used in cover song identification [5, 29]. Particularly, if the *relative* changes of MFCC are captured, as in [29], performance is still reasonable.

Naturally, then, rather than relying on a single feature in isolation, recent works have shown the benefits of feature fusion after song comparisons have been made with each feature set alone. For instance, aggregating ranks from individual features can improve results [20, 21]. Other works show the advantage of using all pairwise similarities computed with different features [6], using a cross-diffusion process known as "similarity network fusion" (SNF) ([32, 33], Section 3.1) to come up with a consensus similarity score between all pairs of songs in a corpus.

In this work, we develop a similarity network-based early fusion technique which achieves state of the art results by combining complementary HPCP, MFCC, and self-similarity MFCCs (SSM MFCCs). Unlike [6], we apply SNF *before alignment* in the space of features. This fusion technique incorporates both cross-similarity between two different songs and self-similarity of each song, so it is able to combine information about matching sections between songs and structural elements within each song. We also apply late fusion on similarity scores between a network of songs to further boost the results. While state of the art supervised techniques on the popular "Covers 80" benchmark dataset yield a mean reciprocal rank (MRR) of 0.625 [6] [1], our completely unsupervised tech-

[1] This technique scored the best in MIREX 2016

nique achieves a MRR of 0.87 (Section 4.1). We also introduce our own dataset consisting of 395 cliques of songs, which we call "Covers1000" (Section 4.2), and on which we report an MRR of 0.9 with our best fusion technique.

2. BEAT-SYNCHRONOUS BLOCKED FEATURES

In this section, we describe three complementary features which we later fuse together in Section 3. One concept we rely on in our feature design is "block-windowing," which was described in [29]. The idea is that when a contiguous set of features in time are stacked into one large vector, they give more information about the local dynamics of a song. This is related to the concept of a delay embedding in dynamical systems [17]. Having many blocks across the song also allows us to control for drift by normalizing within each block. To control for tempo differences, we compute our blocks synchronized with beats, which is a common preprocessing step [8, 11, 29]. We use the simple dynamic programming approach of [9], since it allows the specification of a tempo bias. As in [11] and [29], we bias the beat tracker at 3 different tempo levels (60, 120, 180bpm), and we compare all pairs of tempo levels, for up to 9 unique combinations, since the beat tracker may choose an arbitrary level of subdivision in a rhythm hierarchy. Once the beat onsets are computed, we form a block for every contiguous group of B beat intervals, as in [29].

2.1 HPCP Features

One proven set of pitch-based features for cover song identification are "harmonic pitch class profiles" (HPCPs) [14] [2]. Following [26], we compute a stacked delay embedding of the HPCP features within B beats, with two HPCP windows per beat, for a total of $2B$ windows per block. This has an advantage over other works which do not use a delay, as it gives more context in time, and it is consistent with the block/windowing framework. To normalize for key transpositions, we need to determine an "optimal transposition index" (OTI) between two songs ([24]). Given the average HPCP vector $X \in \mathbb{R}^{+12}$ from song A and the average HPCP vector $Y \in \mathbb{R}^{+12}$ from song B, we compute the correlation $X^T Y$ over all 12 half-step transpositions of the original HPCP features in the block, and we use the transposition that leads to the maximum correlation. Then, we compute cosine distance between all pairs of HPCP blocks between the two songs.

2.2 MFCCs / MFCC Self-Similarity Matrices (SSMs)

In addition to HPCP features, we compute exponentially liftered MFCCs in beat-synchronous blocks. We take the MFCC window size to be 0.5 seconds, and we advance the window intervals evenly from the beginning of the block to the end of a block with a *hop size* of 512 samples. At a sample rate of 44100 Hz, this leads to a window size of 22050 and an overlap of roughly 97.5% between windows (Section 2.2). Longer windows have been shown to increase robustness of SSM matching in [29] and audio fingerprinting [16], which justifies this choice. To allow direct comparisons between different blocks, we interpolate to 400 MFCCs per block, and we perform Z-normalization (as in [29]) to control for loudness and drift, which we found to be an essential step.

[2] To ensure we have a state of the art implementation, we use the Essentia library to compute HPCPs [4].

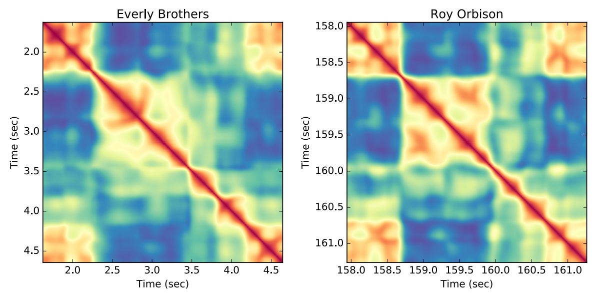


Figure 1: Example 8-beat Z-normalized MFCC SSMs blocks in correspondence between cover versions. A block from "Claudette" by the Everly Brothers and Roy Orbison. The pattern in this block is Guitar + "Oooh ooh Claudette" + Guitar.

In addition to block-synchronized and normalized raw MFCCs, we also compute self-similarity matrices (SSMs) of the Z-normalized MFCCs within each block, as in [29], leading to a sequence of SSMs for each song. That is, unlike [2], who compares SSMs between entire songs, we compare SSMs summarizing blocks of audio on the order of tens of beats, as recommended by [29]. For each beat-synchronous block, we create a Euclidean SSM between all MFCC windows in that block. As with the raw MFCCs, to allow comparisons between blocks, we resize each SSM to a common image dimension $d \times d$. Figure 1 shows an example of MFCC SSM blocks with 8 beats and 500 windows per block which were matched between a song and its cover in the Covers80 dataset. Although the underlying sounds are quite different (male to female, different instruments and balance), the SSMs look similar. [29] argue that this is why, counter to prior intuition, it is possible to use MFCCs in cover songs.

2.3 Cross-Similarity Matrices (CSMs) between Blocks

Given a set of M beat-synchronous block features for a song A and a set of N beat-synchronous block features for a song B, we compare all pairs of blocks between the two songs for that feature set, yielding an $M \times N$ cross-similarity matrix (CSM), which can be used to align the songs. For each song, we have 3 different CSMs for the three different feature sets, which are each aligned at the same beat intervals. For MFCC and MFCC SSMs, we use the Euclidean distance (Frobenius norm) to create the CSM, while for HPCP, we use the cosine distance after OTI. We then use the Smith Waterman algorithm [28] to find the best locally aligned sections between a pair of songs. To apply Smith Waterman, we turn the CSM into a binary matrix B^M, so that $B^M_{ij} = 1$ if CSM_{ij} is within the κN^{th} smallest values in row i of the CSM and if CSM_{ij} is within the κM^{th} smallest values in column j of the CSM, and 0 otherwise, where $\kappa \in (0, 1)$ is a *mutual nearest neighbors* threshold. As shown by [25] and [29], this is a crucial step for improving robustness.

Once we have the B^M matrix, we use a diagonally constrained variant of Smith Waterman to locally align the two songs. The score returned by this algorithm roughly corresponds to the length of the longest interval of consecutive blocks matched between songs, with more tolerance for gaps than a naive cross-correlation. More details can be found in[25] and [29]. For comparison, we perform alignments with the CSMs obtained for each feature individually and on the CSM obtained from fusing them.

3. FEATURE FUSION

Figure 2 shows an overall pipeline for the fusion process. We first briefly review the general mathematical technique that we use to fuse cross-similarity matrices from different feature sets, and then we show specifically how we apply it in our pipeline.

3.1 Similarity Network Fusion (SNF)

Given all pairs of similarity scores between objects using different features, Similarity Network Fusion (SNF) is designed to find a better matrix of all pairwise scores that combines the strengths of each feature [32, 33]. Roughly, it performs a random walk using the nearest neighbor sets from one feature while using the transition probability matrices from all of the other features, while continuously switching which feature set is used to determine the nearest neighbors. More precisely, the algorithm is defined as follows. First, start with a pairwise distance function $\rho(i, j)$ between objects for each feature type, and create an exponential kernel $W(i, j) = e^{-\rho^2(i,j)/2(\sigma_{ij})^2}$, where σ_{ij} is a neighborhood-autotuned scale which is the average of $\rho(i, j)$ and the mean k-nearest neighbor intervals of blocks i and j, for some k (see [33] for more details). Now, create the following Markov transition matrices

$$P(i,j) = \left\{ \begin{array}{cc} \frac{1}{2}\frac{W(i,j)}{\sum_{k\neq i} W(i,k)} & j \neq i \\ 1/2 & \text{otherwise} \end{array} \right\} \quad (1)$$

This is simply a first order Markov chain with a regularized diagonal to promote self-recurrence. Once this matrix has been obtained, create a truncated k-nearest neighbor version of this matrix

$$S(i,j) = \left\{ \begin{array}{cc} \frac{W(i,j)}{\sum_{k\in N(i)} W(i,k)} & j \in N(i) \\ 0 & \text{otherwise} \end{array} \right\} \quad (2)$$

where $N(i)$ are the k nearest neighbors of vertex i, for some chosen k. Now let P^f and S^f be the P and S matrices for the f^{th} feature set, and let $P^f_{t=0} = P^f$. Then define a first order "cross-diffusion" random walk recursively as follows

$$P^f_{t+1} = S^f \left(\frac{\sum_{v\neq f} P^v_t}{m-1} \right) (S^f)^T \quad (3)$$

where m is the total number of features. In other words, a random walk is occurring but with probabilities that are modulated by similarity kernels from other features. As shown by [32], this process will eventually converge, but we can cut it off early. Whenever it stops, the final fused transition probabilities are $\hat{P}_t = \frac{1}{m}\sum_{k=1}^{M} P^k_t$.

3.2 Late SNF

One way to use SNF is to let the matrix $\rho(i, j)$ be all pairwise distance between songs, computed by some alignment [6]. In this case, the result should be a better network of similarity scores between all songs [3]. We follow a similar approach to [6], but we we work with the Smith Waterman scores we get from a unique combination of MFCC, MFCC SSM, and HPCP blocks ([6] applied SNF to different alignment schemes on the same feature set). Given a particular score matrix S between all pairs of songs, we compute the kernel matrix W as $W(i, j) = 1/S(i, j)$. Since Smith Waterman gives a higher score for better matching songs, this ensures that the kernel is close to 0 in this case. At this point, we perform SNF, and we obtain a final $N \times N$ transition probability matrix P. We can then look along each row to find the neighboring songs with maximum fused probability. This process can be thought of as exploiting the *network* of all songs in a collection in an unsupervised manner.

3.3 Early SNF

In addition to SNF after Smith Waterman has given scores, we can perform fusion at the feature level before running Smith Waterman. One advantage of doing fusion before scores are computed is that we don't need a network of songs to compute a score; we can obtain an improved score between two songs without any other context [4]. Our technique for early fusion, which we found to be superior to the "OR fusion" proposed in [13], is to apply SNF on the beat-aligned cross-similarity matrices obtained from two or more different feature sets before creating a binary cross-similarity matrix (CSM) and applying Smith Waterman.

As defined in Section 3.1, SNF would operate on self-simlarity matrices (SSMs), so it cannot be directly applied to cross-similarity matrices. To make it so that CSMs fit into the framework, we create a "parent SSM" for each feature set that holds both SSMs and the CSM for that feature set. In particular, given song A with M blocks in a particular feature set and song B with N blocks in that feature set, form the SSM D_{AB} which is the SSM that results after concatenating song B to the end of song A. Let the SSM for song A be D_A, the SSM for song B be D_B, and the

[3] Note that [27] essentially do the same thing with only one feature set

[4] Note that [6] refer to SNF after Smith Waterman as "early fusion" with respect to rank aggregation, which they call "late fusion," but we call their technique "late fusion" because we fuse before Smith Waterman with SNF, which is even earlier in the pipeline.

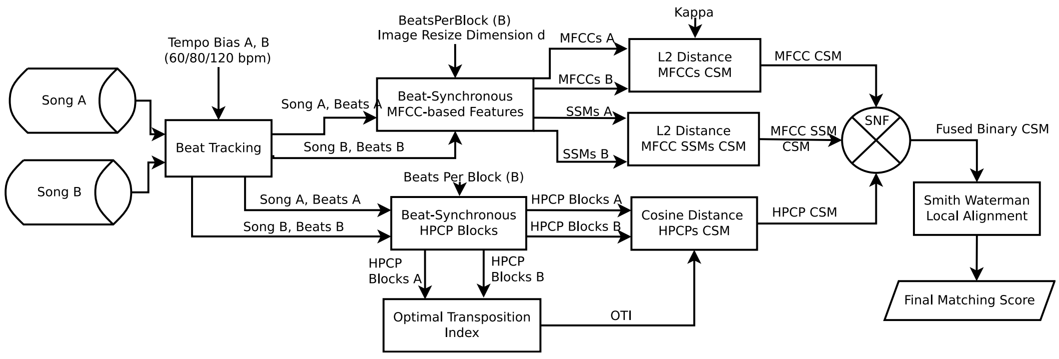

Song A
Song B
Tempo Bias A, B (60/80/120 bpm)
Beat Tracking
BeatsPerBlock (B) Image Resize Dimension d
Song A, Beats A
Song B, Beats B
Beat-Synchronous MFCC-based Features
MFCCs A
MFCCs B
SSMs A
SSMs B
Kappa
L2 Distance MFCCs CSM
L2 Distance MFCC SSMs CSM
MFCC CSM
MFCC SSM CSM
SNF
Fused Binary CSM
Smith Waterman Local Alignment
Final Matching Score
Beats Per Block (B)
Beat-Synchronous HPCP Blocks
HPCP Blocks A
HPCP Blocks B
Cosine Distance HPCPs CSM
HPCP CSM
Optimal Transposition Index
OTI

Figure 2: A block diagram of our system which performs early similarity network fusion of blocked MFCCs, MFCC SSMs, and HPCPs before scoring with Smith Waterman alignment.

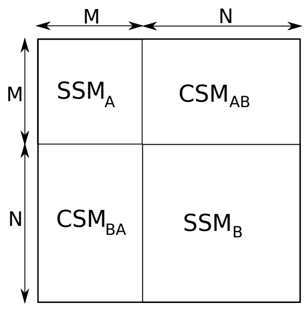


Figure 3: A pictorial representation of the SSM that results when concatenating song B to song A, which we feed to SNF for early fusion of low level features. Including self-similarity blocks of each song helps to promote structural elements in cross-similarity regions during SNF.

CSM between them be C_{AB}. Then D_{AB} can be split into four sub-blocks:

$$D_{AB}(i,j) = \left\{ \begin{array}{cc} D_A(i,j) & i < M, j < M \\ D_B(i-M, j-M) & i >= M, j >= M \\ C_{AB}(i, j-M) & i < M, j >= M \\ C_{BA}(i-M, j) = & \\ C_{AB}^T(j, i-M) & i >= M, j < M \end{array} \right\} \quad (4)$$

Figure 3 shows this pictorially. Given such a matrix for each feature set, we could then run SNF and extract the cross-similarity sub-matrix at the end. The issue with this is the dynamic range of the SSM may be quite different from the dynamic range of the CSM, as it is likely that blocks in song A are much more similar to other blocks in song A than they are to blocks in B. To mitigate this, given a nearest neighbor threshold κ for the CSM, we compute the kernel neighborhood scales σ_{ij} individually for D_A, D_B, and CSM_{AB}, and we put them together in the final kernel matrix W_{AB} according to Figure 3. Once we have such a matrix W_{AB} for each feature set, we can finally perform SNF. At the end, we will end up with a fused probability matrix P, from which we can extract the cross-probability $P_{C_{AB}}$. We can then take mutual highest probabilities (akin to mutual nearest neighbors) to extract a binary matrix and perform Smith Waterman as normal. Figure 4 shows an example of the constructed matrices W_{AB}

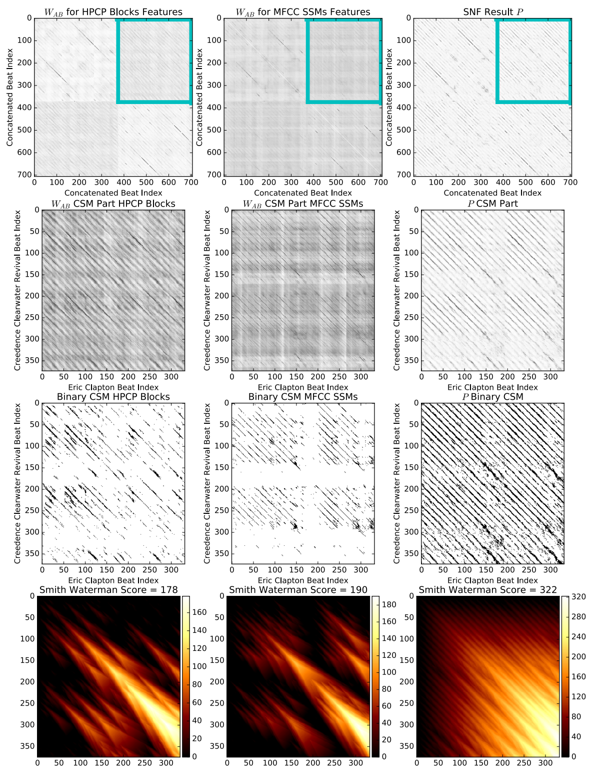


Figure 4: An example of early SNF on blocks of MFCC SSMs and blocks of HPCP features on the song "Before You Accuse Me" with versions by Eric Clapton and Creedence Clearwater Revival. The block size is 20 beats, and there are three iterations of SNF. The kernels W_{AB} are shown for each, and the CSM portion is highlighted with a blue box. The final fused probability matrix P returned from SNF is shown in the upper right. The corresponding CSM portions for all three matrices shown for each on the bottom. In the fused probability matrix, the diagonal regions are much crisper and more distinct from the background than they are for the individual feature sets. The result is that the mutual nearest neighbors binary CSM has longer uninterrupted diagonals, which is reflected by a higher Smith Waterman score.

	MR	MRR	Top-01	Top-10	../80
MFCCs	29.7	0.538	79	97	42/80
SSMs	15.1	0.615	91	111	48/80
HPCPs	18.2	0.673	102	119	53/80
Late SSMs/MFCCs	14.0	0.7	107	125	55/80
Late All	8.63	0.824	127	141	64/80
Early	7.76	0.846	131	143	68/80
Early + Late	7.59	0.873	136	144	69/80
[6]	?	0.625	?	114	?

Table 1: Results of different features and fusion techniques on the Covers 80 dataset.

and the resulting fused probabilities P.

One advantage of this technique is that since the CSM and SSMs are treated together and normalized to a similar range, any recurrent structure which exists in the SSMs can reinforce otherwise weaker structure in the CSMs during the diffusion process. This can potentially help to strengthen weaker beat matches in an otherwise well-matching section, leading to longer uninterrupted diagonals in the resulting binary CSM.

3.4 Early Fusion Examples

Before we launch into a more comprehensive experiment, we show a few examples of early SNF to illustrate the value added. In each example, we used 20-beat blocks, $\kappa = 0.1$ for both similarity fusion and binary nearest neighbors, and 3 iterations of SNF. Figure 5 shows an example where the three individual features are rather poor by themselves, but where they happen to all pick up on similarities in complementary regions. As a result, early SNF does a fantastic job fusing the features. Figure 6 shows an example where MFCC SSMs happen to do better than HPCP, but where the results fusing both are still better than each individually.

4. EXPERIMENTS

We are now ready to evaluate the performance of this new algorithm. In all of our experiments below, we settle on $\kappa = 0.1$ (the mutual nearest neighbor threshold for binary CSMs) and $B = 20$ beats per block. For similarity network fusion, we take 20 nearest neighbors for both early fusion and late fusion, we perform 3 iterations for early fusion, and we perform 20 iterations for late fusion. We also include an "early + late" fusion result, which is applying late fusion to the network of similarities obtained from all of the feature sets (MFCCs, MFCC SSMs, HPCPs) plus the network of similarities obtained from the early fusion of the three feature sets.

4.1 Covers 80 Dataset

To benchmark our algorithm, we begin by testing it on the "Covers 80" dataset [10]. This dataset contains 160 songs which are split into two disjoint subsets A and B, each with exactly one version of a pair of songs, for a total of 80 pairs. [8] and [11] assess performance as follows: given a song in group A, declare its cover song to be the top ranked song in set B, and record the total number of top ranked songs that are correct. To get a better idea of the performance, we also compute the mean rank (MR), mean reciprocal rank (MRR), and the number of songs correctly identified past a certain number. All of these statistics are computed on the full set of 160 songs, which is more difficult than simply looking in set A or set B.

Table 1 shows the results. By themselves, HPCP features perform better than MFCC-based features, which is consistent with findings in the literature. However, there are big improvements when fusing them all. Surprisingly, we obtain a score of 42/80 just by blocking and normalizing the MFCCs. This shows the power of having stacked delay MFCCs and of normalizing within each block to cut down on drift. Also, when fusing MFCCs and MFCC SSMs with late fusion, we get a large performance boost over either alone, showing that SSMs are adding complementary information to the MFCCs they summarize.

	MR	MRR	Top-01	Top-10
MFCCs	83.3	0.618	583	679
SSMs	72.5	0.623	581	698
HPCPs	44.4	0.757	727	809
Late	19.8	0.875	855	931
Early	22.5	0.829	798	884
Early + Late	14	0.904	884	950

Table 2: Results of different features and fusion techniques on the Covers 1000 dataset.

4.2 Covers 1000 Dataset

To test our algorithm more thoroughly, we created our own dataset by manually choosing 1000 cover songs (395 cliques total) based on annotations given by users on `http://www.secondhandsongs.com`[5]. This dataset covers over a century of Western music from 1905 - 2016, and hence, it covers a wide variety of genres and styles. Figure 7 shows the full distribution of years covered. By contrast, the Covers80 dataset contains almost exclusively pop music from the '80s and early '90s.

Most cliques have only two songs as in the Covers80 dataset, but there are a few cliques with 3 and 4 songs. In this case, we report the MR and MRR of the first correctly identified song in the clique. Table 2 shows the results. Similar trends are seen to the Covers80 case, and performance scales to this larger size. One difference is that late fusion on HPCPs/MFCCs/MFCC SSMs performed better relative to early fusion, likely because the network was much richer with the additional volume of songs.

[5] MFCC and HPCP features for our dataset are publicly available at `http://www.covers1000.net`, along with beat intervals and other metadata including song title, album, and year

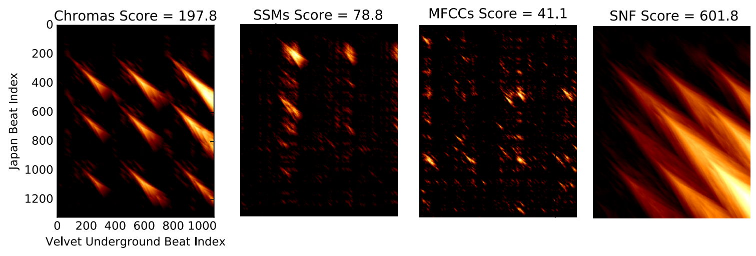


Figure 5: Smith Waterman tables/scores for "All Tomorrow's Parties" by Japan and Velvet Underground.

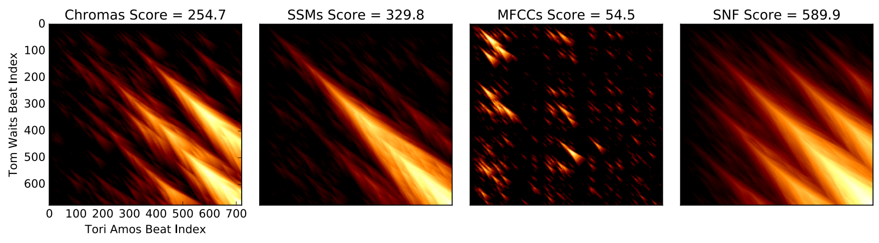


Figure 6: Smith Waterman tables/scores for "Time" by Tom Waits and Tori Amos

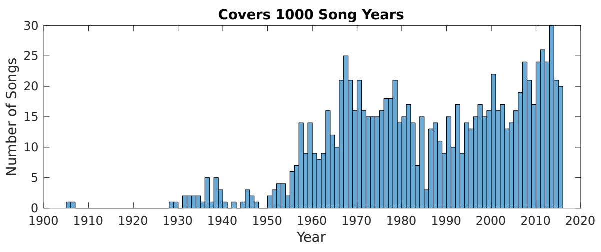


Figure 7: A distribution of years of songs in the Covers 1000 dataset.

4.3 Frank Zappa: "The Black Page"

In our final experiment, we test a clique of 8 cover versions of the song "The Black Page" by Frank Zappa, which is entirely a drum solo that has absolutely no harmonic content. We query each song against all of the songs in the Covers1000 dataset, and we compute the mean average precision (MAP) for the songs in the clique. Unsurprisingly, for HPCP, the MAP is a mere 0.014, while for the rest of the features these songs are quite distinct from the rest of the songs in the Covers 1000 dataset. The best performing feature set is early SNF, with a MAP of 0.98, followed by raw blocked MFCCs at a MAP of 0.97, followed by MFCC SSMs with a MAP of 0.905.

5. DISCUSSION

In this work, we have demonstrated the benefit of combining complementary features at a very early stage of the cover song identification pipeline, in addition to the late fusion techniques in [6]. Unlike [6] and other techniques, our algorithm works on a pair of songs and does not need a network of songs to improve performance, though we show that incorporating information from a network of songs ("late fusion") can further improve results. We showed that HPCP and MFCC features capture complementary information and are able to boost performance substantially over either alone. In the process, we also developed a novel cross-similarity fusion scheme which was validated on several datasets, and which we believe could be useful beyond cover song identification in music structure analysis.

The main drawback of our technique is the requirement of beat tracking. In practice, beat trackers may not return correct onsets. Our current best remedy for this is to use different tempo biases, which blows up computation by a factor of 9. Also, coming up with a single beat level is ill-posed, since most music consists of a hierarchy of rhythmic subdivisions [22]. There does seem to be a recent convergence of techniques for rhythm analysis, though, [7, 18] so hopefully our system will benefit.

In addition to imperfect beat intervals, there are also computational drawbacks in low level alignment, which is why most recent works on cover songs perform approximations to global cross-correlation, such as 2D Fourier Magnitude Coefficients [12, 19]. By contrast, we rely on Smith Waterman, which is a quadratic algorithm, and early SNF adds another quadratic time complexity algorithm even with sparse nearest neighbor sets. To address this, we are in the process of implementing GPU algorithms for every step of our pipeline, and we hope to apply it to the "Second Hand Songs Dataset," which is a subset of the Million Songs Dataset [3].

6. ACKNOWLEDGEMENTS

Christopher Tralie was partially supported by an NSF Graduate Fellowship NSF under grant DGF-1106401 and an NSF big data grant DKA-1447491. We would also like to thank Erling Wold for pointing out the 8 covers of "The Black Page" by Frank Zappa, and we would like to thank the community at `www.secondhandsongs.com` for meticulously annotating songs which helped us to design Covers 1000.

7. REFERENCES

[1] Juan Pablo Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *ISMIR*, volume 7, pages 239–244, 2007.

[2] Juan Pablo Bello. Grouping recorded music by structural similarity. In *ISMIR 2009, Kobe, Japan, 2009*, pages 531–536, 2009.

[3] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR*, volume 2, page 10, 2011.

[4] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R. Zapata, and Xavier Serra. Essentia: An audio analysis library for music information retrieval. In *ISMIR 2013, Curitiba, Brazil, 2013*, pages 493–498, 2013.

[5] Ning Chen, J Stephen Downie, Haidong Xiao, Yu Zhu, and Jie Zhu. Modified perceptual linear prediction liftered cepstrum (mplplc) model for pop cover song recognition. In *ISMIR*, pages 598–604, 2015.

[6] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, pages 1–24, 2017.

[7] Norberto Degara, Enrique Argones Rúa, Antonio Pena, Soledad Torres-Guijarro, Matthew EP Davies, and Mark D Plumbley. Reliability-informed beat tracking of musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):290–301, 2012.

[8] Daniel PW Ellis. Identifying'cover songs' with beat-synchronous chroma features. *MIREX 2006*, pages 1–4, 2006.

[9] Daniel PW Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[10] Daniel PW Ellis. The "covers80" cover song data set. *URL: http://labrosa. ee. columbia. edu/projects/coversongs/covers80*, 2007.

[11] Daniel PW Ellis and Courtenay Valentine Cotton. The 2007 labrosa cover song detection system. *MIREX 2007*, 2007.

[12] Daniel PW Ellis and Bertin-Mahieux Thierry. Large-scale cover song recognition using the 2d fourier transform magnitude. In *The 13th international society for music information retrieval conference*, pages 241–246, 2012.

[13] Rémi Foucard, J-L Durrieu, Mathieu Lagrange, and Gaël Richard. Multimodal similarity between musical streams for cover version detection. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 5514–5517. IEEE, 2010.

[14] Emilia Gómez. Tonal description of polyphonic audio for music content processing. *INFORMS Journal on Computing*, 18(3):294–304, 2006.

[15] Emilia Gómez and Perfecto Herrera. The song remains the same: identifying versions of the same piece using tonal descriptors. In *ISMIR*, pages 180–185, 2006.

[16] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *Ismir*, volume 2002, pages 107–115, 2002.

[17] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*, volume 7. Cambridge university press, 2004.

[18] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An efficient state-space model for joint tempo and meter tracking. In *ISMIR*, pages 72–78, 2015.

[19] Oriol Nieto and Juan Pablo Bello. Music segment similarity using 2d-fourier magnitude coefficients. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 664–668. IEEE, 2014.

[20] Julien Osmalsky, Jean-Jacques Embrechts, Peter Foster, and Simon Dixon. Combining features for cover song identification. In *16th International Society for Music Information Retrieval Conference*, 2015.

[21] Julien Osmalsky, Marc Van Droogenbroeck, and Jean-Jacques Embrechts. Enhancing cover song identification with hierarchical rank aggregation. In *Proceedings of the 17th International for Music Information Retrieval Conference*, pages 136–142, 2016.

[22] Elio Quinton, Christopher Harte, and Mark Sandler. Extraction of metrical structure from music recordings. In *Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx). Trondheim*, 2015.

[23] J Serra. Music similarity based on sequences of descriptors: tonal features applied to audio cover song identification. *Department of Information and Communication Technologies, Universitat Pompeu Fabra, Barcelona, Spain*, 2007.

[24] Joan Serra, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.

[25] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(6):1138–1151, 2008.

[26] Joan Serra, Xavier Serra, and Ralph G Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.

[27] Joan Serrà, Massimiliano Zanin, Perfecto Herrera, and Xavier Serra. Characterization and exploitation of community structure in cover song networks. *Pattern Recognition Letters*, 33(9):1032–1041, 2012.

[28] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[29] Christopher J Tralie and Paul Bendich. Cover song identification with timbral shape sequences. In *16th International Society for Music Information Retrieval (ISMIR)*, pages 38–44, 2015.

[30] Avery Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44–48, 2006.

[31] Avery Wang et al. An industrial strength audio search algorithm. In *ISMIR*, pages 7–13. Washington, DC, 2003.

[32] Bo Wang, Jiayan Jiang, Wei Wang, Zhi-Hua Zhou, and Zhuowen Tu. Unsupervised metric fusion by cross diffusion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2997–3004. IEEE, 2012.

[33] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333–337, 2014.

Exploring the Music Library Association Mailing List: A Text Mining Approach

Xiao Hu[1] **Kahyun Choi**[2] **Yun Hao**[2] **Sally Jo Cunningham**[3] **Jin Ha Lee**[4] **Audrey Laplante**[5] **David Bainbridge**[3] **J. Stephen Downie**[2]

[1]University of Hong Kong
xiaoxhu@hku.hk

[2]University of Illinois
{ckahyu2, yunhao2, jdownie}@illinois.edu

[3]University of Waikato
{sallyjo, davidb}@waikato.ac.nz

[4]Univeristy of Washington
jinhalee@uw.edu

[5]Université de Montréal
audrey.laplante@umontreal.ca

ABSTRACT

Music librarians and people pursuing music librarianship have exchanged emails via the Music Library Association Mailing List (MLA-L) for decades. The list archive is an invaluable resource to discover new insights on music information retrieval from the perspective of the music librarian community. This study analyzes a corpus of 53,648 emails posted on MLA-L from 2000 to 2016 by using text mining and quantitative analysis methods. In addition to descriptive analysis, main topics of discussions and their trends over the years are identified through topic modeling. We also compare messages that stimulated discussions to those that did not. Inspection of semantic topics reveals insights complementary to previous topic analyses of other Music Information Retrieval (MIR) related resources.

1. INTRODUCTION

The Music Library Association Mailing List (MLA-L)[1] has a variety of subscribers including music librarians, MLA members, and professionals and students in music librarianship as well as musicology. The archive of this list serves as an invaluable resource for studying discussions among these people; it is the oldest and largest repository for studying issues in this profession [11].

Music librarians are an integral and important part of the larger Music Information Retrieval (MIR) community. The experiences, expertise, interests and concerns of music librarians are highly relevant to the advancement of MIR research. Similarly, MIR research can improve practices in music librarianship, ultimately enhancing the discovery of music information for diverse types of users [23]. Given the abundance of emails archived in the MLA-L, we can identify main topics discussed throughout the years which can offer insights into real-world music information interactions, particularly concerning the use and management of music information, from the perspectives of music information professionals and their clients. This study seeks to uncover the key topics of discussion related to music information needs and uses in the MLA mailing list by using text mining and quantitative analysis methods.

2. LITERATURE REVIEW

2.1 MLA-L Content Analysis

Interest in, and analysis of, the MLA-L collection is not new: nearly three decades ago when e-mail was the newest form of written communication, the list became the object of investigation (e.g., [5], [26]). Griscom [11] offered a detailed and relatively more recent account of the history and development of the MLA-L. Through qualitative content analysis, Griscom organized postings in the "E-Mail Digest" column of the MLA-L archive into nine categories as shown in Table 1.

Category	Definition
Reference questions	questions on locating songs or music work of specific topics or sources
Cataloging	extended discussions on catalogs of music library collections
Practical matters	problems unique to music libraries such as circulation and preservation of holdings
Technology	questions and opinions about adapting to technological advancements
Ethics	questions on unexpected and controversial topics such as illegal items
Copyright	questions and comments on reproduction matters and copyright laws
Circulation policies	policies and procedures posed by special formats in music libraries
Assisting colleagues	alerts on problems and peculiarities such as production errors.
MLA matters	communications from the board of directors of the association

Table 1. Main categories of MLA-L postings in [11].

While it is noteworthy that a majority of these categories were consistent with the categorization in earlier studies on the MLA-L [5], [26], analytical research on mailing lists or discussion forums of library professionals

[1] https://www.musiclibraryassoc.org/?page=mlal

should not only identify categories of messages, but also consider what specific topics were discussed [10]. More importantly, the data analyzed in the most recent endeavor of MLA-L content analysis [21] were up to year 1998 which was nearly two decades ago, and thus inspiring the updated account via current exploration of the data.

2.2 Text Analysis in MIR

Text analyses of MIR-related resources have increasingly appeared in recent years in the MIR community. Downie and Cunningham [9] contributed an early analysis of postings of music-related information requests on a music newsgroup. In their results, “locate” was identified as a predominant intended use for the requested music information. This is consistent with the category of “reference questions” shown in Table 1. In fact, the category “reference questions” was identified in all the previous studies on the MLA-L [5], [11], [26].

Music related Q&A (question & answers) websites are also a rich resource of MIR-related discussion. Bainbridge et al. [1] analyzed users’ music queries in Google Answers using a grounded theory approach, revealing that bibliographic metadata were commonly included in users’ queries. This leads to corroboration with the “cataloging” category of MLA-L postings [11]; bibliographic metadata such as *performer* and *title* are necessary for catalogs in music libraries. A later study by Lee [16] employed content analysis, also on Google Answer queries, to look into music information seeking behaviors. The results revealed that the “location” of a music information object (e.g., a recording) was also one of the most prominent information needs.

In addition to newsgroups and Q&A websites, publications in MIR have also been analyzed. Lee et al. [16] examined papers in the proceedings of the Conference of the International Society for Music Information Retrieval (ISMIR). Analysis of keywords in titles and abstracts of ISMIR papers identified “audio” and “classification” as the most frequent terms. Most recently, Hu et al. [13] compared the keywords in titles of ISMIR papers written by female to those by male authors, revealing the gender-based differences of topic preferences between authors.

Lyrics, music reviews, users’ interpretations of music, and other types of listeners’ input in social media (e.g., social tags and tweets) have also been analyzed, mainly by automated methods [24], for various tasks in MIR such as genre classification [22], mood classification [14], and subject classification [7]. In these studies, natural language processing methods are applied to convert textual data into numerical data in large scales, which are then fed into a wide range of machine learning approaches to fulfill aforementioned MIR tasks. More often than not, such texts are used in combination with audio signals to further improve performances via multimodal approaches (e.g. [14]).

2.3 Topic Modelling and Trend Analysis

Machine learning and quantitative methods have demonstrated their capability in automating the processing of email messages [6] and other text input by users. Barua et al. [2] used a prevalent statistical modeling technique called Latent Dirichlet Allocation (LDA) [3] to discover topics and their trends in Q&A websites, in order to gain insights into the wants and needs of the participants. Prior studies employing topic modelling on email messages and replies are seemingly scarce. McCallum et al. [20], being one rare case, also used the LDA in analyzing an email corpus.

Exploration of the changes in topics found in a body of messages over time—trend analysis—is particularly important due to the evolving focus in documents such as emails and queries (Blei and Lafferty, 2003 as cited in [27]). Unsupervised topic modeling as an extension to LDA is one way to generate the temporal relationships of topics [12].

Mishne and Glance [21] showed that comments made by users on weblogs could be an indicator of popularity of posts or the weblogs themselves. The mechanism of MLA-L is also in the form of postings welcoming potential replies, and thus it is reasonable to evaluate the popularity of topics based on their corresponding replies.

To bridge the gaps in previous research, this study aims to answer the following research questions: 1) What are the primary topics discussed in the MLA-L list from 2000 to 2016?; 2) How did the strength of the topics change over time?; and 3) Which topics attracted replies and which did not? Answers to these questions will help MIR researchers and practitioners understand information needs of the community and identify potential use cases of MIR tasks and applications.

3. DATA STATISTICS

The corpus used in this study consists of 53,648 emails posted on MLA-L from 2000 to 2016 by approximately 2,713 people (Figure 1). Among these emails, 33,250 (61.98%) received no replies while the other 20,398 emails (38.02%) formed 8,384 distinct online conversations (email threads) with the largest thread containing 52 emails. The average length of an email is 177.5 words (after removing reply and signature blocks; blank emails not included) with the longest email containing 1,389 words, indicating that most of the emails contain substantial content.

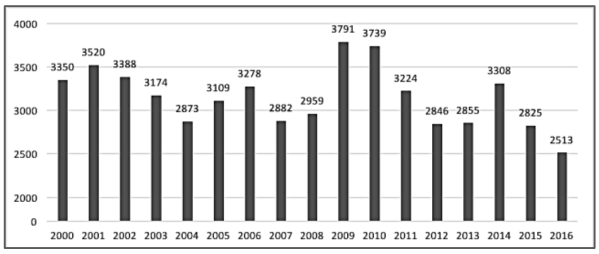


Figure 1. Number of emails in MLA-L across years.

4. PREPROCESSING

The data cleaning steps for preprocessing the large corpus include: 1) removing special formats and converting emails into plain text using OpenRefine[2]; 2) removing reply blocks (quoted emails included in reply emails); 3) detecting signature blocks by detecting variations of the sender's name using the tool Jangada[3] from the last 10 lines of the emails as suggested in [6]; 4) removing mailing list footers; and, finally 5) removing various leave-takings such as "best wishes" and "sincerely." Given the large amount of data, it is not possible to manually evaluate the accuracy of preprocessing on the entire corpus. Instead, inspection of a random sample of 100 preprocessed emails show 89% had the aforementioned noisy parts correctly removed.

The cleaned emails underwent text processing procedures for text mining purposes. Stopwords[4] that are extremely common in English were removed to increase the quality of discovered topics. In addition, words that occurred in more than 15% of the emails, such as "music" and "send", were also removed, since they had little discriminant power and thus could be regarded as domain stopwords. To enhance the readability of the topics, lemmatization[5] was applied to convert words in derivative forms into their lemmas, instead of crude stemming that usually cuts out suffixes of words. Finally, emails that were left with less than four words were eliminated as it would not be reliable to assign them to a certain topic based on too few words. Figure 2 illustrates an example email before and after preprocessing.

Original message: hi, does anyone know of a gospel song (or any song) with reference to "Shadrach, Meshach, and Abednego"? if you can provide me with titles, i would appreciate it.
Preprocessed message: gospel song song reference shadrach meshach abednego provide title

Figure 2. An example email from the MLA-L postings before and after preprocessing.

5. METHODS

5.1 Topic Modeling Setup and Labeling Procedure

We employed one of the popular topic modeling algorithms, Latent Dirichlet Allocation (LDA). It is a generative model that represents documents as probability distributions over topics, and represents topics as probability distributions over words [3]. In other words, given a set of documents, LDA can identify latent topics discussed in these documents based on word-document co-occurrences. As a probabilistic method, LDA can assign each document to a small number of topics with different probabilities. At the same time, each topic is represented by a small set of words that are highly related to this topic. Due to its superior performances compared to other topic modeling methods, LDA has been widely used to discover topics from diverse corpora such as papers, web postings, and even tweets. It is believed that LDA can also discover topics well from email content [20].

In this study, we ran the LDA implementation in the MALLET machine learning toolkit, which has been widely used in topic modeling research [19]. In LDA, the number of topics is tightly linked to the granularity of the learned topics. After changing the number of topics from 10 to 200, we report the case when the number of topics was set as 50, for a proper granularity of topics. To increase the quality of text analysis, we used bigrams (i.e., combinations of two consecutive terms) as well as unigrams (i.e., individual terms) as "words" when learning the topics [25].

The resultant topics then underwent a manual screening process, to filter out noisy, meaningless topics. This is a common practice in applying topic modeling (e.g., [12]). Those noisy topics were introduced by trivial text patterns that frequently appeared in the corpus. In our context, due to the inevitable side effect of the automatic approach to data preprocessing, which was mandated by the scale of the corpus, some signatures and forwarded message headers remained and were inputted into the topic modeling process. As a result, these signatures and senders were grouped into a noisy topic. Another example of a noisy topic is the agglomeration of words in languages other than English. Common email terms (e.g., "post", "email") and greeting/closing words also formed a topic which conveyed little meaning. Fortunately, topic modeling associates each identified topic with representative words, and thus it is convenient and reliable for researchers to examine and weed out noisy topics.

The remaining topics appear to be meaningful. To enhance readability of the topics, we manually labeled each topic with one or two phrases based on the meanings of the top 10 words and the top 50 emails associated with the topic. In particular, frequently appearing words in the subject lines of the highly ranked emails in each topic help us gain a deeper understanding of the topic.

5.2 Topic Trend Analysis

Besides identifying topics in the entire corpus, we also calculated the probability over topics given a particular year, $P(t|y)$, to examine the topical trend over time. To this end, we followed the empirical probability calculation procedure proposed in [12]. Based on the topic modeling result, we first computed a matrix $\boldsymbol{C}$ of $D\times T$ dimensions where D refers to the number of documents (emails) in our corpus and T the number of topics identified. The (d,t)-th element of the matrix $\boldsymbol{C}$ holds the number of words assigned to the t-th topics in the d-th document. From here, we can induce the probability over the topics per year y as follows:

[2] http://openrefine.org/
[3] http://www.cs.cmu.edu/~vitor/codeAndData.html
[4] https://code.google.com/archive/p/stop-words/
[5] http://morphadorner.northwestern.edu/morphadorner/

$$P(t|y) = \frac{\sum_{d=D_y} \boldsymbol{C}(d,t)}{\sum_{d=D_y} N_d}, \quad (1)$$

where D_y is the set of documents that belong to year y, and N_d is the number of words in the d-th document. The probability $P(t|y)$ over the years can then form a time series for topic t.

In order to determine whether there is a statistically significant upward or downward trend for a topic over time, Cox Stuart trend analysis was used with a significance level of 0.05 [18]. Cox Stuart trend analysis splits a time series into two halves and counts the numbers of positive and negative differences between pairs of data points drawn from the two halves. If there are more negative differences than positive ones, then an upward trend is detected, indicating that the topic gained more attention over the years, and vice versa.

5.3 Topics with and without Replies

To answer the third research question, we compared the topics associated with emails with replies and those without. As mentioned before, in the results of LDA, each email is assigned to multiple topics with different probabilities. In this analysis, we aggregated the topics and their probabilities across all emails either with or without replies. Then we ranked the topics based on their aggregated and normalized probabilities. By comparing the top topics of the two sets of emails, we can discover which topics were more engaging and generated more discussions among the MLA community.

6. RESULTS AND DISCUSSION

6.1 Discovered Topics

Upon careful screening, 27 of the resultant topics were identified as meaningful. Table 2 presents these topics, ranked by their topic weights. For each topic, Table 2 presents the topic IDs, the labels, the topic weight (in parentheses), frequent words in the subjects of top emails associated with this topic, the top 10 words assigned to this topic in the LDA results, trend over time (upward or downward as indicated by arrows), and the trendline that plots the probability change of this topic across the years.

The weight of the topic (the Dirichlet parameter) is roughly proportional to the overall portion of the documents assigned to a given topic [19]. Therefore, topics with higher weights are more popular in the corpus, while those with lower weights rarely appear. As shown in Table 2, it is not surprising that the most important topic in our corpus is about requests and questions from patrons (i.e., clients). Unlike other topics, there is no frequent word in subject lines for this topic. A closer examination revealed that it is because each subject belonging to this topic was unique. Based on this topic's highest weight, we can infer that there must have been a wide range of requests and questions from patrons. The second highest ranked topic is musical terms whose top words include a range of music genres, indicating that music librarians not only focus on Classical music (topic #35), but also on a wide diversity of music. Other top ranked topics cover various aspects of librarianship (e.g., #6: cataloging; #14: circulation; #22: collection), and music-specific materials: scores (#30), recordings (#47), and songs (#12).

In order to compare our results to categories identified in previous studies using content analysis, the 27 meaningful topics discovered in this study were manually grouped into nine broader topic categories based on their semantic similarity: Cataloging, Reference Questions, Circulation Policies, Copyright, Audio Technology, MLA, Advertisements, Music Related Terms, and Others. Five of the categories (Reference questions, Cataloging, Copyright, Circulation Policies, MLA) were equivalent with those uncovered by Griscom [11] (c.f. Table 1), while Audio Technology appears closely related to Griscom's "Technology." The Advertisement (CD/DVD sales, Travel information, Job postings) category is novel to our findings.

Our discovered topics are also consistent with results from earlier topic analyses on MIR-related discussions. For example, some of the most frequent words in topic #31 (e.g., "bibliographic," "metadata") are exactly the same as how users of MIR systems predominantly described their needs, particularly on music-related discussion platforms [1], [9]. Similarly, Audio Technology was one of the main topics revealed in this analysis, whereas "audio" has been one of the most commonly used title terms in ISMIR research topics [16]. With the rapid growth of audio-based research in the MIR community, insights and needs on audio technology from the music librarian community can provide important real-life use cases for MIR studies. Another example is Topic #42 which was labelled "Grove music online." It echoes the trend of online access to digital music information, which is also a major theme in MIR research community. As a major research-oriented online resource serving scholars and music professionals, Grove Music Online can be used by MIR researchers for improving MIR services and applications targeting the scholarly and professional user groups.

6.2 Topic Trend Analysis

The temporal trends of these topics are reported in the "trend column" in Table 2. Results of Cox Stuart tests show that six topics had increasing trends (#20, #31, #7, #49, #24, #27.) while four had decreasing trends (#41, #14, #12, and #2). Other topics had no significant trend. It is noteworthy that the topics with decreasing trends ranked higher in Table 2 than those with increasing trends, reflecting a phenomenon that popular topics became less dominating over time while topics with lower weights started gaining popularity in recent years, resulting in diversified topics.

The topics showing downward trends are related to the traditional functions of music libraries: #41 (Patron's requests and questions), #14 (Circulation and library policy in colleges), and #12 (Song requests). These indicate

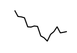
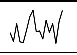
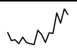
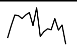
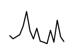
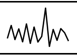
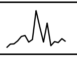
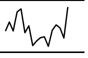
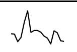
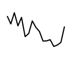
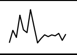
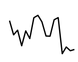
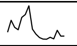
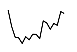
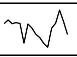
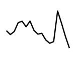
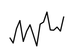
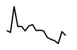
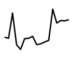
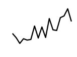
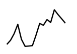
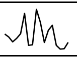
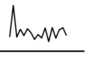
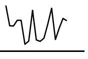
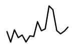
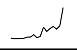
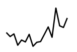
ID	Topic Label (weight)	Frequent words in Subjects	Top 10 Words	Trend	Trendline
41	Patron's request & question (0.086)	N/A	dear, copy, cw, patron, source, dear, cw, piece, score, advance, check	↓	
28	Musical terms (0.062)	Jazz, band, blues, rock music, songs	Jazz, band, record, blue, play, album, rock, live, john, sing	-	
4	MLA Event (0.054)	MLA meeting, conference, mentoring	mla, meeting, program, meet, conference, session, attend, pm, friday, time	-	
6	Cataloging (0.054)	oclc, lc, classification, cataloging, authority record, bib record	Record, title, oclc, number, catalogue, authority, catalog, add, authority record, item	-	
14	Circulation and library policy in colleges (0.046)	school, due date, except to faculty, circulate, Music Practice Room(s) in Library, students, faculty	student, collection, faculty, material, item, class, patron, circulate, score, staff	↓	
35	Classical music (0.045)	orchestra, chamber music, piano, concerto	piano, orchestra, symphony, violin, string, quartet, op, concerto, sonata, piece	-	
22	Music collection (0.039)	Music collection, catalogue, material, archive, donation	collection, manuscript, sheet, material, book, score, item, special, rare, project	-	
30	Scores, edition (0.037)	score, edition, need, actual music piece information (op, major, ..)	score, edition, publish, publisher, volume, copy, print, major, publication, complete	-	
47	Audio recording (0.034)	audio, recording, streaming, iPod, I-tunes, physical format, mp3	naxos, audio, recording, classical, stream, file, listen, digital, service, record	-	
12	Requesting songs (0.032)	Folk song for a wedding, Song in a Sopranos episode, Animal Songs, Songs about aging, Lyrics question,	song, lyric, sing, tune, word, folk, title, tina, popular, dallas	↓	
23	Journal and periodical (0.030)	journal, JSTOR, RIPM publications, ECO music journal, IIMP	journal, article, review, publish, issue, rilm, online, editor, title, publication	-	
39	Job posting (0.028)	Job opening, Job posting, position, job announcement	service, librarian, experience, collection, position, reference, application, degree, faculty, professional	-	
2	Music storage (0.028)	cd, lp, case, vinyl cd sleeves, cd box lids	cd, disc, lp, dvd, record, tape, label, case, box, booklet	↓	
20	MLA board, member (0.027)	MLA Newletter, roundtable, Note-Book, Call for new members, Board meeting, Board reports	mla, committee, member, board, association, report, chair, membership, year, annual	↑	
33	List of books (0.027)	Encylopedia, books, bibliography, Reference titles to give away, book titles	book, press, author, title, publication, york, year, isbn, publish, history	-	
0	Subject Heading, lc, genre, code (0.027)	call numbers(lc, Dewey), language code zxx, Genre heading, field, marc, aacr2, classification	subject, head, heading, term, title, instrument, score, code, form, musical	-	
29	Copyright (0.025)	Copyright question, copyright tips, copyright courses, royalties, purchasing a download, ILL(InterLibrary Loan)	copyright, copy, law, fair, public, license, domain, legal, public_domain, permission	-	
44	Journal and periodical (0.024)	Journal, issue, periodical	issue, journal, volume, spring, copy, fall, american, june, summer, july	-	
10	Conference roommate/transportation-mate solicitation (0.022)	roommates for, Shuttles to, registration	registration, conference, hotel, room, rate, register, tour, roommate, reservation, fee	-	
31	Metadata (0.020)	Music metadata, RDA, MOUG, Bibliographic Control Committee (BCC), ISBD, OCLC-MARC, music cataloging	catalogue, rda, bibliographic, metadata, marc, moug, oclc, service, indiana, access	↑	
7	Call for papers, proposals, awards, etc. (0.019)	Call for Papers, Call for Submissions, Call for Proposals, Call for Applications, Call for poster sessions, call for seminar topics	conference, proposal, paper, submission, session, presentation, submit, poster, deadline, topic	↑	
13	Travel information (0.019)	currency exchange, meeting, travel saving, transportation	san, travel, city, food, train, water, station, street, building, bus	-	
42	Grove music online (0.019)	grove, grove online, grove dictionary, new grove 2(ng2)	grove, online, article, dictionary, reference, print, oxford, grove_online, edition, resource	-	
15	Hymn (0.019)	Hymnals, hymn tune, chant, mass, choral music	church, organ, hymn, choral, saint, psalm, sing, choir, antoinette, mass	-	
49	Job posting (0.019)	Job posting	job, position, service, placement, librarian, apply, mla, placement_service, mla_placement, hire	↑	
24	CD/DVD sales (0.018)	CD HotList, Music Media Monthly, MLA Discount	order, sale, label, cd, offer, release, special, set, time, mla	↑	
27	Call for papers proposals, awards, etc. (0.014)	Call for Papers, Call for Submissions, Call for Proposals, Call for Applications, Call for poster sessions, call for seminar topics	letter, support, grant, application, travel, award, year, annual, meeting, moug	↑	

Table 2. Identified topics by topic modeling

that MLA members tended to engage more on other issues than these traditional library functions in recent years. This shift of attention may be attributed to advancement of technologies in recent years, including those in MIR. For instance, the downward trend of topic #12 (Requesting songs) might be related to the fact that music librarians and their clients have been equipped with alterative means to discover and access songs and other musical materials, such as search engines and online music repositories. Another trendline that may also reflect technology advancement is topic # 2 (music storage). The sharp decrease of this topic corresponds to the decline of CDs, LPs, tapes as formats of physical music materials. In this regard, technologies and resources facilitating music information retrieval and access are of great demand, particularly when the ways and channels people look for music information are changing so rapidly.

On the other hand, topics with upward trends include #20, which consists of messages about MLA board and members, demonstrating a vibrant and active professional association in the field of music librarianship. The second topic with a growing popularity is #31 (Metadata), which corroborates with recent developments in the metadata field such as RDA (Resource Description and Access, a standard for cataloging released in 2010). The other topics with upward trends all fall into the Advertisement category: #7 and #27 (Call for papers, proposals, awards), #49 (Job postings) and #24 (CD/DVD sales). This again reflects the flourishing development of the field and the community. In fact, the MLA and the U.S. branch of the International Association of Music Libraries (IAML-US) were merged in 2011, which has substantially boosted the status of MLA in the profession.[6]

6.3 Topics of Emails with/without Replies

We compared the email messages that stimulated discussion among participants of the MLA-L to those that did not. Figure 3 shows topics and their normalized weights in emails with and without replies respectively. Among the 27 topics identified by topic modeling, three pairs were merged for this analysis as the semantics of each pair are almost identical: #39 and #49 (Job postings), # 23 and #44 (Journals and periodicals) and #7 and #27 (Call for papers/proposals/awards). As shown in Figure 3, there is an overlap among highly-ranked topics between the two lists, such as "Cataloging", "Patron's request & questions", and "Classical music". This is not surprising as these are the most popular topics in the entire dataset (Table 2).

It is more interesting to see that there are topics ranked high in emails with replies but low in those without, such as "Subject heading, LC, genre, code" (#0), "Requesting songs" (#12), "Audio recording" (#47), and "Copyright" (#29)—indicating that emails in these topics often started discussions among subscribers. In particular, emails in "Requesting songs" (#12) and "Audio recording" (#47) are likely to contain music information needs and queries for music information. Similar to postings in Q&A websites, these email exchanges provide insights on 1) what kind of music information was needed by music librarians who in turn were trying to meet the needs of their patrons (i.e., the end users); and 2) how well-trained music information professionals looked for music information. Some of the heated discussions in threads with a large number of replies are likely to include queries that were interesting yet hard to find information for. These are excellent resources to discover not only new use cases but also search strategies for novel MIR systems.

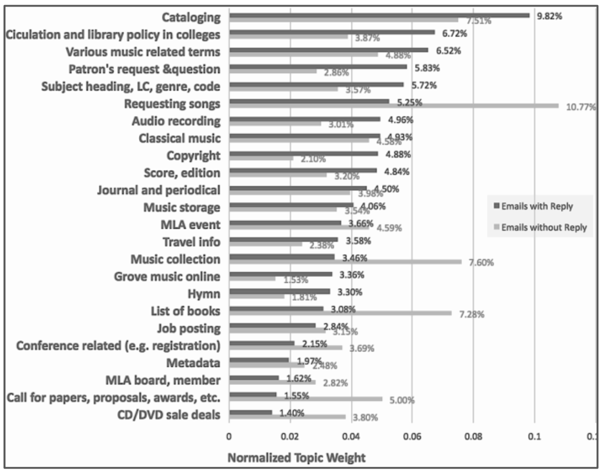


Figure 3. Topics in emails with and without replies.

Topics that are much more popular in emails without replies than in those with replies include "MLA event" (#4), "Call for papers/proposals/awards" (#7 and #27), "MLA board, members" (#20), and "CD/DVD sales" (#24). These topics are mostly of the nature of announcement and thus are unlikely to trigger discussions.

7. CONCLUSION

This study collected email messages posting in the Music Librarian Association mailing list (MLA-L) from 2000 to 2016, and analyzed the content through text mining. Main topics of discussions and their trends over the years are identified using Latent Dirichlet allocation (LDA). Twenty-seven meaningful topics were found and their semantics and trends were discussed in the context of MIR research. Topics in emails with and without replies were compared. As music librarians are gateways and bridges between music resources and users, the goals of music librarians and those of MIR researchers and practitioners are consistent: to help and facilitate users to access and make better use of music information. Therefore, the concerns and focuses reflected in the MLA-L are worthy of attention from the MIR community.

Future work will include detailed content analysis of the emails in such topics as "Requesting songs" and "Patron's Request & questions", to identify the needs of music information professionals and their users, and to learn about effective strategies of identifying and locating hard-to-find music information.

[6] https://www.musiclibraryassoc.org/?page=AboutMLA

8. ACKNOWLEDGEMENT

We thank the anonymous reviewers for their helpful comments. This study is partially supported by a seed research grant funded by the University of Hong Kong, and the Data-Driven Scholarship Autumn Institute jointly sponsored by the iSchool at the University of Illinois and the Faculty of Education at the University of Hong Kong.

9. REFERENCES

[1] D. Bainbridge et al., "How People Describe Their Music Information Needs: A Grounded Theory Analysis of Music Queries," in *Proc. of ISMIR*, 2003.

[2] A. Barua et al., "What Are Developers Talking About? An Analysis of Topics and Trends in Stack Overflow," *Empirical Software Eng.*, 19(3), pp. 619-654, 2014.

[3] D. M. Blei and J. D. Lafferty, "Dynamic Topic Models," in *Proc. of the 23rd Int. Conf. on Machine Learning*, Pittsburgh, PA, 2006, pp. 113-120.

[4] D. M. Blei et al., "Latent Dirichlet Allocation," *J. Mach. Learning Research*, vol. 3, pp. 993-1022, 2003.

[5] D. Campana, "Information Flow: Written Communication Among Music Librarians," *Notes*, ser. 2, 47(3), pp. 686-707, Mar. 1991.

[6] V. R. Carvalho and W. W. Cohen, "Learning to Extract Signature and Reply Lines from Email," in *Proc. Conf. Email and Anti-Spam*, 2004.

[7] K. Choi et al., "Topic Modeling Users' Interpretations of Songs to Inform Subject Access in Music Digital Libraries," in *Proc. of JCDL*, 2015, pp. 183-186.

[8] D. R. Cox and A. Stuart, "Some Quick Sign Tests for Trend in Location and Dispersion," *Biometrika*, 42(1/2), pp. 80-95, 1955.

[9] J. S. Downie and S. J. Cunningham, "Toward a Theory of Music Information Retrieval Queries: System Design Implications," in *Proc. of ISMIR*, 2002.

[10] M. M. Edwards, "A Content Analysis of the PUBYAC Discussion List," M.S. thesis, UNC, Chapel Hill, 1999.

[11] R. Griscom, Richard, "MLA-L at Twenty," *Notes*, vol. 65, no. 3, pp. 433-463, 2009.

[12] D. Hall et al., "Studying the History of Ideas Using Topic Models," in *Proc. Conf. Empirical Methods in Natural Language Processing*, 2008, pp. 363-371.

[13] X. Hu et al., "WiMIR: An Informetric Study on Women Authors In ISMIR," in *Proc. of ISMIR*, 2016, pp. 765-771.

[14] X. Hu et al., "A Framework for Evaluating Multimodal Music Mood Classification," *JASIST*, 68(2), pp. 273-285, 2017.

[15] X. Hu and J. S. Downie, "Improving Mood Classification in Music Digital Libraries by Combining Lyrics and Audio," in *Proc. of JCDL*, 2010, pp. 159-168.

[16] J. H. Lee, "Analysis of User Needs and Information Features in Natural Language Queries Seeking Music Information," *JASIST*, 61(5), pp. 1025-1045, 2010.

[17] J. H. Lee et al., "An Analysis of ISMIR Proceedings: Patterns of Authorship, Topic, and Citation," in *Proc. of ISMIR*, 2009, pp. 57-62.

[18] T. Martino. (2009). *Trend Analysis with the Cox-Stuart Test in R*. http://statistic-on-air.blogspot.com /2009/08/trend-analysis-with-cox-stuart-test-in. html

[19] A. K. McCallum. (2002). *Mallet: A machine Learning for Language Toolkit* [Online]. Available: http://mallet.cs.umass.edu

[20] A. McCallum et al., "The Author-Recipient-Topic Model for Topic and Role Discovery in Social Networks: Experiments with Enron and Academic Email," in *Workshop on Link Analysis, Counterterrorism and Security*, 2005, pp. 33.

[21] G. Mishne and N. Glance, "Leave a Reply: An Analysis of Weblog Comments," in *3rd Annual Workshop on the Weblogging Ecosystem*, 2006.

[22] R. Neumayer and A. Rauber, "Integration of text and audio features for genre classification in music information retrieval," in *Proc. of ECiR*, 2007, pp. 724-727.

[23] J. Riley and C. A. Mayer, "Ask a Librarian: The Role of Librarians in the Music Information Retrieval Community," in *Proc. of ISMIR*, 2006, pp. 13-18.

[24] M. Schedl et al., "Music Information Retrieval: Recent Developments and Applications," *Foundations and Trends® in Inform. Retrieval*, 8(2-3), pp. 127-261, 2014.

[25] C. Tan et al., "The Use of Bigrams to Enhance Text Categorization," *Inform. Process. & Manage.,* 38(4), pp. 529-546, 2002.

[26] L. Troutman, "MLA-L: A New mode of Communication," *Fontes Artis Musicae*, pp. 271-281, 1995.

[27] J. W. Uys et al., "Leveraging Unstructured Information Using Topic Modelling," in *Portland Int. Conf. Management of Engineering & Technology (PICMET)*, 2008, pp. 955-961.

FAST AND ACCURATE: IMPROVING A SIMPLE BEAT TRACKER WITH A SELECTIVELY-APPLIED DEEP BEAT IDENTIFICATION

Akira Maezawa
Yamaha Corporation

ABSTRACT

In music applications, audio beat tracking is a central component that requires both speed and accuracy, but a fast beat tracker typically has many beat phase errors, while an accurate one typically requires more computation. This paper achieves a fast tracking speed and a low beat phase error by applying a slow but accurate beat phase detector at only the most informative spots in a given song, and interpolating the rest by a fast tatum-level tracker. We present (1) a framework for selecting a small subset of the tatum indices that information-theoretically best describes the beat phases of the song, (2) a fast HMM-based beat tracker for tatum tracking, and (3) an accurate but slow beat detector using a deep neural network (DNN). The evaluations demonstrate that the proposed DNN beat phase detection halves the beat phase error of the HMM-based tracker and enables a 98% decrease in the required number of DNN invocations without dropping the accuracy.

1. INTRODUCTION

Offline audio beat tracking, the task of identifying beats in a music audio signal, is now a critical component in music applications, having uses in digital audio workstations, synthesizers, music recommendation and many others. In beat tracking, it is important to both estimate a reasonable tempo (inversely proportional to the period between two beats), as well as the timings of beat occurrence, or the beat phase. In these applications, a beat tracker must be fast and accurate for common types of musical pieces such as popular music and electronic dance music. The capability to analyze one song in a few seconds is often desirable in end-user products, while being satisfactorily accurate.

There is a trade-off, however, between the speed and the accuracy of a beat tracker. On the one hand, a fast beat tracker tends to make mistakes due to some simplifying assumptions. On the other hand, an accurate beat tracker that employs a more elaborate model like deep neural networks (DNN) tends to require more computation, which is proportional to the song duration.

We observe two points in these extrema. First, many of the errors in a fast tracker are attributed to incorrect beat *phase* estimation, especially for current popular music. For example, a simple beat tracker often mistakenly tracks the off-beat. This means that a simple method is already quite capable of tracking the tatum, *i.e.,* some integer subdivision of the beat, but poorly identifies *which* of them are the beats.

Second, using an accurate but slow beat tracker to sweep through an entire song is often wasteful. For many pieces where tatum tracking is possible with a simple method, the primary role of an elaborate method is in beat *phase* identification. In many musical pieces, however, the meter is mostly stationary, so the beat phase identification needs to be done only sparingly. If the beats are identified at the most informative spots in the music for beat phase identification, the rest may be interpolated by exploiting the stationarity of the meter.

In this paper, we combine the best of both worlds – a fast tracking of beats with a moderate amount of beat phase errors, and an accurate identification of the beats through the use of elaborate methods. Our key idea, as shown in Figure 1, is to efficiently fix the beat phase estimation errors of a simple but fast beat tracker, by *sparingly* applying an accurate but slow beat phase identifier, only at the most informative spots in the song. To elaborate, we (1) detect the tatum reliably with a simple beat tracker, (2) select a small disjoint subset of the tatums that best describes the beat phases of the entire song, (3) apply an elaborate beat

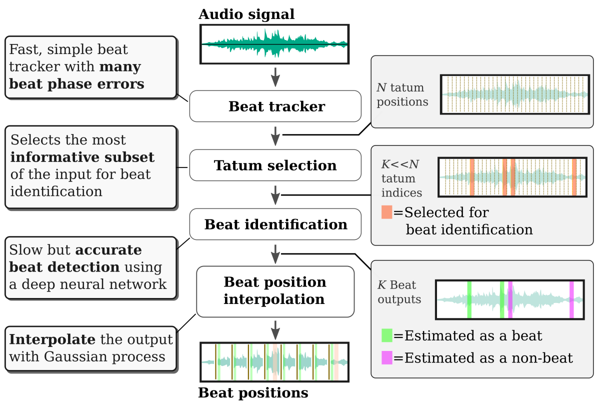

Audio signal
Fast, simple beat tracker with **many beat phase errors**
Beat tracker
N tatum positions
Selects the most **informative subset** of the input for beat identification
Tatum selection
K<<*N* tatum indices
🟥=Selected for beat identification
Beat identification
Slow but **accurate beat detection** using a deep neural network
K Beat outputs
Beat position interpolation
Interpolate the output with Gaussian process
🟩=Estimated as a beat
🟪=Estimated as a non-beat
Beat positions

Figure 1. The overview of our method. Our system improves a simple beat tracker with little computational overhead by adding a slow but accurate beat phase estimator. Computation is reduced by only using a subset of the detected tatums for phase estimation, and interpolating the estimated phase output.

identifier only at the selected tatum subset, and (4) interpolate the beat identification for the remaining tatums. Such a framework is enabled by exploiting a strong tatum-level correlation of the beat phase: it allows us to select a small set of tatums that best describes, in the sense of mutual information, the rest of the beats, and to interpolate the rest.

Our contributions are (1) a low-overhead framework for improving an existing simple beat tracker by cascading a more elaborate beat phase detector, achieved by identifying the most informative tatums in a song for beat phase identification; (2) a beat phase identification method using a DNN that accurately identifies the beat phase of a tatum-sliced data; and (3) a simple and fast HMM-based beat detector that jointly decodes the BPM and beat phase.

2. RELATED WORK

2.1 Beat Tracking and Downbeat Estimation

Beat tracking is the task of identifying the beats in a music audio signal, a task that has been studied extensively. Earlier methods match hand-crafted onset features while assuming the evolution of the tempo, through soft rules [9] dynamic programming [8] or hidden Markov models [20], often with an explicit tempo induction step [5, 6]. One of the key design issues is the choice of the hand-crafted features, such as changes of harmony [9] or variants of spectral flux [8, 13, 18], and features indicating the salient beat interval [13].

Beat phase error is a common failure mode in beat trackers [4, 5, 18]. For example, it is often common for a beat tracker to track half a beat behind an acceptable beat position, or to track the off-beats (*e.g.*, tracking second and fourth beats in a $4/4$ time) when tracking at half the underlying tempo. The frequency of such a failure mode occurs suggests that tatum tracking is relatively easily done with a simple and fast method, but identifying the beat within the detected tatums is a more delicate problem.

To estimate the beat phase, and more generally downbeats, modeling of the rhythmic patterns [9, 16], or extracting the features indicating the spectral change at multiple temporal level [13] have been shown to be useful. More recently, significant improvements in downbeat estimation have been achieved through the use of DNN, which delegates the delicate task of designing the relevant features to machine learning. For example, convolutional neural networks [7] or recurrent neural networks [2, 15] have shown significant improvements, at a cost of more computation.

2.2 Sensor Placement and Submodular Optimization

The core idea of our paper is to find the “best” tatum positions to apply the computationally-heavy DNN output so that the most information may be extracted with each invocation of the DNN. In a related problem of acquiring data with costly sensors, the problem of determining the “best” way to place each sensor as to get the most information out is known as the *sensor placement problem* [14]. Sensor placement problem is often tackled by exploiting the spatial correlation. For example, if the spatial distribution

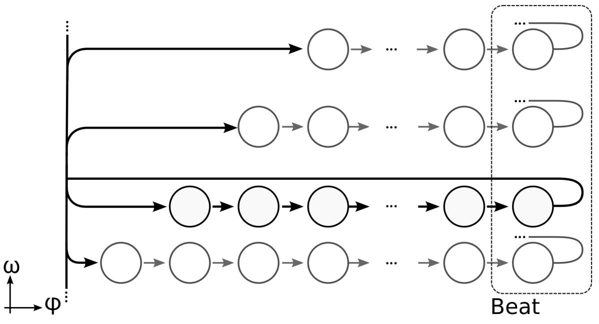


Figure 2. The state transition for the simple beat estimator. The beat phase counts down deterministically, and the next beat duration is chosen according to a beat period transition probability.

of the temperature needs to be acquired, it is better to place the temperature sensors far apart with than near each other, since the sensor readings at two nearby points, as opposed to far-away points, are more strongly correlated and thus are less revealing.

The sensor placement problem can be formulated as to maximize the mutual information between the placed sensors and some other points of interest for which the sensors are not placed. While this problem is NP-hard, the *submodularity* of the mutual information may be exploited to arrive at a greedy near-optimal algorithm [19]. Submodularity amounts to concavity for sets, and means that a given function f over a set A satisfies $f(X \cup \{i\}) \geq f(Y \cup \{i\})$ for all $X \subseteq Y \subseteq A$ and $i \in A \setminus Y$.

3. PROPOSED METHOD

Our method consists of (1) a simple tatum tracker that quickly tracks the tatum in an audio signal, (2) a slow but accurate beat identifier that identifies which of the tracked tatums are the beats, and (3) a tatum index selector, that selects a few tatum indices for applying the beat identifier, as to extract the most information regarding the presence of the beat.

3.1 Tracking Tatums with a Fast HMM Beat Tracker

We first track the tatum in a given audio signal. This is achieved by first using a simple beat tracker to extract the beat positions. Then, the tracked beat positions are subdivided equidistantly by a given factor Δn to obtain the tatums ($\Delta n = 4$ in this paper). Notice that while the beat detector may often track the wrong beat phase, it usually tracks the tatum properly.

To track the beat, we use an HMM-based beat detector, which uses onset and tempo features to jointly decode the beat position and the tempo. For the onset feature at frame t, we compute the first-order difference of the log-magnitude spectrum flux o_t, and for the BPM feature, we compute a comb filter-bank with the onset feature r_t, similar to [13].

We assume that the observed feature sequence is generated from an underlying sequence of discretized beat duration (related to tempo) ω, and a "count-down" timer indicating the number of feature frames until the next beat. We assume that each normalized ϕ is associated to a unique onset feature observation likelihood $p(o|\phi)$, and each value of the beat duration ω is associated with a unique tempo feature observation likelihood $p(r|\omega)$. Based on these assumptions, the observation likelihood is given as follows:

$$p(o_t, r_t|\phi_t, \omega_t) = p(o_t|\phi_t)p(r_t|\omega_t). \tag{1}$$

We choose $p(o|\phi)$ to be a Normal distribution whose mean and the variance are selected based on the value of ϕ normalized by the beat duration. We choose three sets of the mean and the variance, based on whether the normalized beat phase is 0, 0.5 and others; the parameters are trained with maximum likelihood. Furthermore, $p(r|\omega)$ is chosen to be a von Mises-Fisher distribution [1], whose parameters switches for each value of ω.

The time sequence of the beat duration ω_t and the count-down timer ϕ_t evolves such that (1) the ϕ_t decreases deterministically until reaching zero while ω_t remains constant and (2) when ϕ_t is zero, the beat duration ω_t switches to a new value according to a tempo transition matrix, and ϕ_{t+1} is set to ω_{t+1}. This amounts to the following generative process, also illustrated in Figure 2:

$$(\phi_t, \omega_t)|(\phi_{t-1}, \omega_{t-1}) \sim \begin{cases} \delta(\phi_t, \phi_{t-1} - 1)\delta(\omega_t, \omega_{t-1}) & \phi_{t-1} > 0 \\ R_{\omega_{t-1},\omega_t}\delta(\phi_t, \Phi_{\omega_t}) & \phi_{t-1} = 0 \end{cases}, \tag{2}$$

where Φ_ω is the number of audio frames corresponding to the beat duration for the beat duration ω, and R_{ω_1,ω_2} is a transition matrix that describes the probability of transitioning from beat duration ω_1 to ω_2. We reduce the search space by pruning negligible values of R and limiting the set of beat durations ω to consider, similar to [17]. The beat positions are decoded using the Viterbi algorithm to arrive at a set of N estimated *tatum* positions $\{\tau_n | n \in \mathcal{T} = \{1, 2 \ldots N\}\}$.

This model is quite similar to the bar pointer model [24], except (1) we apply the bar pointer model only to decode the beats and not the underlying meter or rhythm and (2) we use both the tempo and the onset likelihoods to decode the beats and the beat durations. The inference is more efficient compared to the bar-pointer model because the search space is much smaller – *i.e.*, the state space is at the beat level instead of the bar level, the beat duration is discretized, and the allowed transition is pruned.

Despite the efficient processing, this method, like many beat trackers, suffers from beat phase estimation errors, occurring approximately once every ten songs, for example, for songs with strong syncopations. Thus, we consider using a more elaborate beat phase detection that is capable of directly modeling the kind of long-term characteristics required for beat phase identification.

[1] The likelihood is given by $p(x; \mu, \kappa) \propto \exp(\kappa\mu^T x)$ for some μ, x in $(D-1)$-sphere, D being the dimension of the comb-filter output, and κ is a scalar parameter.

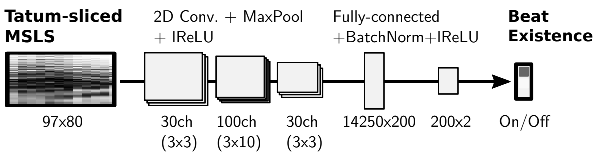


Figure 3. The architecture for predicting the Beat phase.

3.2 Identifying the Beats with Deep Neural Networks

To detect the presence of a beat at some tatum index, we use a DNN-based classifier of beats given tatum-sliced features. A DNN-based model is preferable because the notion of a beat depends on many factors like the rhythm and the harmony, and a manual feature design on such a problem is difficult.

To identify the beat position using a DNN, we use as the input the mel-scale log spectrogram (MSLS) that has been computed at each tatum. For each tatum, an 80-dimensional MSLS is extracted over a tatum window of 48 tatums before and after the current tatum, creating an input of $\mathbb{R}^{96\times 80}$.

The network consists of three convolutional layers, each with a leaky ReLU activation followed by max-pooling. The number of channels and the kernel sizes, in increasing order of layers, are 30, 100, 30 and (3×3), (3×10) and (3×3), respectively. It is followed by dropout regularization [23] during learning and a fully-connected layer with 200-dimensional output with a batch-normalization layer [11] and a leaky ReLU activation. Finally, a fully connected layer with a softmax activation extracts the posterior probability of the beat presence. Thus, for some tatum index n, the DNN outputs $b_n \in \{0, 1\}$, which is 1 if tatum index n is a beat and 0 otherwise. Note that the model has no recurrent connections, allowing a random access to the tatum index.

Given a ground-truth annotation of the beat presence $\hat{b}_n$, we minimize the cross-entropy loss $L(\Theta)$:

$$L(\Theta) = \sum_n \hat{b}_n \log b_n(\Theta) + (1 - \hat{b}_n)\log(1 - b_n(\Theta)), \tag{3}$$

with n indexed over the training dataset. The optimization is done stochastically, using ADAM [12] with weight decay regularization. The mini-batch is shuffled randomly and we augment the data by pitch-shifting the input audio by -7 to +7 semitones, similar to [22].

This model is similar to the network in [7] in that we also use tatum-level features, but we (1) use the full MSLS instead of a band-passed input, allowing simultaneous extraction of both harmonic and rhythmic features that contribute to beats, and (2) use both convolutional and fully connected layers.

3.2.1 On the Choice of the Tatum Window Size

To justify the use of MSLS evaluated over a windows of 48 tatums before and after the current tatum, we have tested the accuracy of our beat identification method when changing the window radius. The accuracy on the validation data by changing the number of *beats* (assuming 4 tatums per

L [beats]	1	2	4	8	10	12
Accuracy[%]	83	87	90	93	93	93

Table 1. Validation accuracy of the beat estimator when changing the tatum window radius L, in beats.

beat) is shown in Table 1. Since the performance saturates at 10 beats, use of 48 tatums provides sufficient performance. Such a change in accuracy shows that a long-term analysis spanning a few measures is indeed required for properly identifying the beat.

3.3 Choosing Tatums for Beat Identification

Since the identification of the beats from a single frame is prone to errors, it is typical to identify the beat presence b_n for all N frames to arrive at the final beat position estimate. However, identifying the beat using all the detected tatums is computationally expensive. Therefore we consider using a small subset of the detected tatums with $K \ll N$ elements for identifying the beats at the selected tatums and interpolating the rest.

Let us formalize the problem. Let b_n be the beat phase estimation at some tatum index n. b_n can be evaluated by invoking an accurate but computationally expensive beat phase estimator as was discussed. Let $\mathcal{T} = \{1, 2, \cdots N\}$ be the set of tatum indices. The ultimate goal is to obtain an *estimate* of b_n for all n in $\mathcal{T}$ reliably while evaluating b_n only at a few spots. Thus, the goal is to find some small subset $\mathcal{D}_K \subset \mathcal{T}$ of K elements for which we do invoke the beat phase estimation algorithm. For the remaining tatums $\bar{\mathcal{D}}_K$, we interpolate the beat phase estimate by evaluating the expectation of $b_{\bar{\mathcal{D}}_K}$ given $b_{\mathcal{D}_K}$.

To both identify the small subset $\mathcal{D}_K$ and interpolate the beat phase output, we exploit the strong tatum-level correlation of the beat identification output. To illustrate, Figure 4 shows the auto-correlation r_i of the beat identification output b_n. Notice that a non-negligible correlation exists not only nearby but also far away, up to about 100 tatums. This means that the existence of a beat at some tatum index provides information about the beat existence of the neighboring tatums.

To exploit such a covariance we assume that b_n is a Gaussian process [21]. That is, for some disjoint ordered sets of indices $\mathcal{D}, \mathcal{U} \subseteq \mathcal{T}$, the joint pdf of $b_{\mathcal{D}} = \{b_i | i \in \mathcal{D}\}$ and $b_{\mathcal{U}} = \{b_i | i \in \mathcal{U}\}$ is expressed as follows:

$$\begin{pmatrix} b_{\mathcal{D}} \\ b_{\mathcal{U}} \end{pmatrix} \sim \mathcal{N}\left(\mu, \begin{pmatrix} \Sigma_{\mathcal{D}\mathcal{D}} & \Sigma_{\mathcal{D}\mathcal{U}} \\ \Sigma_{\mathcal{U}\mathcal{D}} & \Sigma_{\mathcal{U}\mathcal{U}} \end{pmatrix}\right). \quad (4)$$

Here, Σ_{AB} is the cross-covariance matrix $\in \mathbb{R}^{|A| \times |B|}$ between $\{b_i | i \in A\}$ and $\{b_i | i \in B\}$, such that element (i, j) of $\Sigma_{A,B}$ is the covariance between the ith element of A and the jth element of B. μ is the expectation of b_i that is computed from a training dataset.

3.3.1 Index Selection for Beat Phase Identification

Assuming the underlying Gaussian process, we seek to identify a set of tatum indices $\mathcal{D}_K \in \mathcal{T}$ subject to $|\mathcal{D}_K| =$

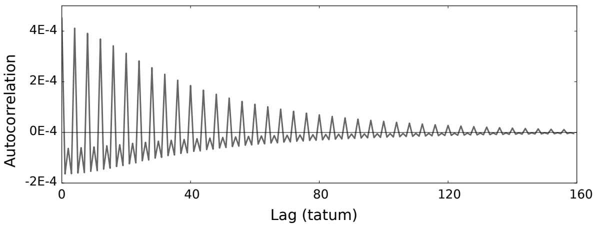


Figure 4. The auto-correlation of the estimated beat phase.

K, as to maximize the mutual information between beat phase output $b_{\mathcal{D}_K}$ and the unobserved beat phase output $b_{\bar{\mathcal{D}}_K}$. This problem is NP-hard [14], but thanks to the submodularity of mutual information, a near-optimal greedy algorithm exists [19]. To solve the problem with a greedy near-optimal algorithm, we iteratively add a new index i to the set of indices $\mathcal{D}_k$ that maximizes the increase in mutual information, *i.e.*, $\mathcal{D}_k = \mathcal{D}_{k-1} \cup \{i\}$ where:

$$i = \arg \max_{i' \in \bar{\mathcal{D}}_{k-1}} \mathrm{MI}(\mathcal{D}_{k-1} \cup \{i'\}) - \mathrm{MI}(\mathcal{D}_{k-1}), \quad (5)$$

where $\mathrm{MI}(\mathcal{D})$ denotes the mutual information between $\mathcal{D}$ and $\bar{\mathcal{D}}$. This can be seen as a special case of the sensor placement problem based on mutual information maximization, where we ignore any points for which the sensor may not be placed.

Equation 5 for a Gaussian process amounts to setting $i = \arg \max_{i' \in \bar{\mathcal{D}}_k} \delta_{k-1,i'}$ at step k, with the following $\delta_{i,k}$:

$$\delta_{k,i} = \frac{\Sigma_{i,i} - \Sigma_{\{i\},\mathcal{D}_k} \Sigma^{-1}_{\mathcal{D}_k,\mathcal{D}_k} \Sigma_{\mathcal{D}_k,\{i\}}}{\Sigma_{i,i} - \Sigma_{\{i\},\mathcal{C}_{ki}} \Sigma^{-1}_{\mathcal{C}_{ki},\mathcal{C}_{ki}} \Sigma_{\mathcal{C}_{ki},\{i\}}}, \quad (6)$$

where $\mathcal{C}_{ki} = \overline{\mathcal{D}_k \cup \{i\}}$. Here, the numerator amounts to the conditional variance of b_i given $b_{\mathcal{D}_k}$ and the denominator amounts to the conditional variance of b_i given the remaining elements. Intuitively, therefore, this objective seeks to find an index i that is unpredictable based on the already-observed data $b_{\mathcal{D}_k}$, while being representative of the non-selected indices, *i.e.*, easily predictable from seeing the data of the non-selected indices.

It is important to notice that the computation of Equation 6 is independent of the actual values of b_n. Thus, $\mathcal{D}_K$ can be evaluated *without invoking the computationally expensive DNN beat identifier*. Furthermore, given the auto-correlation computed beforehand, the set $\mathcal{D}_K$ depends only on the number of tatums in a given song and not the actual observations. Thus, $\mathcal{D}_K$ may be pre-computed for all practical values of the number of tatums, incurring zero runtime overhead.

3.3.2 Analysis of the Selected Indices

To see how the indices are chosen with our method, Figure 5 shows the index chosen as the index selection algorithm proceeds. Notice how in the initial stage ($K = 2$ and 4; red and yellow boxes), the algorithm selects the middle and the edges of the song while selecting pairs of indices that are 2 tatums apart, congruent mod 4. This shows that

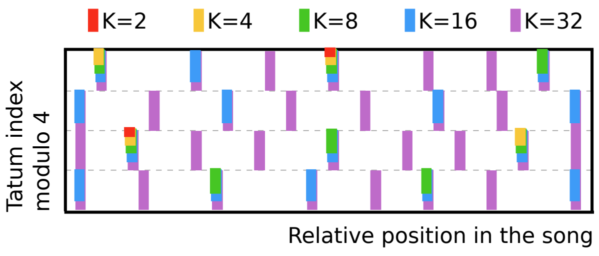


Figure 5. The indices chosen by the the mutual information maximization criterion as the algorithm progresses with $K = 2$, 4, up to 32 (best viewed in color).

the method tries to disambiguate the beat versus the off-beat, while staking out the entire song. As the algorithm progresses it selects tatum indices so that it (1) is more-or-less uniformly sampled throughout the song and (2) samples indices two tatums apart due to the weak correlation with lag 2 as was shown in Figure 4.

3.3.3 Interpolation of the Beat Phase Outputs

For interpolation, we evaluate the conditional expectation of $b_{\mathcal{T}}$ given $b_{\mathcal{D}_K}$:

$$\mathrm{E}[b_i|b_{\mathcal{D}_K}] = \Sigma_{i,\mathcal{D}_K}\Sigma^{-1}_{\mathcal{D}_K,\mathcal{D}_K} b_{\mathcal{D}_K}. \quad (7)$$

The evaluation of this function over all tatum indices requires a total of K invocations to the beat phase estimator.

Finally, given the interpolated beat phase estimates computed from Equation 7, we use in this paper a simple heuristic to decide the beat positions. For the given level of beat subdivision Δn used to compute the tatums, we find the beat phase given a beat subdivision. For each beat phase hypothesis ρ, we compute the following quantity:

$$R_\rho = \sum_{k=0}^{N/\Delta n} E[b_{k\Delta n+\rho}|b_{\mathcal{D}_K}] \quad (8)$$

Then, the beats are estimated as all tatum positions with indices $\hat{\rho} + k\Delta n$ with $k \in N$ and $\hat{\rho} = \arg\max_\rho R_\rho$. This heuristic is valid if the tatum tracking is successful and the number of tatums per beat remains fixed at Δn.

4. EXPERIMENTAL EVALUATION

4.1 Dataset

For the training and the validation dataset of the DNN beat identification and for estimating the auto-correlation of the beat identifier output at the tatum level, we used 100 popular songs from the RWC Popular Music Database [10]. For the test dataset, we prepared an in-house dataset consisting of 410 popular music in the United States and Japan. The median duration of the songs was four minutes. The tatum was extracted with the proposed method and the beat phase was hand-annotated based on a music score data created by professional musicians.

Method	Beat phase error	Real-time factor
BOCK15	14.0%	0.149
Baseline	12.2%	0.012
Proposed (Full)	6.1%	0.328
Proposed ($K = 4$)	16.2%	0.016
Proposed ($K = 8$)	10.1%	0.017
Proposed ($K = 16$)	7.6%	0.020
Proposed ($K = 32$)	6.1%	0.026

Table 2. The beat phase estimation error for songs that succeeded at tatum tracking, and the mean real-time factor.

4.2 Experiment 1: Beat Detection Improvement

First, we evaluated the beat phase estimation error of the proposed DNN beat identification method.

4.2.1 Experimental Condition

We extracted the beats using four methods: (1) an implementation [2] of the DNN-based beat detector in [3] with the tempo estimation method of [1], denoted "BOCK15," (2) the tatum detector used as a beat tracker (denoted "Baseline"), (3) the tatum detector with the DNN beat identifier evaluated over the entire data (denoted "Proposed (full)"), (4) the tatum detector with the DNN beat identifier evaluated over a subset of the data that has been selected with the proposed method (denoted "Proposed ($K = n$)" when using n tatum indices to evaluate the DNN). Since our focus is on fixing beat detection that succeeds at tatum tracking but fails at beat phase identification, we compared the methods for songs for which tatum extraction was successful (393 songs out of 410).

In addition, we computed the real-time factor, measured on a machine with Intel Core i5 processor running at 2.6 GHz with 4 GB of RAM with no GPU, utilizing one CPU core. Notice that the condition "Proposed (full)" is the baseline, in terms of computational speed, for most previous DNN-based downbeat detectors such as [7], as the previous method applies DNN to the entire audio data. The baseline method was implemented in C++ using SSE2 for SIMD instructions. The DNN was implemented in Python using the Chainer [3] library (an SSE2-optimized implementation of the DNN in C++ yielded in a similar benchmark and thus is omitted).

4.2.2 Results and Discussion

The results are shown in Table 2. It can be seen that by choosing $K = 32$, the method performs identically to when using the entire data for beat estimation, while being twelve times faster (38x faster than real-time). The modest increase in computation time over the Baseline suggests that there is only a marginal overhead, especially when the beat phase identification is executed in tandem with the beat extraction.

[2] Obtained from `https://github.com/CPJKU/madmom`, commit `de906fb`

[3] `https://github.com/chainer/chainer`

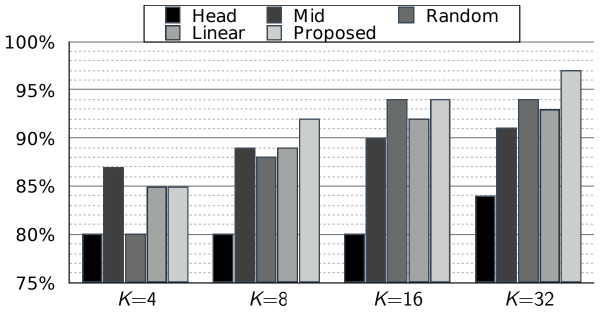


Figure 6. The match to the DNN beat detector when sampling K tatums in a song using different methods. Greater value indicates that the estimated beat phase better matches that estimated by the DNN.

To get a better idea of the computational costs, the beat detector takes 740 msec to process a minute of audio using a single core. The selection of indices incurred virtually no overhead, since we pre-computed the indices for all possible number of tatums. Note that this kind of index pre-computation is viable for $K = 32$; even when evaluating the indices for songs with 65000 tatums, the total memory for storing the indices is at most only 4 MB.

Each call to the DNN used about 100 msec. Of this, the convolutional layer comprised about 60% of the total time, and the fully-connected layer comprised about 40%. In our implementation, the convolution layer is performed fully for each call to the DNN, but this is redundant because portions of MSLS spectrograms may be shared across different audio frames. For songs with fewer than $K \times 96$ tatums (*e.g.,* song of six and a half minutes with a BPM of 120 and $K = 32$), there are redundant outputs of the convolutive layers. Such a redundancy potentially enables us to further speed up the convolutive layers.

4.3 Experiment 2: Tatum Selection Strategy

Second, we evaluated the capability for the proposed tatum index selection method to approximate the DNN evaluated over all tatums, by comparing the index selection method to other possible ways of selecting the tatum index subset.

4.3.1 Experimental Condition

For each song in the test dataset, including those for which tatum tracking had failed, we selected K indices with $K = [4, 8, 16, 32]$, with five different strategies: (1) select the first K tatums (denoted ”Head”), (2) select the middle K tatums (denoted ”Mid”), (3) select K uniformly-sampled tatums (denoted ”Random”), (4) select K linearly-spaced tatums (denoted ”Linear”), and (5) select K tatums with the proposed method. Then we evaluated, for each strategy and K, the agreement rate of the beat detection output, evaluated with Equation 8, between that obtained using (1) the indices obtained by each strategy and (2) all index. Notice that a high agreement for a given index selection scheme suggests its capability to approximate the beat identification using all indices.

4.3.2 Results and Discussion

Figure 6 shows the agreement rate between each tatum selection strategy and the full DNN.

When trying to identify the beat with only more than four tatums, the proposed method consistently outperformed the baselines. When choosing only four tatums ($K = 4$), the strategy of choosing the middle performs the best, perhaps because it is better to focus on one region to estimate the beat phase instead of dispersing the selection throughout the piece. The result nonetheless demonstrates the capability of our method to select a “good” set of indices for beat identifications compared to other intuitively-arrived methods.

Comparing the result with the previous experiment, with $K = 32$ the agreement of the proposed method does not reach 100% even though the beat detection accuracy for $K = 32$ is identical to those using the entire DNN output. This means that interpolated beats disagree for songs for which tatum tracking has failed. Such a disagreement occurs because (1) if the tatum tracking fails, the assumed covariance of the DNN output Σ poorly describes the underlying DNN output and (2) our beat position decoding method relies on a proper tracking of tatums with no change of meter.

5. CONCLUSION

This paper presented a method to improve a fast and simple beat tracker with little computatinal overhead by using an elaborate DNN-based beat identifier to fix the error in the simple beat tracker at carefully-selected tatums.

We addressed the critical issue of achieving both the accuracy enjoyed by a DNN-based beat identification of slow and elaborate methods, and the fast speed enjoyed by a simple but erroneous beat tracking methods. This was tackled by applying a DNN beat identification *sparingly*, only at the most informative tatum indices given by a simple beat-tracker. The selection was done as to maximize the mutual information between the selected and the non-selected indices for invoking the DNN.

Evaluation demonstrated that the DNN halved the beat phase error, and the tatum selection strategy provided the same performance as when sweeping through the entire audio signal, resulting in a twelve-fold speed improvement for a typical song. Furthermore, the subset selection method was also shown to be consistently efficient at approximating the DNN output compared to other index selection methods.

Future work includes (1) application of the index selection framework to other tasks in MIR such as downbeat estimation, (2) relaxing the assumptions made for the index selection, such as assuming a known and a fixed covariance of the output, (3) allowing the parameter K to be determined automatically, (4) improving the heuristics for deciding the beat positions.

6. REFERENCES

[1] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Accurate tempo estimation based on recurrent neural networks and resonating comb filters. In *Proc. International Conference on Music Information Retrieval*, pages 625–631, 2015.

[2] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. International Conference on Music Information Retrieval*, 2016.

[3] Sebastian Böck and Markus Schedl. Enhanced beat tracking with context-aware neural networks. In *Proc. International Conference on Digital Audio Effects*, pages 135–139, 2011.

[4] Matthew E.P. Davies, Norberto Degara, and Mark 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.

[5] Matthew E.P. Davies and Mark D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, 2007.

[6] Simon Dixon. Evaluation of the audio beat tracking system Beatroot. *Journal of New Music Research*, 36(1):39–50, 2007.

[7] Simon Durand, Juan P. Bello, Bertrand David, and Gaël Richard. Feature adapted convolutional neural networks for downbeat tracking. In *Proc. International Conference on Acoustics, Speech and Signal Processing*, pages 296–300, 2016.

[8] Daniel P.W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[9] Masataka Goto. An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, 30(2):159–171, 2001.

[10] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical, and jazz music databases. In *Proc. International Conference on Music Information Retrieval*, pages 287–288, 2002.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Anssi P. Klapuri, Antti J. Eronen, and Jaakko T. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):342–355, 2006.

[14] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[15] Florian Krebs, Sebastian Böck, Matthias Dorfer, and Gerhard Widmer. Downbeat tracking using beat-synchronous features and recurrent neural networks. In *Proc. International Conference on Music Information Retrieval*, pages 129–135, 2016.

[16] Florian Krebs, Sebastian Böck, and Gerhard Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proc. International Conference on Music Information Retrieval*, pages 227–232, 2013.

[17] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An efficient state-space model for joint tempo and meter tracking. In *Proc. International Conference on Music Information Retrieval*, pages 72–78, 2015.

[18] Jean Laroche. Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 51(4):226–233, 2003.

[19] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functionsi. *Mathematical Programming*, 14(1):265–294, 1978.

[20] Geoffroy Peeters and Helene Papadopoulos. Simultaneous beat and downbeat-tracking using a probabilistic framework: Theory and large-scale evaluation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1754–1769, 2011.

[21] Carl E. Rasmussen. Gaussian processes for machine learning. 2006.

[22] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proc. International Conference on Music Information Retrieval*, pages 121–126, 2015.

[23] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[24] Nick Whiteley, Ali Taylan Cemgil, and Simon J Godsill. Bayesian modelling of temporal structure in musical audio. In *Proc. International Conference on Music Information Retrieval*, pages 29–34, 2006.

FMA: A DATASET FOR MUSIC ANALYSIS

Michaël Defferrard[†] **Kirell Benzi**[†] **Pierre Vandergheynst**[†] **Xavier Bresson**[‡]
[†]LTS2, EPFL, Switzerland [‡]SCSE, NTU, Singapore Work conducted when XB was at EPFL.
{michael.defferrard,kirell.benzi,pierre.vandergheynst}@epfl.ch, xbresson@ntu.edu.sg

ABSTRACT

We introduce the Free Music Archive (FMA), an open and easily accessible dataset suitable for evaluating several tasks in MIR, a field concerned with browsing, searching, and organizing large music collections. The community's growing interest in feature and end-to-end learning is however restrained by the limited availability of large audio datasets. The FMA aims to overcome this hurdle by providing 917 GiB and 343 days of Creative Commons-licensed audio from 106,574 tracks from 16,341 artists and 14,854 albums, arranged in a hierarchical taxonomy of 161 genres. It provides full-length and high-quality audio, pre-computed features, together with track- and user-level metadata, tags, and free-form text such as biographies. We here describe the dataset and how it was created, propose a train/validation/test split and three subsets, discuss some suitable MIR tasks, and evaluate some baselines for genre recognition. Code, data, and usage examples are available at https://github.com/mdeff/fma.

1. INTRODUCTION

While the development of new mathematical models and algorithms to solve challenging real-world problems is obviously of first importance to any field of research, evaluation and comparison to the existing state-of-the-art is necessary for a technique to be widely adopted by research communities. Such tasks require open benchmark datasets to be reproducible. In computer vision, the community has developed established benchmark datasets such as MNIST [22], CIFAR [18], or ImageNet [4], which have proved essential to advance the field. The most celebrated example, the ILSVRC2012 challenge on an unprecedented ImageNet subset of 1.3M images [34], demonstrated the power of deep learning (DL), which won the competition with an 11% accuracy advantage over the second best [19], and enabled incredible achievements in both fields [21].

Unlike the wealth of available visual or textual content, the lack of a large, complete and easily available dataset for MIR has hindered research on data-heavy models such as DL. Table 1 lists the most common datasets used for content-based MIR. GTZAN [46], a collection of 1000 clips from 10 genres, was the first publicly available benchmark dataset for genre recognition (MGR). As a result, despites its flaws (mislabeling, repetitions, and distortions), it continues to be the most used dataset for MGR [43]. Moreover, it is small and misses metadata which e.g. prevents researchers to control for artists or album effects. Looking at Table 1, the well-known MagnaTagATune [20] and the Million Song Dataset (MSD) [3] as well as the newer AudioSet [10] and AcousticBrainz [32] appear as contenders for a large-scale reference dataset. MagnaTagATune, which was collected from the Magnatune label and tagged using the TagATune game, includes metadata, features and audio. The poor audio quality and limited number of songs does however limit its usage. MSD and AudioSet, while very large, force researchers to download audio clips from online services. AcousticBrainz's approach to the copyright issue is to ask the community to upload music descriptors of their tracks. Although it is the

dataset[1]	#clips	#artists	year	audio
RWC [12]	465	-	2001	yes
CAL500 [45]	500	500	2007	yes
Ballroom [13]	698	-	2004	yes
GTZAN [46]	1,000	$\sim$ 300	2002	yes
MusiClef [36]	1,355	218	2012	yes
Artist20 [7]	1,413	20	2007	yes
ISMIR2004	1,458	-	2004	yes
Homburg [15]	1,886	1,463	2005	yes
103-Artists [30]	2,445	103	2005	yes
Unique [41]	3,115	3,115	2010	yes
1517-Artists [40]	3,180	1,517	2008	yes
LMD [42]	3,227	-	2007	no
EBallroom [23]	4,180	-	2016	no[2]
USPOP [1]	8,752	400	2003	no
CAL10k [44]	10,271	4,597	2010	no
MagnaTagATune [20]	25,863[3]	230	2009	yes[4]
Codaich [28]	26,420	1,941	2006	no
FMA	**106,574**	**16,341**	**2017**	**yes**
OMRAS2 [24]	152,410	6,938	2009	no
MSD [3]	1,000,000	44,745	2011	no[2]
AudioSet [10]	2,084,320	-	2017	no[2]
AcousticBrainz [32]	2,524,739[5]	-	2017	no

[1] Names are clickable links to datasets' homepage.
[2] Audio not directly available, can be downloaded from ballroomdancers.com, 7digital.com, youtube.com.
[3] The 25,863 clips are cut from 5,405 songs.
[4] Low quality 16 kHz, 32 kbit/s, mono mp3.
[5] As of 2017-07-14, of which a subset has been linked to genre labels for the MediaEval 2017 genre task.

Table 1: Comparison between FMA and alternative datasets.

100% track_id	100% title	93% number
2% information	14% language_code	100% license
4% composer	1% publisher	1% lyricist
98% genres	98% genres_all	47% genre_top
100% duration	100% bit_rate	100% interest
100% #listens	2% #comments	61% #favorites
100% date_created	6% date_recorded	22% tags
100% album_id	100% title	
94% type	96% #tracks	
76% information	16% engineer	18% producer
97% #listens	12% #comments	38% #favorites
97% date_created	64% date_released	18% tags
100% artist_id	100% name	25% members
38% bio	5% associated_labels	
43% website	2% wikipedia_page	
	5% related_projects	
37% location	23% longitude	23% latitude
11% #comments	48% #favorites	10% tags[1]
99% date_created	8% active_year_begin	
	2% active_year_end	

[1] One of the tags is often the artist name. It has been subtracted.

Table 2: List of available per-track, per-album and per-artist metadata, i.e. the columns of `tracks.csv`. Percentages indicate coverage over all tracks, albums, and artists.

largest database to date, it will never distribute audio. On the other hand, the proposed dataset offers the following qualities, which in our view are essential for a reference benchmark.

Large scale. Large datasets are needed to avoid overtraining and to effectively learn models that incorporate the ambiguities and inconsistencies that one finds with musical categories. They are also more diverse and allows to average out annotation noise as well as characteristics who might be confounded with the ground truth and exploited by learning algorithms. While FMA features less clips than MSD or AudioSet, every other dataset with available quality audio are two orders of magnitude smaller (Table 1).

Permissive licensing. MIR research has historically suffered from the lack of publicly available benchmark datasets, which stem from the commercial interest in music by record labels, and therefore imposed rigid copyright. The FMA's solution is to aim for tracks which license permits redistribution. All data and code produced by ourselves are licensed under the CC BY 4.0 and MIT licenses.

Available audio. Table 1 shows that while the smaller datasets are usually distributed with audio, most of the larger do not. They either (i) only contain features derived from the audio, or (ii) provide links to download the audio from an online service.[1] The problem with (i) is that researchers are stuck with the chosen features and are prevented to leverage feature learning or end-to-end learning systems like DL. Moreover, we should be wary of proprietary features like those computed by commercial services such as Echonest. The problem with (ii) is that researchers have no control, i.e. we have no assurance that the files or services will not disappear or change without notice.

Quality audio. Distributed or downloadable audio are usually clips of 10 to 30 seconds and sometimes of low quality, e.g. 32 kbit/s for MagnaTagATune or an average of 104 kbit/s for MSD [37]. The problem with clips is that the beginning 30 seconds of tracks may yield different results than the middle or final 30 seconds, and that researchers may not have control over which part they get. In comparison, FMA comes with full-length and high-quality audio.

Metadata rich. The dataset comes with rich metadata, shown in Table 2. While not complete in any means, it compares favorably with the MSD which only provides artist-level metadata [3] or GTZAN which offers none.

Easily accessible. Working with the dataset only requires to download some `.zip` archives containing `.csv` metadata and `.mp3` audio. No need to crawl the web and circumvent rate limits or access restrictions. Besides, we provide some usage examples in the `usage.ipynb` Jupyter notebook to start using the data quickly.

Future proof and reproducible. All files and archives are checksummed and hosted in a long-term digital archive. Doing so alleviates the risks of songs to become unavailable. Moreover, we share all the code used to (i) collect the data, (ii) analyze it, (iii) generate the subsets and splits, (iv) compute the features and (v) test the baselines. The developed code can serve as a starting point for researchers to compute their own features or evaluate their methods. Finally, anybody can recreate or extend the collection, thanks to public songs and APIs.

Note that an alternative to open benchmarking is the approach taken by the MIREX evaluation challenges: the evaluation (by the organizers) of submitted algorithms on private datasets [6]. This practice however incurs an approximately linear cost with the number of submissions, which put the long-term sustainability of MIREX at risk [26]. By releasing this open dataset, we realize part of the vision of McFee *et al.* in "a distributed, community-centric paradigm for system evaluation, built upon the principles of openness, transparency, and reproducibility".

2. DATASET

2.1 The Free Music Archive

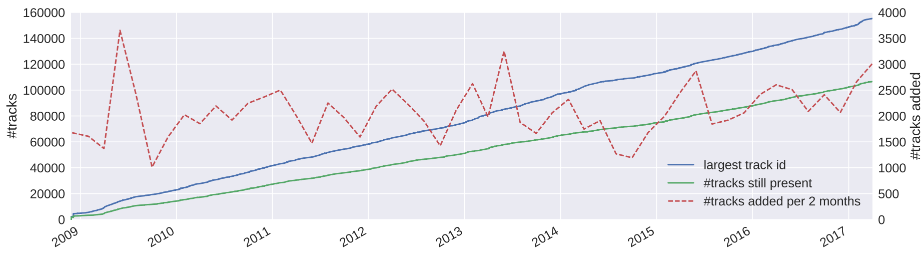
The dataset, both the audio and metadata, is a dump of the Free Music Archive, a free and open library directed by WFMU, the longest-running freeform radio station in the United States. Inspired by Creative Commons and the open-source software movement, the FMA provides a platform for curators, artists, and listeners to harness the potential of music sharing. The website provides a large catalog of artists and tracks, hand-picked by established audio curators. Each track is legally free to download as artists decided to release their works under permissive licenses. While there exists other sources of CC-licensed music, notably Jamendo, FMA is unique as it combines user-generated content with the curatorial role that WFMU and others have always played.[2]

2.2 Creation

As of April 1st 2017, when the dataset was gathered, the online archive largest track id was 155,320, of which

[1] Going to the source distributor is a way to adhere with copyright.

[2] Interview with Jason Sigal of the Free Music Archive, Rhizome.



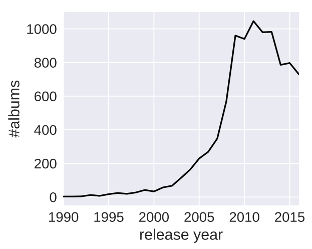


Figure 1: (left) Growth of the archive, created in 11/2008. (right) Number of albums released per year (min 1902, max 2017).

track						album			artist	
track_id	title	genres_all	genre_top	dur.	listens	title	listens	tags	name	location
150073	Welcome to Asia	[2, 79]	International	81	683	Reprise	4091	[world music, dubtronica, fusion]	DubRaJah	Russia
140943	Sleepless Nights	[322, 5]	Classical	246	1777	Creative Commons Vol. 7	28900	[classical, alternate, soundtrack, piano, ...	Dexter Britain	United Kingdom
64604	i dont want to die alone	[32, 38, 456]	Experimental	138	830	Summer Gut String	7408	[improvised, minimalist, noise, ...	Buildings and Mountains	Oneonta, NY
23500	A Life In A Day	[236, 286, 15]	Electronic	264	1149	A Life in a Day	6691	[idm, candlestick, romanian, candle, ...	Candlestickmaker	Romania
131150	Yeti-Bo-Betty	[25, 12, 85]	Rock	124	183	No Life After Crypts	3594	[richmond, fredericksburg, trash rock, ...	The Crypts!	Fredericksburg

Table 3: Some rows and columns of the metadata table, stored in `tracks.csv`.

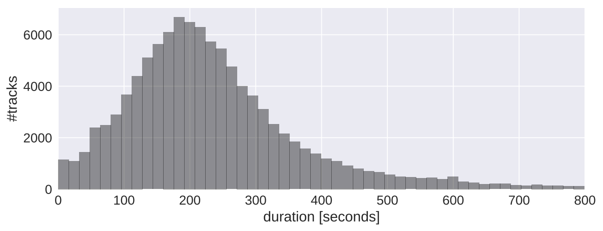


Figure 2: Track duration (min 0, max 3 hours).

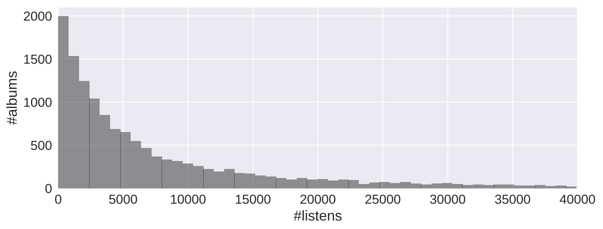


Figure 3: Album listens (min 0, max 3.6 millions).

109,727 were valid. The missing 45,594 ids probably correspond to deleted tracks. Figure 1 illustrates the growth of the dataset. In addition to per-track metadata, the used hierarchy of 161 genres and extended per-album (480 not found) and per-artist (250 not found) metadata were collected via the available API.[3] Finally, mp3 audio was downloaded over HTTPS. Out of all collected track ids, 180 mp3s could not be downloaded, 286 could not be trimmed by ffmpeg, and features could not be extracted from 71. Finally, the license of 2,616 tracks prohibited their redistribution, leaving us with 106,574 tracks.

While it may be argued that the dataset should be cleaned, we wanted it to resemble real world data. As such, we did not remove tracks which have too many genres, are too long, belong to rare genres, etc. Moreover, it is hard to set a threshold, algorithms shall handle outliers, and the small number of outliers will not impact performance much anyway. Researchers are obviously free to discard any track for training.

[3] See `webapi.ipynb` to query the API with our helpers.

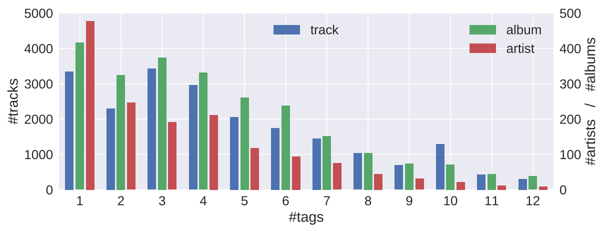


Figure 4: Per-track, album and artist tags (min 0, max 150).

2.3 Content

The collected metadata[4] was cleaned, uniformly formatted, merged and stored in `tracks.csv`[5] which Table 3 shows an excerpt. That file is a relational table where each row represents a track and columns are listed in Table 2. For ease of use, we kept all the metadata in a single table despite the redundancy incurred by the fact that all tracks from a given artist share all artist related columns. The problem is mitigated in practice by compression for storage and by *categorical variables* for memory usage.

All the metadata available through the API has been archived. It includes song title, album, artist, and per-track genres; user data such as per-track/album/artist favorites, play counts, and comments; free-form text such as per-track/album/artist tags, album description and artist biography. Coverage varies across fields and is reported in Table 2. Note that all that metadata has been produced by the artists when uploading their music and that while the content is curated, the curators focus on the musical content not the metadata. Figures 1, 2 and 3 show the distribution of albums per year, track durations, and play counts per album. See the `analysis.ipynb` notebook for a much more detailed analysis of the content.

The audio for each track is stored in a file which name is the track id. All tracks are mp3-encoded, most of them with sampling rate of 44,100 Hz, bit rate 320 kbit/s (263 kbit/s on average), and in stereo.

[4] `raw_tracks.csv`, `raw_albums.csv`, `raw_artists.csv`
[5] See `creation.ipynb` for the code which created the dataset.

id	parent	top_level	title	#tracks
38	None	38	Experimental	38,154
15	None	15	Electronic	34,413
12	None	12	Rock	32,923
1235	None	1235	Instrumental	14,938
25	12	12	Punk	9,261
89	25	12	Post-Punk	1,858
1	38	38	Avant-Garde	8,693

Table 4: An excerpt of the genre hierarchy, stored in `genres.csv`. Some of the 16 top-level genres appear in the top part, while some second- and third-level genres appear in the bottom part.

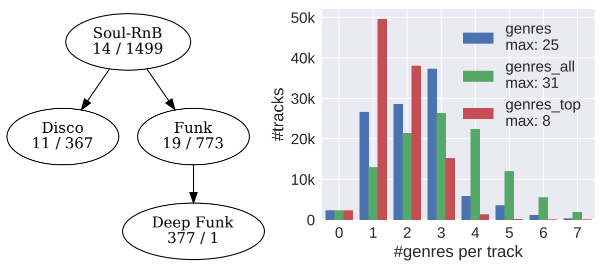


Figure 5: (left) Example of genre hierarchy for the top-level Soul-RnB genre. Left number is the `genre_id`, right is the number of tracks per genre. (right) Number of genres per track. A 3 genres limit has been introduced early on by the administrators.

2.4 Genres

The FMA is especially suited for MGR as it features fine genre information, i.e. multiple (sub-)genres associated to individual tracks, has a built-in genre hierarchy (Table 4), and is annotated by the artists themselves. While the artists are the best placed to judge the positioning of their creations, they might be inconsistent and motivated by factors not necessarily objective, such as achieving a higher play count. As labeling noise is unavoidable, those labels should ideally be one of many ground truths, to be complemented by crowd-sourcing and experts (from different music metadata websites).

While there is no agreement on a taxonomy of genres [35], we followed the hierarchy used by the archive, which is the one the authors had in mind when annotating their tracks. That hierarchy is composed of 161 genres of which 16 are roots, the others being sub-genres. Table 4 shows an excerpt of that information along with the number of tracks per genre and the associated top-level genre, that is the root of the genre tree. Figure 5 shows an excerpt of the tree.

In the per-track table, the `genres` column contain the genre ids indicated by the artist. Then, given such hierarchical information, we constructed a `genres_all` column which contains all the genres encountered when traversing the tree from the indicated genres to the roots. The root genres are stored in the `genres_top` column. Figure 5 and 6 shows the number of genres per track and tracks per genre.

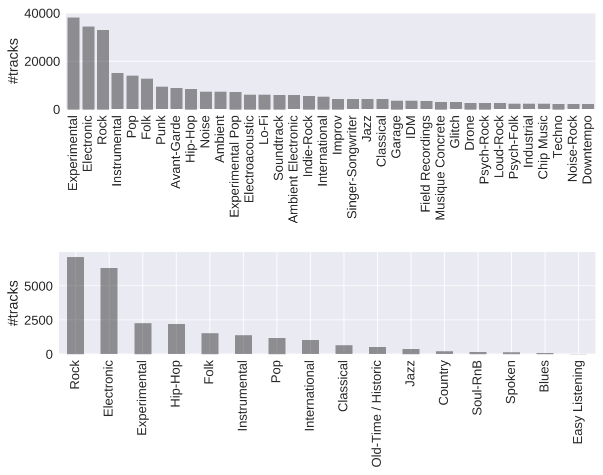


Figure 6: (top) Tracks per (sub-)genre on the full set (min 1, max 38,154). (bottom) Tracks per all 16 root genres on the medium subset (min 21, max 7,103). Note how experimental music is much less represented in the curated medium subset.

2.5 Features

To allow researchers to experiment without dealing with feature extraction, we pre-computed the features listed in Table 6. These are all the features the librosa Python library, version 0.5.0 [25], was able to extract. Each feature set (except zero-crossing rate) is computed on windows of 2048 samples spaced by hops of 512 samples. Seven statistics were then computed over all windows: the mean, standard deviation, skew, kurtosis, median, minimum and maximum. Those 518 pre-computed features are distributed in `features.csv` for all tracks.[6]

2.6 Subsets

For the dataset to be useful as a development set or for people with lower computational resources, we propose the following sets, each of which is a subset of the larger set:

1. **Full**: the complete dataset, described above. All 161 genres, unbalanced with 1 to 38,154 tracks per genre (Figure 6) and up to 31 genres per track (Figure 5).
2. **Large**: the full dataset with audio limited to 30 seconds clips extracted from the middle of the tracks (or entire track if shorter than 30 seconds). That trimming reduces the size of the data by a factor 10.
3. **Medium**: while root genre recognition should be treated as a multi-label problem in general, we constructed this subset for the simpler problem of single-label prediction. It makes sense as half the tracks have a single root genre (Figure 5). As such, we selected those tracks with only one top genre and sampled the clips according to the completeness of their metadata and their popularity, hoping to select tracks of higher quality. That selection left us with 25,000 30s clips, genre unbalanced with 21 to 7,103 clips per top genre (Figure 6), but only one of the 16 top genres per clip.

[6] See `features.py` for the code which computed the features.

4. **Small**: to construct a balanced subset, we selected with the same process the top 1,000 clips from the 8 most popular genres of the medium set. The subset is thus composed of 8,000 30s clips from 8 top genres, balanced with 1,000 clips per genre, 1 root genre per clip. This subset is similar to the very popular GTZAN [46] with the added benefits of the FMA, that is metadata, pre-computed features, and copyright-free audio.

Table 5 highlights the main differentiating factors between the proposed subsets.

2.7 Splits

We propose an 80/10/10% split into training, validation and test sets to make research on the FMA reproducible. Training and validation shall be merged if cross-validation is used instead. Below are the followed constraints:

1. Stratified sampling to preserve the percentage of tracks per genre (important for minority genres). Each root genre is guaranteed to be represented in all splits, but the ratio is only exact for the small subset (800/100/100). The seven smallest sub-genres (less than 20 tracks in total) are however not guaranteed to appear in all splits of the full and large sets.
2. An artist filter for artists to be part of one set only, thus avoiding any artist and album effect. It has been shown that the use of songs from the same artist in both training and test sets leads to over-optimistic accuracy and may favor some approaches [8, 29].

The above constraints are satisfied for all subsets, and a track is assigned to the same split across all of them.[5] The 2,231 tracks without genre label are assigned to the training set (full and large sets) as they might be useful as additional training samples for semi-supervised algorithms.

3. USAGE

With its rich set of metadata, user data, audio and features, the FMA is amenable to many tasks in MIR. We share below some possible uses which serve to illustrate the breadth of data available in the dataset.

3.1 Music Classification and Annotation

Music classification is a key problem in MIR with many potential applications. For one, a classification system enables end users to search for the types of music they are interested in. On the other hand, different music types are managed more effectively and efficiently once they are categorized into different groups [9]. The classification tasks which can readily be evaluated on FMA include genre recognition, artist identification, year prediction, and automatic tagging. Automatic tagging [2] is a classification problem which covers different semantic categories, where tags are labels which can be any musical term that describes the genre, mood, instrumentation, and style of the song. It helps to convert the music retrieval problem to text retrieval by substituting songs with tags. In addition to supervised methods which classify music given an arbitrary taxonomy, another approach is to cluster data in an unsupervised way so that a categorization will emerge from the data itself based on objective similarity measures. Then, does genre or another taxonomy naturally come up?

dataset	clips	genres	length	size	
			[s]	[GiB]	#days
small	8,000	8	30	7.4	2.8
medium	25,000	16	30	23	8.7
large	106,574	161	30	98	37
full	106,574	161	278	917	343

Table 5: Proposed subsets of the FMA.

feature set	dim.	LR	kNN	SVM	MLP
1 Chroma [11]	84	44	44	48	49
2 Tonnetz [14]	42	40	37	42	41
3 MFCC [33]	140	58	55	61	53
4 Spec. centroid	7	42	45	46	48
5 Spec. bandwidth	7	41	45	44	45
6 Spec. contrast [17]	49	51	50	54	53
7 Spec. rolloff	7	42	46	48	48
8 RMS energy	7	37	39	39	39
9 Zero-crossing rate	7	42	45	45	46
3 + 6	189	60	55	63	54
3 + 6 + 4	273	60	55	63	53
1 to 9	518	61	52	63	58

Table 6: Test set accuracies of various features and classifiers for top genre recognition on the medium subset.

3.2 Genre Recognition

Music genres are categories that have arisen through a complex interplay of cultures, artists, and market forces to characterize similarities between compositions and organize music collections. Yet, the boundaries between genres still remain fuzzy, making the problem of music genre recognition (MGR) a nontrivial task [35]. While its utility has been debated, mostly because of its ambiguity and cultural definition, it is widely used and understood by end-users who find it useful to discuss musical categories [27]. As such, it is one of the most researched areas of MIR. We propose the following prediction problems of increasing difficulty:

1. Single top genre on the balanced small subset.
2. Single top genre on the unbalanced medium subset.
3. Multiple top genres on the large / full set.
4. Multiple (sub-)genres on the large / full set.

Table 6 reports accuracies for problem 2 with nine mainstream feature sets and some combinations as well as four standard classifiers using scikit-learn, version 0.18.1 [31]. Specifically, we employed linear regression (LR) with an L^2 penalty, k-nearest neighbors (kNN) with $k = 200$, support vector machines (SVM) with a radial basis function (RBF) kernel and a multilayer perceptron (MLP) with 100 hidden neurons. All classifiers were tested with otherwise default settings.[7] Reported performance should not be taken as the state-of-the-art but rather as

[7] See `baselines.ipynb` for all details.

a lower-bound and an indication of the task's difficulty. Moreover, the developed code can serve as a reference and is easily modified to accommodate other features and classifiers.

A major motivation to construct this dataset was to enable the use of the powerful DL set of techniques to music analysis, an hypothesized cause of stagnation on MIREX tasks [38]. With availability of audio, DL architectures such as convolutional neural networks and recurrent neural networks can be applied to the waveform to avoid any feature engineering. While those approaches have fallen behind learning from higher-level representations such as spectrograms [5], a greater exploration of the design space will hopefully provide alternatives to solving MIR challenges [16].

3.3 Data Analysis

While our intention was to release a large volume of audio for machine learning algorithms, analyzing audio is certainly of interest to musicologists and researchers who want to study relations with higher-level representations. Moreover, the availability of complete tracks allows proper study of music properties, for example music structure analysis. Finally, the metadata is surely a valuable addition to existing datasets (e.g. MusicBrainz, AllMusic, Discogs, Last.fm) for metadata analysis.

4. DISCUSSION

While the FMA can be used to evaluate many tasks, metadata is missing for e.g. mood classification or instrument recognition. However, a more thorough investigation of the available tags may reveal their feasibilities. Similarly, cover song detection may be doable if multiple versions of many songs are featured. While the present dump only captures listening and downloading counts in aggregates, [8] the lists of which songs, albums and artists a user marked as favorites or commented are public, as well as *user mixes*. While not public, listening and downloading activities are logged and might be shared after anonymization. [9] Moreover, users form a public social network via *friend requests*. Collecting this information would open the possibility of a large-scale evaluation of content-based recommender systems. Cover images for tracks, albums, and artists are another public asset which may be of interest. Finally, we can expect the dataset to be cross-referenced with other resources to unlock additional tasks, as has happened for example with the MSD and AllMusic, last.fm and beaTunes for genre recognition [37, 39], musixmatch for lyrics, SecondHandSongs for cover songs, or This Is My Jam for user play counts.

Diversity is another issue. As suggested by Figure 6, this collection is biased toward experimental, electronic, and rock music. Moreover, it does not contain mainstream music and few commercially successful artists. A common criticism of basing research on CC-licensed music is that the music is of substantially lower "quality". Moreover, it is unknown whether datasets made up of mainstream or non-mainstream music have similar properties and if algorithms tailored on one perform similarly on the other. While those points are valid for high-level tasks such as recommendation (which depend on a variety of factors beyond the acoustic content), this is a much more tenuous case for the majority of tasks, in particular perceptual tasks. Nevertheless, algorithms should ideally be evaluated on multiple datasets, which will help answer such questions.

[8] That information can be useful to e.g. analyze and predict hits.

[9] Private discussion with the website administrators.

5. CONCLUSION AND PERSPECTIVES

Benchmarking is an important aspect in experimental sciences — results reported by individual research groups need to be comparable. Important aspects of these are datasets that can be easily shared among researchers, together with a set of defined tasks and splits. The FMA enables researchers to test algorithms on a large-scale collection, closer to real-world environments. Even though it is still two orders of magnitude behind commercial services who have access to tens of millions of tracks, [10] it is of the same scale as the largest image dataset which opened the door to dramatic performance improvements for many tasks in computer vision. By providing audio, we do not limit the benchmarking to pre-computed features and allow scientists to develop and test new feature sets, learn features, or learn mappings directly from the audio. For now, music classification, and MGR in particular, is the most straightforward use case for FMA. The inclusion of a genre hierarchy makes it specially interesting, as it offers possibilities rarely found in alternative collections.

In addition to the proposed usage and many others people will find, future work on the dataset itself should focus on (i) validating the ground truth by measuring agreement by independent annotators and (ii) obtaining additional metadata and labels. If the community finds interest in the dataset and validate its use, that can be achieved by scraping the website for information not available through the API, cross-referencing with other resources, or crowdsourcing (with e.g. Mechanical Turk or CrowdFlower).

In a post about the dataset, Cheyenne Hohman, the Director at the Archive, wrote that "by embracing the ... philosophy of Creative Commons, artists are not only making their music available for the public to listen to, but also for educational and research applications". Let's hope for a future where sharing is first and researchers feed open platforms with algorithms while they feed us with data.

6. ACKNOWLEDGMENTS

We want to thank the team supporting the Free Music Archive as well as all the contributing artists and curators for the fantastic content they made available. We want to thank the anonymous ISMIR reviewers for their thorough reviews and many constructive comments which have improved the quality of this work. We are grateful to

[10] 37M Echonest, 30M Spotify, 45M last.fm, 45M 7digital, 26M iTunes

SWITCH and EPFL for hosting the dataset within the context of the SCALE-UP project, funded in part by the swissuniversities SUC P-2 program. Xavier Bresson is supported by NRF Fellowship NRFF2017-10.

7. REFERENCES

[1] A Berenzweig, B Logan, D PW Ellis, and B Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music J.*, 2004.

[2] T Bertin-Mahieux, D Eck, and M Mandel. Automatic tagging of audio: The state-of-the-art. *Machine Audition: Principles, Algorithms and Systems*, 2010.

[3] T Bertin-Mahieux, D PW Ellis, B Whitman, and P Lamere. The million song dataset. In *ISMIR*, 2011.

[4] J Deng, W Dong, R Socher, LJ Li, K Li, and L Fei-Fei. Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition*, 2009.

[5] S Dieleman and B Schrauwen. End-to-end learning for music audio. In *ICASSP*, 2014.

[6] J Downie, A Ehmann, M Bay, and M Jones. The music information retrieval evaluation exchange: Some observations and insights. *Advances in Music Information Retrieval*, 2010.

[7] D PW Ellis. Classifying music audio with timbral and chroma features. In *ISMIR*, 2007.

[8] A Flexer. A closer look on artist filters for musical genre classification. In *ISMIR*, 2007.

[9] Z Fu, G Lu, K M Ting, and D Zhang. A survey of audio-based music classification and annotation. *IEEE Trans. on Multimedia*, 2011.

[10] J F Gemmeke, D PW Ellis, D Freedman, A Jansen, W Lawrence, R C Moore, M Plakal, and M Ritter. Audio set: An ontology and human-labeled dartaset for audio events. In *ICASSP*, 2017.

[11] M Goto. A chorus section detection method for musical audio signals and its application to a music listening station. *IEEE Trans. on Audio, Speech, and Language Processing*, 2006.

[12] M Goto, H Hashiguchi, T Nishimura, and R Oka. Rwc music database: Popular, classical and jazz music databases. In *ISMIR*, 2002.

[13] F Gouyon, S Dixon, E Pampalk, and G Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proc. of the AES 25th Int. Conf.*, 2004.

[14] C Harte, M Sandler, and M Gasser. Detecting harmonic change in musical audio. In *In Proc. of Audio and Music Computing for Multimedia Workshop*, 2006.

[15] H Homburg, I Mierswa, B Möller, K Morik, and M Wurst. A benchmark dataset for audio classification and clustering. In *ISMIR*, 2005.

[16] E J Humphrey, Juan P Bello, and Y LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *ISMIR*, 2012.

[17] DN Jiang, L Lu, HJ Zhang, JH Tao, and LH Cai. Music type classification by spectral contrast feature. In *IEEE Int. Conf. on Multimedia and Expo*, 2002.

[18] A Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009.

[19] A Krizhevsky, I Sutskever, and G E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[20] 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.

[21] Y LeCun, Y Bengio, and G Hinton. Deep learning. *Nature*, 2015.

[22] Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. In *Proc. of the IEEE*, 1998.

[23] U Marchand and G Peeters. The extended ballroom dataset. 2016.

[24] M Mauch, C Cannam, M Davies, S Dixon, C Harte, S Kolozali, D Tidhar, and M Sandler. Omras2 metadata project 2009. In *ISMIR*, 2009.

[25] B McFee et al. librosa 0.5.0, 2017.

[26] B McFee, E J Humphrey, and J Urbano. A plan for sustainable mir evaluation. In *ISMIR*, 2016.

[27] C McKay and I Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR*, 2006.

[28] C McKay, D McEnnis, and I Fujinaga. A large publicly accassible prototype audio database for music research. In *ISMIR*, 2006.

[29] E Pampalk, A Flexer, and G Widmer. Improvements of audio-based music similarity and genre classificaton. In *ISMIR*, 2005.

[30] E Pampalk, A Flexer, G Widmer, et al. Improvements of audio-based music similarity and genre classificaton. In *ISMIR*, 2005.

[31] F Pedregosa et al. Scikit-learn: Machine learning in python. *J. of Machine Learning Research*, 2011.

[32] A Porter, D Bogdanov, R Kaye, R Tsukanov, and X Serra. Acousticbrainz: a community platform for gathering music information obtained from audio. In *ISMIR*, 2015.

[33] L R Rabiner and BH Juang. *Fundamentals of speech recognition*. 1993.

[34] O Russakovsky et al. Imagenet large scale visual recognition challenge. *Int. J. of Computer Vision*, 2015.

[35] N Scaringella, G Zoia, and D Mlynek. Automatic genre classification of music content: a survey. *IEEE Signal Processing Magazine*, 2006.

[36] M Schedl, N Orio, C Liem, and G Peeters. A professionally annotated and enriched multimodal data set on popular music. In *Proc. of the ACM Multimedia Systems Conference*, 2013.

[37] A Schindler, R Mayer, and A Rauber. Facilitating comprehensive benchmarking experiments on the million song dataset. In *ISMIR*, 2012.

[38] R Scholz, G Ramalho, and G Cabral. Cross task study on mirex recent results: An index for evolution measurement and some stagnation hypotheses. In *ISMIR*, 2016.

[39] H Schreiber. Improving genre annotations for the million song dataset. In *ISMIR*, 2015.

[40] K Seyerlehner, G Widmer, and P Knees. Frame level audio similarity - a codebook approach. In *Proc. of the Int. Conf. on Digital Audio Effects*, 2008.

[41] K Seyerlehner, G Widmer, and T Pohle. Fusing block-level features for music similarity estimation. In *Proc. of the Int. Conf. on Digital Audio Effects*, 2010.

[42] C N Silla Jr, A L Koerich, and C AA Kaestner. The latin music database. In *ISMIR*, 2008.

[43] B L Sturm. The state of the art ten years after a state of the art: Future research in music information retrieval. *J. of New Music Research*, 2014.

[44] D Tingle, Y E Kim, and D Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proc. of the Int. Conf. on Multimedia Information Retrieval*, 2010.

[45] D Turnbull, L Barrington, D Torres, and G Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Trans. on Audio, Speech, and Language Processing*, 2008.

[46] G Tzanetakis and P Cook. Musical genre classification of audio signals. *IEEE Trans. on Speech and Audio Processing*, 2002.

MIDINET: A CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORK FOR SYMBOLIC-DOMAIN MUSIC GENERATION

Li-Chia Yang, Szu-Yu Chou, Yi-Hsuan Yang
Research Center for IT innovation, Academia Sinica, Taipei, Taiwan
{richard40148, fearofchou, yang}@citi.sinica.edu.tw

ABSTRACT

Most existing neural network models for music generation use recurrent neural networks. However, the recent WaveNet model proposed by DeepMind shows that convolutional neural networks (CNNs) can also generate realistic musical waveforms in the audio domain. Following this light, we investigate using CNNs for generating melody (a series of MIDI notes) one bar after another in the symbolic domain. In addition to the generator, we use a discriminator to learn the distributions of melodies, making it a generative adversarial network (GAN). Moreover, we propose a novel conditional mechanism to exploit available prior knowledge, so that the model can generate melodies either from scratch, by following a chord sequence, or by conditioning on the melody of previous bars (e.g. a priming melody), among other possibilities. The resulting model, named MidiNet, can be expanded to generate music with multiple MIDI channels (i.e. tracks). We conduct a user study to compare the melody of eight-bar long generated by MidiNet and by Google's MelodyRNN models, each time using the same priming melody. Result shows that MidiNet performs comparably with MelodyRNN models in being realistic and pleasant to listen to, yet MidiNet's melodies are reported to be much more interesting.

1. INTRODUCTION

Algorithmic composition is not a new idea. The first computational model for algorithmic composition dates back to 1959 [16], according to the survey of Papadopoulos and Wiggins [23]. People have also used (shallow) neural networks for music generation since 1989 [30]. It was, however, only until recent years when deep neural networks demonstrated their ability in learning from big data collections that generating music by neural networks became a trending topic. Lots of deep neural network models for music generation have been proposed just over the past two years [4, 7, 10, 15, 18, 19, 21, 22, 26, 28, 31, 33].

The majority of existing neural network models for music generation use recurrent neural networks (RNNs) and their variants, presumably for music generation is inherently about generating sequences [2, 3, 9, 14]. These models differ in the model assumptions and the way musical events are represented and predicted, but they all use information from the previous events to condition the generation of the present one. Famous examples include the MelodyRNN models [33] for *symbolic-domain* generation (i.e. generating MIDIs) and the SampleRNN model [19] for *audio-domain* generation (i.e. generating WAVs).

Relatively fewer attempts have been made to use deep convolutional neural networks (CNNs) for music generation. A notable exception is the WaveNet model [31] proposed recently for audio-domain generation. It generates one audio sample at a time, with the predictive distribution for each sample conditioned on previous samples through dilated causal convolutions [31]. WaveNet shows it possible for CNNs to generate realistic music. This is encouraging, as CNNs are typically faster to train and more easily parallelizable than RNNs [32].

Following this light, we investigate in this paper a novel CNN-based model for symbolic-domain generation, focusing on melody generation. [1] Instead of creating a melody sequence continuously, we propose to generate melodies *one bar (measure) after another*, in a successive manner. This allows us to employ convolutions on a 2-D matrix representing the presence of notes over different time steps in a bar. We can have such a score-like representation for each bar for either a real or a generated MIDI.

Moreover, to emulate *creativity* [23] and encourage diverse generation result, we use random noises as input to our *generator* CNN. The goal of the generator is to transform random noises into the aforementioned 2-D score-like representation, that "appears" to be from real MIDI. This transformation is achieved by a special convolution operator called transposed convolution [8]. Meanwhile, we learn a *discriminator* CNN that takes as input a 2-D score-like representation and predicts whether it is from a real or a generated MIDI, thereby informing the generator how to appear to be real. This amounts to a generative adversarial network (GAN) [11–13, 24, 27], which learns the generator and discriminator iteratively under the concept of minimax two-player game theory.

This GAN alone does not take into account the temporal dependencies across different bars. To address this issue, we propose a novel conditional mechanism to use

[1] In general, a melody may be defined as a succession of (monophonic) musical notes expressing a particular musical idea.

	MelodyRNN [33]	Song from PI [7]	DeepBach [15]	C-RNN-GAN [21]	MidiNet (this paper)	WaveNet [31]
core model	RNN	RNN	RNN	RNN	CNN	CNN
data type	symbolic	symbolic	symbolic	symbolic	symbolic	audio
genre specificity	—	—	Bach chorale	—	—	—
mandatory prior knowledge	priming melody	music scale & melody profile	—	—	—	priming wave
follow a priming melody	√	√			√	√
follow a chord sequence					√	
generate multi-track music		√	√		√	√
use GAN				√	√	
use versatile conditions					√	
open source code	√			√	√	

Table 1. Comparison between recent neural network based music generation models

music from the previous bars to condition the generation of the present bar. This is achieved by learning another CNN model, which we call the *conditioner* CNN, to incorporate information from previous bars to intermediate layers of the generator CNN. This way, our model can "look back" without a recurrent unit as used in RNNs. Like RNNs, our model can generate music of arbitrary number of bars.

Because we use random noises as inputs to our generator, our model can generate melodies *from scratch*, i.e. without any other prior information. However, due to the conditioner CNN, our model has the capacity to exploit whatever prior knowledge that is available and can be represented as a matrix. For example, our model can generate music by following a chord progression, or by following a few starting notes (i.e. a priming melody). Given the same priming melody, our model can generate different results each time, again due to the random input.

The proposed model can be extended to generate different types of music, by using different conditions. Based on an idea called *feature matching* [27], we propose a way to control the influence of such conditions on the generation result. We can then control, for example, how much the current bar should sound like the previous bars. Moreover, our CNNs can be easily extended to deal with tensors instead of matrices, to exploit multi-channel MIDIs and to generate music of multiple tracks or parts. We believe such a highly adaptive and generic model structure can be a useful alternative to RNN-based designs. We refer to this new model as the MidiNet.

In our experiment, we conduct a user study to compare the melodies generated by MidiNet and MelodyRNN models [33]. For fair comparison, we use the same priming melodies for them to generate melodies of eight-bar long (including the primers), without any other prior information. To demonstrate the flexibility of MidiNet, we provide the result of two additional settings: one uses additionally chord progressions of eight-bar long to condition the generation, and the other uses a slightly different network architecture to generate more creative music. For reproducibility, the source code and pre-trained models of MidiNet are released online [2].

[2] `https://github.com/RichardYang40148/MidiNet`

2. RELATED WORK

A large number of deep neural network models have been proposed lately for music generation. This includes models for generating a melody sequence or audio waveforms by following a few priming notes [10,18,19,22,31,33], accompanying a melody sequence with music of other parts [15], or playing a duet with human [4,26].

Table 1 compares MidiNet with a number of major related models. We briefly describe each of them below.

The MelodyRNN models [33] proposed by the Magenta Project from the Google Brain team are possibly among the most famous examples of symbolic-domain music generation by neural networks. In total three RNN-based models were proposed, including two variants that aim to learn longer-term structures, the lookback RNN and the attention RNN [33]. Source code and pre-trained models for the three models are all publicly available. [3] As the main function of MelodyRNN is to generate a melody sequence from a priming melody, we use the MelodyRNN models as the baseline in our evaluation.

Song from PI [7] is a hierarchical RNN model that uses a hierarchy of recurrent layers to generate not only the melody but also the drums and chords, leading to a multi-track pop song. This model nicely demonstrates the ability of RNNs in generating multiple sequences simultaneously. However, it requires prior knowledge of the musical scale and some profiles of the melody to be generated [7], which is not needed in many other models, including MidiNet.

DeepBach [15], proposed by Sony CSL, is specifically designed for composing polyphonic four-part chorale music in the style of J. S. Bach. It is an RNN-based model that allows enforcing user-defined constraints such as rhythm, notes, parts, chords and cadences.

C-RNN-GAN [21] is to date the only existing model that uses GAN for music generation, to our best knowledge. It also takes random noises as input as MidiNet does, to generate diverse melodies. However, it lacks a conditional mechanism [17, 20, 25] to generate music by following either a priming melody or a chord sequence.

[3] `https://github.com/tensorflow/magenta/tree/master/magenta/models/melody_rnn` (accessed 2017-4-26)

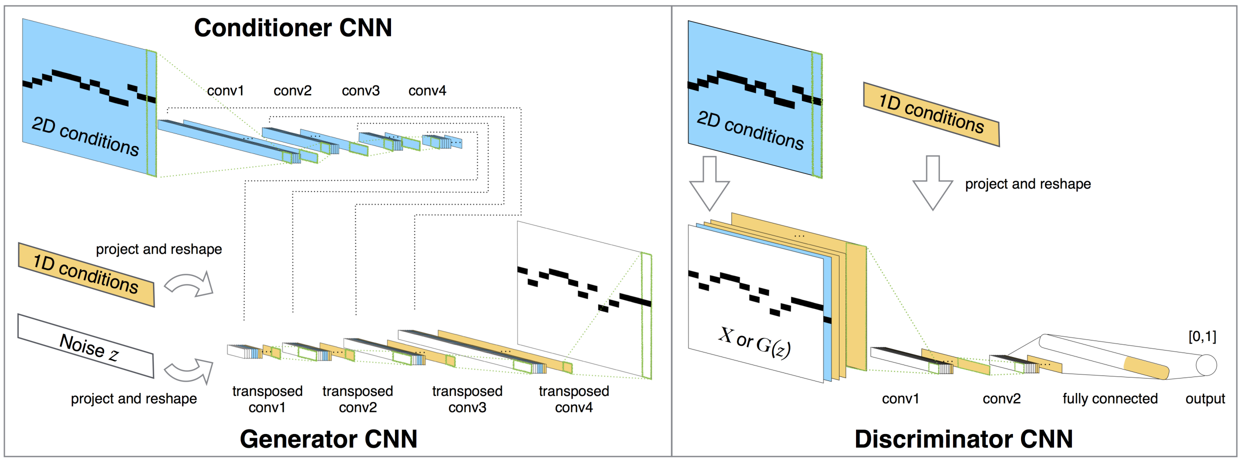


Figure 1. System diagram of the proposed MidiNet model for symbolic-domain music generation.

WaveNet [10, 31] is a CNN-based model proposed by DeepMind for creating raw waveforms of speech and music. The advantage of audio-domain generation is the possibility of creating new sounds, but we choose to focus on symbolic-domain generation in this paper.

3. METHODS

A system diagram of MidiNet is shown in Figure 1. Below, we present the technical details of each major component.

3.1 Symbolic Representation for Convolution

Our model uses a symbolic representation of music in fixed time length, by dividing a MIDI file into bars. The note events of a MIDI channel can be represented by an h-by-w real-valued matrix $\mathbf{X}$, where h denotes the number of MIDI notes we consider, possibly including one more dimension for representing silence, and w represents the number of time steps we use in a bar. For melody generation, there is at most one active note per time step. We use a binary matrix $\mathbf{X} \in \{0,1\}^{h \times w}$ if we omit the velocity (volume) of the note events. We use multiple matrices per bar if we want to generate multi-track music.

In this representation, we may not be able to easily distinguish between a long note and two short repeating notes (i.e. consecutive notes with the same pitch). Future extensions can be done to emphasize the note onsets.

3.2 Generator CNN and Discriminator CNN

The core of MidiNet is a modified deep convolutional generative adversarial network (DCGAN) [24], which aims at learning a discriminator D to distinguish between real (authentic) and generated (artificial) data, and a generator G that "fools" the discriminator. As typical in GANs, the input of G is a vector of random noises $\mathbf{z} \in \mathbb{R}^l$, whereas the output of G is an h-by-w matrix $\widehat{\mathbf{X}} = G(\mathbf{z})$ that "appears" to be real to D. GANs learn G and D by solving:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{X}\sim p_{\text{data}}(\mathbf{X})}[\log(D(\mathbf{X}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \,, \tag{1}$$

where $\mathbf{X} \sim p_{\text{data}}(\mathbf{X})$ denotes the operation of sampling from real data, and $\mathbf{z} \sim p_z(\mathbf{z})$ the sampling from a random distribution. As typical in GANs, we need to train G and D iteratively multiple times, to gradually make a better G.

Our discriminator is a typical CNN with a few convolution layers, followed by fully-connected layers. These layers are optimized with a cross-entropy loss function, such that the output of D is close to 1 for real data (i.e. $\mathbf{X}$) and 0 for those generated (i.e. $G(\mathbf{z})$). We use a sigmoid neuron at the output layer of D so its output is in [0,1].

The goal of the generator CNN, on the other hand, is to make the output of D close to 1 for the generated data. For generation, it has to transform a vector $\mathbf{z}$ into a matrix $\widehat{\mathbf{X}}$. This is achieved by using a few fully connected layers first, and then a few transposed convolution layers [8] that "upsamples" smaller vectors/matrices into larger ones.

Owing to the nature of minimax games, the training of GANs is subject to issues of instability and mode collapsing [12]. Among the various possible techniques to improve the training of GANs [1, 5, 27], we employ the so-called feature matching and one-sided label smoothing [27] in our model. The idea of feature matching is to add additional L2 regularizers to Eq. 1, such that the distributions of real and generated data are enforced to be close. Specifically, we add the following two terms when we learn G:

$$\lambda_1 \|\mathbb{E}\,\mathbf{X} - \mathbb{E}\,G(\mathbf{z})\|_2^2 + \lambda_2 \|\mathbb{E}\,f(\mathbf{X}) - \mathbb{E}\,f(G(\mathbf{z}))\|_2^2 \,, \tag{2}$$

where f denotes the first convolution layer of the discriminator, and λ_1, λ_2 are parameters to be set empirically.

3.3 Conditioner CNN

In GAN-based image generation, people often use a vector to encode available prior knowledge that can be used to condition the generation. This is achieved by reshaping the vector and then adding it to different layers of G and D, to provide additional input [20]. Assuming that the conditioning vector has length n, to add it to an intermediate layer of shape a-by-b we can duplicate the values ab times to get a tensor of shape a-by-b-by-n, and then concatenate it with the intermediate layer in the feature map axis. This is illustrated by the light orange blocks in Figure 1. We call such a conditional vector *1-D conditions*.

As the generation result of our GAN is an h-by-w matrix of notes and time steps, it is convenient if we can perform conditioning directly on each entry of the matrix. For example, the melody of a previous bar can be represented as another h-by-w matrix and used to condition the generation of the present bar. We can have multiple such matrices to learn from multiple previous bars. We can directly add such a conditional matrix to the input layer of D to influence all the subsequent layers. However, to exploit such *2-D conditions* in G, we need a mechanism to reshape the conditional matrix to smaller vectors of different shapes, to include them to different intermediate layers of G.

We propose to achieve this by using a conditioner CNN that can be viewed as a *reverse* of the generator CNN. As the blue blocks in Figure 1 illustrates, the conditioner CNN uses a few convolution layers to process the input h-by-w conditional matrix. The conditioner and generator CNNs use exactly the same filter shapes in their convolution layers, so that the outputs of their convolution layers have "compatible" shapes. In this way, we can concatenate the output of a convolution layer of the conditioner CNN to the input of a corresponding transposed convolution layer of the generator CNN, to influence the generation process. In the training stage, the conditioner and generator CNNs are trained simultaneously, by sharing the same gradients.

3.4 Tunning for Creativity

We propose two methods to control the trade-off between creativity and discipline of MidiNet. The first method is to manipulate the effect of the conditions by using them only in part of the intermediate transposed convolution layers of G, to give G more freedom from the imposed conditions. The second method capitalizes the effect of the feature matching technique [27]: we can increase the values of λ_1 and λ_2 to make the generated music sounds closer to existing music (i.e. those observed in the training set).

4. IMPLEMENTATION

4.1 Dataset

As the major task considered in this paper is melody generation, for training MidiNet we need a MIDI dataset that clearly specifies per file which channel corresponds to the melody. To this end, we crawled a collection of 1,022 MIDI tabs of pop music from TheoryTab, [4] which provides exactly two channels per tab, one for melody and the other for the underlying chord progression. With this dataset, we can implement at least two versions of MidiNets: one that learns from only the melody channel for fair comparison with MelodyRNN [33], which does not use chords, and the other that additionally uses chords to condition melody generation, to test the capacity of MidiNet.

[4] `https://www.hooktheory.com/theorytab`

	dimensions 1–12	13
major	C, C#, D, D#, E, F, F#, G, G#, A, A#, B	0
minor	A, A#, B, C, C#, D, D#, E, F, F#, G, G#	1

Table 2. 13-dimensional chord representation

For simplicity, we filtered out MIDI tabs that contain chords other than the 24 basic chord triads (12 major and 12 minor chords). Next, we segmented the remaining tabs every 8 bars, and then pre-processed the melody channel and the chord channel separately, as described below.

For melodies, we fixed the smallest note unit to be the sixteenth note, making $w = 16$. Specifically, we prolonged notes which have a pause note after them. If the first note of a bar is a pause, we extended the second note to have it played while the bar begins. There are other exceptions such as triplets and shorter notes (e.g. 32nd notes), but we chose to exclude them in this implementation. Moreover, for simplicity, we shifted all the melodies into two octaves, from `C4` to `B5`, and neglected the velocity of the note events. Although our melodies would use only 24 possible notes after these preprocessing steps, we considered all the 128 MIDI notes (i.e. from `C0` to `G10`) in our symbolic representation. In doing so, we can detect model collapsing [12] more easily, by checking whether the model generates notes outside these octaves. As there are no pauses in our data after preprocessing, we do not need a dimension for silence. Therefore, $h = 128$.

For chords, instead of using a 24-dimensional one-hot vector, we found it more efficient to use a chord representation that has only 13 dimensions— the first 12 dimensions for marking the key, and the last for the chord type (i.e. major or minor), as illustrated in Table 4.1. We pruned the chords such that there is only one chord per bar.

After these preprocessing steps, we were left with 526 MIDI tabs (i.e. 4,208 bars). [5] For data augmentation, we circularly shifted the melodies and chords to any of the 12 keys in equal temperament, leading to a final dataset of 50,496 bars of melody and chord pairs for training.

4.2 Network Specification

Our model was implemented in TensorFlow. For the generator, we used as input random vectors of white Gaussian noise of length $l = 100$. Each random vector go through two fully-connected layers, with 1024 and 512 neurons respectively, before being reshaped into a 1-by-2 matrix. We then used four transposed convolution layers: the first three use filters of shape 1-by-2 and two strides [8], and the last layer uses filters of shape 128-by-1 and one stride. Accordingly, our conditioner has four convolution layers, which use 128-by-1 filters for the first layer, and 1-by-2 filters for the other three. For creating a monophonic note sequence, we added a layer to the end of G to turn off per time step all but the note with the highest activation.

As typical in GANs, the discriminator is likely to overpower the generator, leading to the so-called vanishing gra-

[5] In contrast, MelodyRNN models [33] were trained on *thousands* of MIDI files, though the exact number is not yet disclosed.

dient problem [1,12]. We adopted two strategies to weaken the discriminator. First, in each iteration, we updated the generator and conditioner twice, but the discriminator only once. Second, we used only two convolution layers (14 filters of shape 128-by-2, two strides, and 77 filters of shape 1-by-4, two strides) and one fully-connected layer (1,024 neurons) for the discriminator.

We fine-tuned the other parameters of MidiNet and considered the following three variants in our experiment.

4.2.1 *Model 1: Melody generator, no chord condition*

This variant uses the melody of the previous bar to condition the generation of the present bar. We used this 2-D condition in all the four transposed convolution layers of G. We set the number of filters in all the four transposed convolution layers of G and the four convolution layers of the conditioner CNN to 256. The feature matching parameters λ_1 and λ_2 are set to 0.1 and 1, respectively. We did not use the 2-D condition for D, requiring it to distinguish between real and generated melodies from the present bar.

In the training stage, we firstly added one empty bar before all the MIDI tabs, and then randomly sampled two consecutive bars from any tab. We used the former bar as an instance of real data (i.e. X) and the input to D, and the former bar (which is a real melody or all zeros) as a 2-D condition and the input to the conditioner CNN Once the model was trained, we used G to generate melodies of 8-bar long in the following way: the first bar was composed of a real, priming melody sampled from our dataset; the generation of the second bar was made by G, conditioned by this real melody; starting from the third bar, G had to use the (artificial) melody it generated previously for the last bar as the 2-D condition. This process repeated until we had all the eight bars. [6]

4.2.2 *Model 2: Melody generator with chord condition, stable mode*

This variant additionally uses the chord channel. Because our MIDI tabs use one chord per bar, we used the chord (a 13-dimensional vector; see Table 4.1) of the present bar as a 1-D condition for generating the melody for the same bar. We can say that our model is generating a melody sequence that fits the given chord progression.

To highlight the chord condition, we used the 2-D previous-bar condition only in the last transposed convolution layer of G. In contrast, we used the 1-D chord condition in all the four transposed convolution layer of G, as well as the input layer for D. Moreover, we set $\lambda_1 = 0.01$, $\lambda_2 = 0.1$, and used 128 filters in the transposed convolution layers of G and only 16 filters in the convolution layers of the conditioner CNN. As a result, the melody generator is more *chord-dominant* and *stable*, for it would mostly follow the chord progression and seldom generate notes that violate the constraint imposed by the chords.

[6] It is also possible to use multiple previous bars to condition our generation, but we leave this as a future extension.

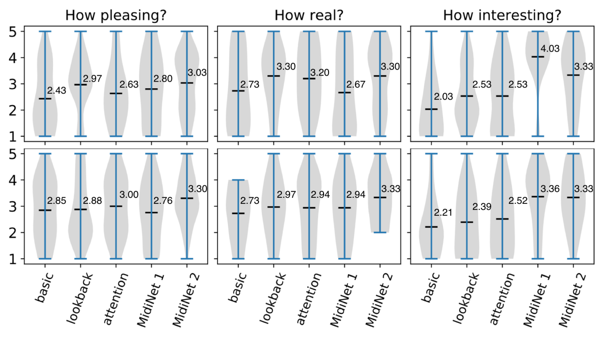


Figure 2. Result of a user study comparing MelodyRNN and MidiNet models, for people (top row) with musical backgrounds and (bottom) without musical backgrounds. The middle bars indicate the mean values. Please note that MidiNet Model 2 takes the chord condition as additional information.

4.2.3 *Model 3: Melody generator with chord condition, creative mode*

This variant realizes a slightly more creative melody generator by placing the 2-D condition in every transposed convolution layer of G. In this way, G would sometimes violate the constraint imposed by the chords, to somehow adhere to the melody of the previous bar. Such violations sometimes sound unpleasant, but can be sometimes creative. Unlike the previous two variants, we need to listen to several melodies generated by this model to handpick good ones. However, we believe such a model can still be useful for assisting and inspiring human composers.

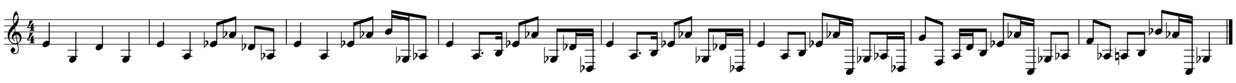
5. EXPERIMENTAL RESULT

To evaluate the aesthetic quality of the generation result, a user study that involves human listeners is needed. We conducted a study with 21 participants. Ten of them understand basic music theory and have the experience of being an amateur musician, so we considered them as people with musical backgrounds, or *professionals* for short.

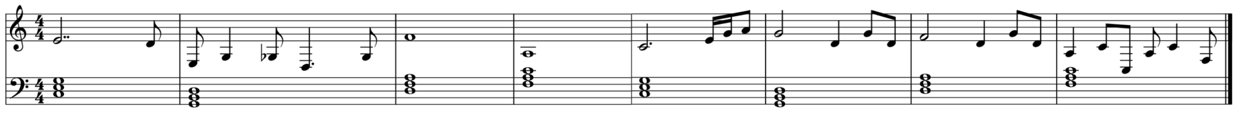
We compared MidiNet with three MelodyRNN models pre-trained and released by Google Magenta: the basic RNN, the lookback RNN, and the attention RNN [33]. We randomly picked 100 priming melodies from the training data [7] and asked the models create melodies of eight bars by following these primers. We considered two variants of MidiNet in the user study: model 1 (Section 4.2.1) for fair comparison with MelodyRNN, and model 2 (Section 4.2.2) for probing the effects of using chords. Although the result of model 2 was generated by additionally following the chords, we did not playback the chord channel in the user study.

We randomly selected the generation result of three out of the 100 priming melodies for each participant to listen to, leading to three sets of music. To avoid bias, we randomly shuffled the generation result by the five considered

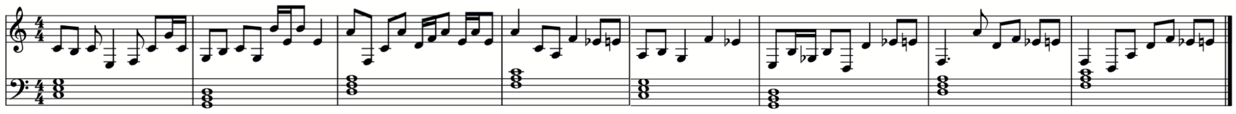
[7] Even though these priming melodies are in the training data, MidiNet generates melodies that are quite different from the existing ones.



(a) MidiNet model 1



(b) MidiNet model 2



(c) MidiNet model 3

Figure 3. Example result of the melodies (of 8 bars) generated by different implementations of MidiNet.

models, such that in each set the ordering of the five models is different. The participants were asked to stay in a quiet room separately and used a headphone for music listening through the Internet, one set at a time. We told them that some of the music "might be" real, and some might be generated by machine, although all of them were actually automatically generated. They were asked to rate the generated melodies in terms of the following three metrics: *how pleasing*, *how real*, and *how interesting*, from 1 (low) to 5 (high) in a five-point Likert scale.

The result of the user study is shown in Figure 2 as violin plots, where the following observations can be made. First, among the MelodyRNN models, lookback RNN and attention RNN consistently outperform basic RNN across the three metrics and two user groups (i.e. people with and without musical backgrounds), which is expected according to the report of Magenta [33]. The mean values for lookback RNN are around 3 (medium) for being pleasant and realistic, and around 2.5 for being interesting.

Second, MidiNet model 1, which uses only the previous bar condition, obtains similar ratings as the MelodyRNN models in being pleasant and realistic. This is encouraging, as MelodyRNN models can virtually exploit all the previous bars in generation. This result demonstrates the effectiveness of the proposed conditioner CNN in learning temporal information. Furthermore, we note that the melodies generated by MidiNet model 1 were found much more interesting than those generated by MelodyRNN. The mean value in being interesting is around 4 for people with musical backgrounds, and 3.4 for people without musical backgrounds. The violin plot indicates that the ratings of the professionals are mostly larger than 3.

Third, MidiNet model 2, which further uses chords, obtains the highest mean ratings in being pleasant and realistic for both user groups. In terms of interestingness, it also outperforms the three MelodyRNN models, but is inferior to MidiNet model 1, especially for professionals.

According to the feedback from the professionals, a melody sounds artificial if it lacks variety or violates principals in (Western) music theory. The result of MidiNet model 1 can sound artificial, for it relies on only the previous bar and hence occasionally generates "unexpected" notes. In contrast, the chord channel provides a musical context that can be effectively used by MidiNet model 2 through the conditional mechanism. However, occasional violation of music theory might be a source of interestingness and thereby creativity. For example, the professionals reported that the melodies generated by MelodyRNN models are sometimes too repetitive, or "safe," making them artificial and less interesting. It might be possible to further fine tune our model to reach a better balance between being real and being interesting, but we believe our user study has shown the promise of MidiNet.

Figure 3 shows some melodies generated by different implementations of MidiNet, which may provide insights into MidiNet's performance. Figure 3(a) shows that MidiNet model 1 can effectively exploit the previous bar condition—most bars start with exactly the same first two notes (as the priming bar) and they use similar notes in between. Figure 3(b) shows the result of MidiNet model 2, which highlights the chord condition. Figure 3(c) shows that MidiNet can generate more creative result by making the chord condition and previous bar condition equally strong. We can see stronger connections between adjacent bars from the result of this MidiNet model 3. For more audio examples, please go to `https://soundcloud.com/vgtsv6jf5fwq/sets`.

6. CONCLUSION

We have presented MidiNet, a novel CNN-GAN based model for MIDI generation. It has a conditional mechanism to exploit versatile prior knowledge of music. It also has a flexible architecture and can generate different types of music depending on input and specifications. Our evaluation shows that it can be a powerful alternative to RNNs.

For future work, we would extend MidiNet to generate multi-track music, to include velocity and pauses by training the model by using richer and larger MIDI data. We are also interested in using ideas of reinforcement learning [29] to incorporate principles of music theory [18], and to take input from music information retrieval models such as genre recognition [6] and emotion recognition [34].

7. REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[2] Jamshed J. Bharucha and Peter M. Todd. Modeling the perception of tonal structure with neural nets. *Computer Music Journal*, 13(4):44–53.

[3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[4] Mason Bretan, Gil Weinberg, and Larry Heck. A unit selection methodology for music generation using deep neural networks. *arXiv preprint arXiv:1612.03789*, 2016.

[5] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Proc. Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[6] Keunwoo Choi, George Fazekas, Mark B. Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. *arXiv preprint arXiv:1609.04243*, 2016.

[7] Hang Chu, Raquel Urtasun, and Sanja Fidler. Song from PI: A musically plausible network for pop music generation. *arXiv preprint arXiv:1611.03477*, 2016.

[8] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[9] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *Proc. IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756, 2002.

[10] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Douglas Eck, Karen Simonyan, and Mohammad Norouzi. Neural audio synthesis of musical notes with WaveNet autoencoders. *arXiv preprint arXiv:1704.01279*, 2017.

[11] Jon Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014:5, 2014.

[12] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2017.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[14] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[15] Gaëtan Hadjeres and François Pachet. DeepBach: a steerable model for bach chorales generation. *arXiv preprint arXiv:1612.01010*, 2016.

[16] Lejaren Hiller and Leonard M. Isaacson. *Experimental Music: Composition with an Electronic Computer*. New York: McGraw-Hill, 1959.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[18] Natasha Jaques, Shixiang Gu, Richard E. Turner, and Douglas Eck. Tuning recurrent neural networks with reinforcement learning. *arXiv preprint arXiv:1611.02796*, 2016.

[19] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron C. Courville, and Yoshua Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. *arXiv preprint arXiv:1612.07837*, 2016.

[20] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[21] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*, 2016.

[22] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A. Hasegawa-Johnson, and Thomas S. Huang. Fast WaveNet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

[23] George Papadopoulos and Geraint Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *Proc. AISB Symposium on Musical Creativity*, pages 110–117, 1999.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[25] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*, 2016.

[26] Adam Roberts, Jesse Engel, Curtis Hawthorne, Ian Simon, Elliot Waite, Sageev Oore, Natasha Jaques, Cinjon Resnick, and Douglas Eck. Interactive musical improvisation with Magenta. In *Proc. Neural Information Processing Systems*, 2016.

[27] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. In *Proc. Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[28] Zheng Sun, Jiaqi Liu, Zewang Zhang, Jingwen Chen, Zhao Huo, Ching Hua Lee, and Xiao Zhang. Composing music with grammar argumented neural networks and note-level encoding. *arXiv preprint arXiv:1611.05416*, 2016.

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[30] Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43.

[31] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[32] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelCNN decoders. In *Proc. Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[33] Elliot Waite, Douglas Eck, Adam Roberts, and Dan Abolafia. Project Magenta: Generating long-term structure in songs and stories, 2016. `https://magenta.tensorflow.org/blog/2016/07/15/lookback-rnn-attention-rnn/`.

[34] Yi-Hsuan Yang and Homer H. Chen. *Music Emotion Recognition*. CRC Press, 2011.

MODELING HARMONY WITH SKIP-GRAMS

David R. W. Sears **Andreas Arzt** **Harald Frostel**
Reinhard Sonnleitner **Gerhard Widmer**
Department of Computational Perception, Johannes Kepler University, Linz, Austria
david.sears@jku.at

ABSTRACT

String-based (or *viewpoint*) models of tonal harmony often struggle with data sparsity in pattern discovery and prediction tasks, particularly when modeling composite events like triads and seventh chords, since the number of distinct n-note combinations in polyphonic textures is potentially enormous. To address this problem, this study examines the efficacy of *skip-grams* in music research, an alternative viewpoint method developed in corpus linguistics and natural language processing that includes sub-sequences of n events (or n-grams) in a frequency distribution if their constituent members occur within a certain number of skips.

Using a corpus consisting of four datasets of Western classical music in symbolic form, we found that including skip-grams reduces data sparsity in n-gram distributions by (1) minimizing the proportion of n-grams with negligible counts, and (2) increasing the coverage of contiguous n-grams in a test corpus. What is more, skip-grams significantly outperformed contiguous n-grams in discovering conventional closing progressions (called *cadences*).

1. INTRODUCTION

Corpus studies employing string-based (or *viewpoint*) methods in music research often suffer from the *contiguity fallacy*—the assumption that note or chord events on the musical surface depend only on their immediate neighbors. For example, in symbolic music corpora, researchers often divide the corpus into contiguous sequences of n events (called n-grams) for the purposes of pattern discovery [4], classification [5], similarity estimation [16], and prediction [17]. And yet since much of the world's music is hierarchically organized such that certain events are more stable (or prominent) than others [1], *non-contiguous* events often serve as focal points in the sequence [11]. As a consequence, the contiguous n-gram method yields increasingly sparse distributions as n increases, resulting in the well-known *zero-frequency problem* [27], in which n-grams encountered in the test set do not appear in the training set. Perhaps worse, the most highly recurrent temporal

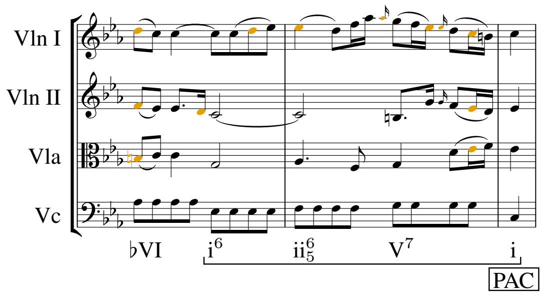


Figure 1: Haydn, String Quartet in C minor, Op. 17/4, i, mm. 6–8. Non-chord tones are shown with orange noteheads, and Roman numeral annotations appear below, with the chords of the perfect authentic cadence (PAC) progression embraced by a horizontal square bracket.

patterns in tonal music—melodic formulæ, conventional chord progressions, etc.—are rarely included.

By way of example, consider the closing measures of the main theme from the first movement of Haydn's string quartet Op. 17, No. 4, shown in Figure 1. The passage culminates in a *perfect authentic cadence*, a syntactic closing formula that features a conventional chord progression (V–I) and a falling upper-voice melody ($\hat{2}$–$\hat{1}$). In the music theory classroom, students are taught to reduce this musical surface to a succession of chord symbols, such as the Roman numeral annotations shown below. Yet despite the ubiquity of this pattern throughout the history of Western tonal music, string-based methods generally fail to retrieve this sequence of chords due to the presence of intervening non-chord tones (shown in orange), a limitation one study has called the *interpolation problem* [3].

To discover the organizational principles underlying tonal harmony using data-driven methods, this study examines the efficacy of *skip-grams* in music research, an alternative viewpoint method developed in corpus linguistics and natural language processing that includes subsequences in an n-gram distribution if their constituent members occur within a certain number of skips. In language corpora, skip-grams have been shown to reduce data sparsity in n-gram distributions [13], discover multi-word expressions (or *collocations*) in pattern discovery tasks [22], and minimize model uncertainty in word prediction tasks [12].

Models for the discovery of harmonic progressions in polyphonic corpora typically exclude higher-order sequences (when $n > 2$) due to the sparsity of their distributions [18], so this paper examines the utility of skip-grams for 2-grams, 3-grams, and 4-grams. We begin in Section 2 by describing the *voice-leading type* (VLT), an optimally reduced chord typology that models every possible combination of note events in the dataset, but that reduces the number of distinct chord types based on music-theoretic principles. Following a formal definition of skip-grams in Section 3, Section 4 describes the datasets used in the present research and then presents the experimental evaluations, which consider whether skip-grams reduce data sparsity in n-gram distributions by (1) minimizing the proportion of rare n-grams (i.e., that feature negligible counts), and (2) covering more of the contiguous n-grams in a test corpus. We conclude by considering avenues for future research.

2. DATA-DRIVEN CHORD TYPOLOGIES

Corpus studies in music research often treat the *note* event as the unit of analysis, examining features like chromatic pitch [18], melodic interval [23], and chromatic scale degree [15]. Using computational methods to identify *composite* events like triads and seventh chords in complex polyphonic textures is considerably more complex, since the number of distinct n-note combinations associated with any of the above-mentioned features is enormous.

To derive chord progressions from symbolic corpora using data-driven methods, many music analysis software frameworks perform a *full expansion* of the symbolic encoding, which duplicates overlapping note events at every unique onset time. [1] Shown in Figure 2, expansion results in the identification of 23 unique onset times. Since expansion is less likely to under-partition more complex polyphony compared to other partitioning methods [4], we adopt this technique for the analyses that follow.

To reduce the vocabulary of potential chord types, previous studies have represented each chord according to the simultaneous relations between its note-event members (e.g., vertical intervals) [21], the sequential relations between its chord-event neighbors (e.g., melodic intervals) [4], or some combination of the two [19]. The skip-gram method can model any of these representation schemes, but for the purposes of this study, we have adopted the *voice-leading type* (VLT) representation developed in [19, 20], which produces an optimally reduced chord typology that still models every possible combination of note events in the dataset. The VLT scheme consists of an ordered tuple (S, I) for each chord in the sequence, where S is a set of up to three intervals above the bass in semitones modulo the octave, resulting in 13^3 (or 2197) possible combinations; [2] and I is the melodic interval (again modulo the octave) from the preceding bass note to the present one.

Because the VLT representation makes no distinction between chord tones and non-chord tones, the syntactic

[1] In *Humdrum*, this technique is called *ditto* [14], while *Music21* calls it *chordifying* [6].

[2] The value of each vertical interval is either undefined (denoted by $\perp$), or represents one of twelve possible interval classes, where 0 denotes a perfect unison or octave, 7 denotes a perfect fifth, and so on.

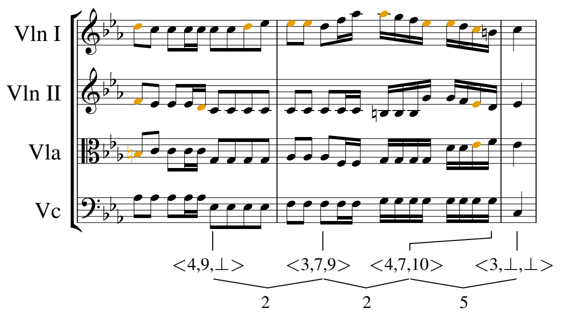


Figure 2: Full expansion of Op. 17/4, i, mm. 6–8. Non-chord tones are shown with orange noteheads, and the most representative chord onsets of the PAC progression are annotated with the VLT scheme.

domain of voice-leading types is still very large. To reduce the domain to a more reasonable number, we have excluded pitch class repetitions in S (i.e., voice doublings), and we have allowed permutations. Following [19], the assumption here is that the precise location and repeated appearance of a given interval are inconsequential to the identity of the chord. By allowing permutations, the major triads $\langle 4, 7, 0\rangle$ and $\langle 7, 4, 0\rangle$ therefore reduce to $\langle 4, 7, \perp\rangle$. Similarly, by eliminating repetitions, the chords $\langle 4, 4, 10\rangle$ and $\langle 4, 10, 10\rangle$ reduce to $\langle 4, 10, \perp\rangle$. This procedure restricts the domain to 233 unique VLTs when $n = 1$ (i.e., when I is undefined). Figure 2 presents the VLT encoding for the PAC progression annotated in Figure 1, with the vertical interval classes S provided below each chord onset, and the melodic interval classes I inserted under horizontal angle brackets.

3. DEFINING SKIP-GRAMS

In corpus linguistics, researchers often discover recurrent patterns by dividing the corpus into n-grams, and then determining the number of instances (or *tokens*) associated with each unique n-gram *type* in the corpus. N-grams consisting of one, two, or three events are often called *unigrams*, *bigrams*, and *trigrams*, respectively, while longer n-grams are typically represented by the value of n.

3.1 Contiguous N-grams

Each piece m consists of a contiguous sequence of VLTs, so let k represent the length of the sequence in each piece, and let C denote the total number of pieces in the corpus. The number of contiguous n-gram tokens in the corpus is

$$\sum_{m=1}^{C} k_m - n + 1 \qquad (1)$$

This formula ensures that the total number of tokens is necessarily smaller than the total number of events in the sequence when $n > 1$.

3.2 Non-Contiguous N-grams

The most serious limitation of contiguous n-grams is that they offer no alternatives; every event depends only on its

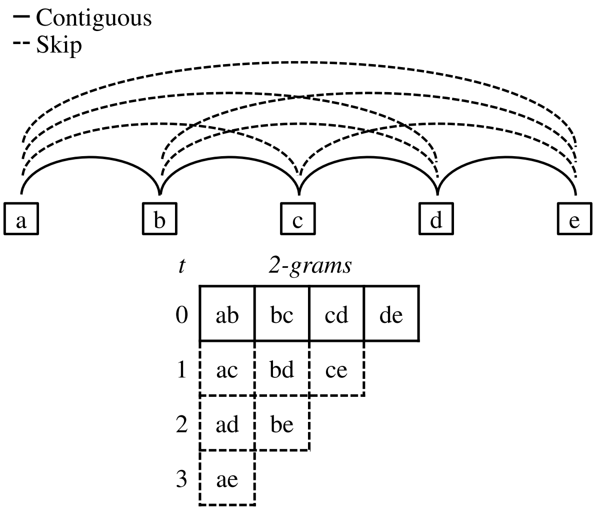


Figure 3: Top: A 5-event sequence, with arcs denoting all contiguous (solid) and non-contiguous (dashed) 2-gram tokens. Bottom: All 2-gram tokens, with t indicating the number of skips.

immediate neighbors. Without this limitation, the number of associations between events in the sequence necessarily explodes in combinatorial complexity as n and k increase.

The top plot in Figure 3 depicts the contiguous and non-contiguous 2-gram tokens for a 5-event sequence with solid and dashed arcs, respectively. According to (1), the number of contiguous 2-grams in a 5-event sequence is $k - n + 1$, or 4 tokens. If all possible non-contiguous relations are also included, the number of tokens is given by the combination equation:

$$\binom{k}{n} = \frac{k!}{n!(k-n)!} = \frac{k(k-1)(k-2)\dots(k-n+1)}{n!} \tag{2}$$

The notation $\binom{k}{n}$ denotes the number of possible combinations of n events from a sequence of k events. By including the non-contiguous associations, the number of 2-grams for a 5-event sequence increases to 10. As n and k increase, the number of patterns can very quickly become unwieldy: a 20-event sequence, for example, contains 190 possible 2-grams, 1140 3-grams, 4845 4-grams, and 15,504 5-grams.

3.2.1 Fixed-Skip N-grams

To overcome the combinatoric complexity of counting tokens in this way, researchers in natural language processing have limited the investigation to what we will call *fixed-skip n*-grams [13], which only include n-gram tokens if their constituent members occur within a fixed number of skips t. Shown in the bottom plot in Figure 3, ac and bd constitute 1-skip tokens (i.e., $t = 1$), while ad and be constitute 2-skip tokens. Thus, up to 7 tokens occur when $t = 1$, up to 9 occur when $t = 2$, and up to 10 occur when $t = 3$.

3.2.2 Variable-Skip N-grams

For natural language texts, the temporal structure of a sequence of linguistic utterances is not clearly defined. Yet for music corpora, temporal characteristics like onset time and duration play an essential role in the realization and reception of musical works. For example, the upper boundary under which listeners can group successive events into temporal sequences is around 2s [10]. Thus, as an alternative to the fixed-skip method, we also include *variable-skip n*-grams, which include n-gram tokens if the inter-onset interval(s) (IOI) between their constituent members occur within a specified upper boundary (e.g., 2s).

4. EXPERIMENTAL EVALUATIONS

This section describes the datasets in the present research and then examines whether the inclusion of skip-grams (1) minimizes the proportion of n-gram types with negligible counts, and (2) covers more of the contiguous n-gram tokens in a test corpus.

4.1 Datasets & Pre-Processing

Shown in Table 1, this study includes four datasets of Western classical music that feature symbolic representations of both the notated score (e.g., metric position, rhythmic duration, pitch, etc.) and a recorded expressive performance (e.g., onset time and duration in seconds, velocity, etc.). Altogether, the corpus totals over 20 hours of music.

The **Kodály/Haydn** dataset consists of 50 Haydn string quartet movements encoded in MIDI format [21]. The data were manually aligned at the downbeat level to recorded performances by the Kodály Quartet, and then the onset time for each chord event in the symbolic representation was estimated using linear interpolation.

The **Batik/Mozart** dataset consists of 13 complete Mozart piano sonatas encoded in MATCH format [24]. The data were aligned to performances by Roland Batik that were recorded on a Bösendorfer SE 290 computer-controlled piano, which is equipped with sensors on the keys and hammers to measure the timing and dynamics of each note [25].

The remaining two datasets were encoded in MusicXML format, and were also aligned to performances that were recorded on a Bösendorfer computer-controlled piano. The **Zeilinger/Beethoven** dataset consists of 9

Composer (Performer)	N_{pieces}	N_{chords}	$N_{\text{tokens}>3}$
Haydn (Kodály)	50	73,704	0
Mozart (Batik)	39	63,418	969
Beethoven (Zeilinger)	30	42,157	910
Chopin (Magaloff)	156	147,871	3666
Total	275	327,150	5545

Note. $N_{\text{tokens}>3}$ denotes n-gram tokens that initially consisted of more than three interval classes.

Table 1: Datasets and descriptive statistics for the corpus.

complete Beethoven piano sonatas performed by Clemens Zeilinger [8], while the **Magaloff/Chopin** dataset consists of 156 Chopin piano works that were performed by Nikita Magaloff [8, 9].

Performing a full expansion on all four datasets produced 327,150 unique onsets from which to derive chords. Unfortunately, some onsets presented more than three vertical interval classes, but since the VLT scheme only permits up to three interval classes S above the bass, it was necessary to replace these chords. Each onset containing more than three distinct vertical interval classes was replaced either with (1) the closest maximal subset estimated from the immediate surrounding context (i.e., ± 5 chords); (2) the most common maximal subset estimated from the entire piece; or finally (3) the most common maximal subset estimated from all pieces in the corpus.

4.2 Reducing Sparsity

In natural language corpora, n-gram distributions of individual words ($n = 1$) and multi-word expressions ($n < 5$) demonstrate a power-law relationship between frequency and rank, with the most frequent (i.e., top-ranked) types accounting for the majority of the tokens in the distribution [26]. In music corpora, however, this relationship becomes increasingly linear as n increases due to the greater proportion of types featuring negligible counts. Such rare n-grams are thus more difficult to retrieve and model in discovery and prediction tasks, so this section examines whether the inclusion of skip-grams minimizes the proportion of rare n-grams in chord distributions.

4.2.1 Methods

Contiguous n-gram distributions were calculated from $n = 1$ to $n = 7$, along with 4-grams that include the following skip levels: *Fixed* – up to 1, 2, 3, or 4 skips; *Variable* – all possible skips occurring within a maximum IOI of .5, 1, 1.5, or 2s.

Skip	N_{types}	N_{tokens}
No Skip		
	135,331	326,034
Fixed – Skip boundary (#)		
1	850,222	2,604,972
2	2,364,840	8,780,643
3	4,765,289	20,786,976
4	8,207,123	40,548,000
Variable – IOI[a] *boundary* (*s*)		
0.5	2,213,148	10,150,852
1	12,498,736	90,278,381
1.5	31,591,468	306,289,766
2	59,147,107	718,717,231

[a] IOI denotes the maximum permitted inter-onset interval in seconds between adjacent members of each n-gram.

Table 2: Counts associated with 4-gram types and tokens using both fixed and variable skips.

4.2.2 Results

Table 2 presents the counts for 4-gram types and tokens with both fixed and variable skips. As expected, including skips of either type significantly increased the number of types and tokens. When skips were not included, the corpus produced over 300 thousand tokens, but this number increased to over 40 million tokens for skip-grams including up to 4 skips, or over 700 million tokens for skip-grams including all skips occurring within an IOI of 2s.

To visualize the increasing impact of data sparsity on the n-gram distribution as n increases, the top plot in Figure 4 presents the cumulative probability distributions for contiguous n-gram types from $n = 1$ to $n = 7$. Types appearing to the right of each marker feature only one token in the corpus. When n is small, the distributions loosely conform to the family of power laws used in linguistics to describe the frequency-of-occurrence of words in language corpora, where a small proportion of types account for most of the encountered tokens. When n increases, however, the proportion of types featuring negligible counts also increases, resulting in increasingly uniform distributions.

Shown in the bottom plot in Figure 4, the power-law relationship returns in the 4-gram distributions when skips are included. What is more, the proportion of types featuring negligible counts also decreases, thereby minimizing

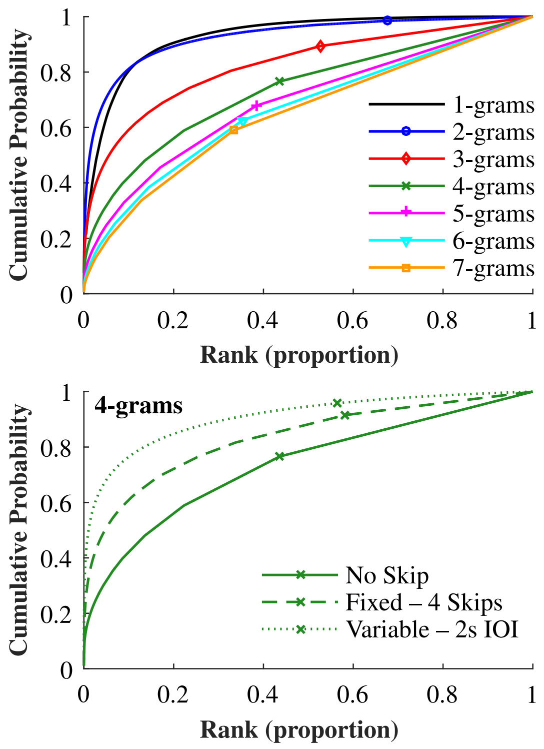


Figure 4: Cumulative probability distributions for (top) contiguous n-gram types, with types appearing to the right of each marker featuring only one token in the corpus; and (bottom) 4-gram types featuring no skips, up to four skips, or all skips occurring within an IOI of 2s.

the potential for data sparsity in the VLT distribution.

4.3 Increasing Coverage

This section examines whether the inclusion of skip-gram types during training covers more of the contiguous n-gram tokens in a test corpus.

4.3.1 Methods

2-gram, 3-gram, and 4-gram distributions were calculated for the following skip levels: *Fixed* – no skip, or up to 1, 2, 3, or 4 skips; *Variable* – no skip, or all possible skips occurring within an IOI of .5, 1, 1.5, or 2s. To evaluate skip-gram coverage, we employed 10-fold cross-validation stratified by composer [7], using the proportion of contiguous n-gram types in the test set that appeared in the training set as a measure of performance. To create folds containing the same number of compositions *and* chords, we computed the mean number of chords that should appear in each fold m, and then selected the fold indices for which each fold (1) contained an approximately equal number of compositions, and (2) contained a total number of chords that was $\pm 1\%$ of m.

4.3.2 Analysis

To examine the potential increase in coverage at each successive (fixed or variable) skip, we calculated a planned comparison statistic that does not assume equal variances, called the Welch t test.[3] The mean of each skip was compared to the mean of the previous skip using backward-difference coding (e.g., *Fixed*: 2 skips vs. 1 skip, 3 skips vs. 2 skips, etc.). To minimize the risk of committing a Type I error, each comparison was corrected with Bonferroni adjustment, which divides the significance criterion by the number of planned comparisons.

4.3.3 Results

Figure 5 displays line plots of the mean proportion of contiguous n-gram tokens from the test that appeared during training using either fixed or variable skips. Table 3 provides the mean coverage estimates and planned comparisons. For 2-grams, on average the contiguous types covered nearly 96% of the tokens in the test set. When skips were included, this estimate improved significantly to 98.3% of the tokens for up to two fixed skips, or up to 99.2% percent of the tokens for all skips occurring within an IOI of 1.5 s.

As n increased, the proportion of tokens that appeared during training using contiguous n-grams decreased substantially. For 3-grams, the contiguous types only covered 70.7% of the tokens on average. This estimate improved dramatically when either fixed or variable skips were included, however. For the fixed-skip factor, including up to

[3] In hypothesis testing, planned comparisons typically follow an omnibus statistic like the F ratio, which indicates whether the differences between the means of a given factor are significant. In this case, the Welch F test was significant for every model, so we forgo reporting those statistics here, and instead simply report the planned comparisons, which indicate whether coverage *increased* significantly as the number of skips (or the size of the temporal boundary) increased.

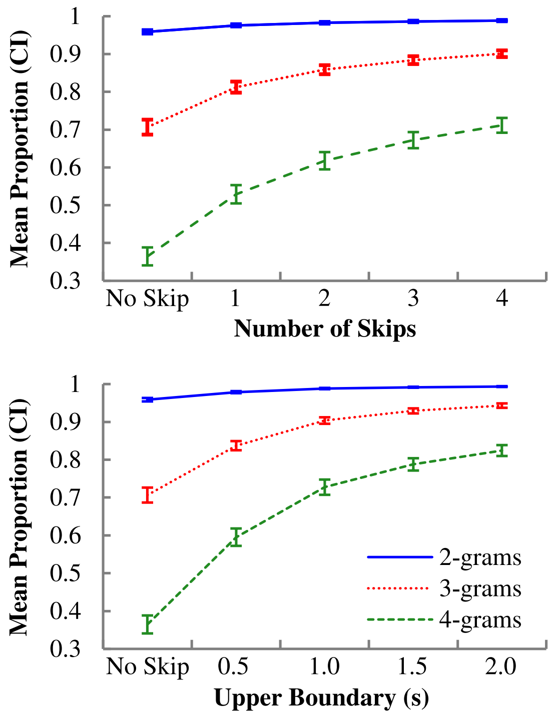


Figure 5: Line plots of the mean proportion of n-gram tokens from the test that were covered during training using either fixed (top) or variable (bottom) skips. Whiskers represent the 95% confidence interval (CI) around the mean.

four skips during training covered an additional 20% of the tokens during test, resulting in a mean coverage estimate of over 90%. In the variable-skip condition, this estimate further improved to 94.3% when all skips occurring within an IOI of 2s were included. Finally, for 4-grams, the contiguous types covered just 36.5% of the tokens, but this estimate improved to 71.1% in the fixed-skip condition, and to 82.4% in the variable-skip condition.

5. SUMMARY AND CONCLUSION

To reduce data sparsity in n-gram distributions of tonal harmony, this study examined the efficacy of skip-grams, an alternative viewpoint method that includes sub-sequences in an n-gram distribution if their constituent members occur within a certain number of skips (*fixed*), or a specified temporal boundary (*variable*). To that end, we compiled four datasets of Western classical music that feature symbolic representations of the notated score. Our findings demonstrate that the inclusion of skip-grams reduces sparsity in higher-order n-gram distributions by (1) minimizing the proportion of n-grams with negligible counts, thus recovering the power-law relationship between frequency and rank when $n < 5$ that was previously lost in the corresponding contiguous distributions, and (2) increasing the coverage of the contiguous n-grams in a test set, thereby mitigating the severity of the zero-frequency problem.

In our view, this approach would directly benefit tasks

	2-grams			**3-grams**			**4-grams**		
Skip	M_{coverage}	t	p	M_{coverage}	t	p	M_{coverage}	t	p
No Skip									
	.959			.707			.365		
Fixed – Skip boundary (#)									
1	.976	7.144	<.001	.813	9.726	<.001	.529	10.963	<.001
2	**.983**	4.000	.003	.859	5.518	<.001	.618	6.023	<.001
3	.986	2.529	.085	.884	3.620	.008	.672	3.948	.003
4	.988	1.848	.327	**.901**	2.814	.046	**.711**	3.063	.027
Variable – IOI boundary (s)									
0.5	.979	8.439	<.001	.837	12.744	<.001	.595	15.795	<.001
1	.988	6.598	<.001	.904	10.132	<.001	.727	9.786	<.001
1.5	**.992**	3.647	.010	.929	5.313	<.001	.788	5.266	<.001
2	.993	2.311	.132	**.943**	3.564	.009	**.824**	3.808	.005

Table 3: Mean coverage estimates and planned comparisons for 2-gram, 3-gram, and 4-gram tokens using either fixed or variable skips.

related to pattern discovery and prediction, since recurrent temporal patterns rarely appear on the musical surface, thereby forcing n-gram models to either exclude higher-order n-grams (e.g., where $n > 2$) due to the sparsity of the distributions, or calculate *escape probabilities* to accommodate patterns that do not appear (contiguously) in the training set [2]. Consider, for example, the two four-chord cadential progressions in Table 4: the *semplice* cadence, which features a dominant-to-tonic progression in root position (e.g., I^6-ii^6-V^7-I); and the *composta* cadence, which also features a six-four suspension above the cadential dominant (e.g., ii^6-"I^6_4"-V^7-I). These cadences are ubiquitous in music of the classical style, and yet the VLT configurations representing these progressions rarely appear on the surface; the semplice cadence *never* appears contiguously, while the composta cadence is featured in just seven pieces. When skips are included, however, the two progressions appear in 32 and 77 of the 245 pieces in the corpus, respectively.

Skip	I^6-ii^6-V^7-I	ii^6-"I^6_4"-V^7-I
No Skip		
	0	7
Fixed – Skip boundary (#)		
1	3	16
2	10	36
3	13	50
4	15	63
Variable – IOI[a] *boundary* (s)		
0.5	5	8
1	10	33
1.5	21	51
2	32	77

Note. VLT encodings for these progressions appear in the major and minor mode, and feature the pre-dominant and dominant harmonies both with and without the seventh (e.g., ii^6 and ii^6_5).

Table 4: Number of pieces containing semplice or composta four-chord progressions using both fixed and variable skips.

Due to the combinatoric complexity of the task, one limitation of the skip-gram method is that execution times become unfeasible beyond certain values of n and t. Nevertheless, if the organizational principles underlying hierarchical stimulus domains like natural language or polyphonic music reflect limitations of human auditory processing, it seems reasonable to impose similar restrictions on the sorts of contiguous and non-contiguous relations the skip-gram method should model. Given the restrictions imposed in this study, retrieving all 4-gram tokens from a sequence of 1,000 chords using commodity hardware produced runtimes of less than 100ms in the largest fixed-skip condition ($t = 4$ skips), and less than 3s in the largest variable-skip condition ($t = 2$s), proving skip-gram modeling is entirely attainable in a research setting.

Of course, counting all possible skip-grams in this way assumes no a priori knowledge about the sorts of non-contiguous relations analysts might hope to discover. For example, collocation extraction algorithms in the NLP community typically exclude infrequent n-grams, or use parts-of-speech tags to privilege syntactically meaningful utterances [22]. Music researchers could adopt similar methods by excluding (or weighting) each n-gram by the temporal proximity or periodicity of its members [21], or privileging patterns that appear in strong metric positions or feature changes of harmony. Together with the skip-gram method, these techniques could usher in a new suite of inductive, data-driven tools for the discovery of musical organization.

6. ACKNOWLEDGMENTS

This research is supported by the European Research Council (ERC) under the EUs Horizon 2020 Framework Programme (ERC Grant Agreement number 670035, project "Con Espressione").

7. REFERENCES

[1] J. J. Bharucha and C. L. Krumhansl. The representation of harmonic structure in music: Hierarchies of stability as a function of context. *Cognition*, 13:63–102, 1983.

[2] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.

[3] T. Collins, A. Arzt, H. Frostel, and G. Widmer. Using geometric symbolic fingerprinting to discover distinctive patterns in polyphonic music corpora. In D. Meredith, editor, *Computational Music Analysis*, pages 445–474. Springer International Publishing, Cham, 2016.

[4] D. Conklin. Representation and discovery of vertical patterns in music. In C. Anagnostopoulou, M. Ferrand, and A. Smaill, editors, *Music and Artifical Intelligence: Lecture Notes in Artificial Intelligence 2445*, volume 2445, pages 32–42. Springer-Verlag, 2002.

[5] D. Conklin. Multiple viewpoint systems for music classification. *Journal of New Music Research*, 42(1):19–26, 2013.

[6] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In J. S. Downie and R. C. Veltkamp, editors, *Proc. 11th International Society for Music Information Retrieval (ISMIR)*, pages 637–642, 2010.

[7] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[8] S. Flossmann. *Expressive Performance Rendering with Probabilistic Models — Creating, Analyzing, and Using the Magaloff Corpus*. Phd thesis, Johannes Kepler University, Linz, Austria, 2010.

[9] S. Flossmann, W. Goebl, M. Grachten, B. Niedermayer, and G. Widmer. The Magaloff project: An interim report. *Journal of New Music Research*, 39(4):363–377, 2010.

[10] P. Fraisse. Rhythm and tempo. In D. Deutsch, editor, *The Psychology of Music*, pages 149–180. Academy Press, New York, 1982.

[11] R. O. Gjerdingen. "Historically informed" corpus studies. *Music Perception*, 31(3):192–204, 2014.

[12] J. T. Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15:404–434, 2001.

[13] D. Guthrie, B. Allison, W. Liu, L. Guthrie, and Y. Wilks. A closer look at skip-gram modelling. In *Proc. 5th International Conference on Language Resources and Evaluation (LREC-2006)*, pages 1222–1225. European Language Resources Association, 2006.

[14] D. Huron. *The Humdrum Toolkit: Software for Music Research*. Center for Computer Assisted Research in the Humanities, Stanford, CA, 1993.

[15] E. H. Margulis and A. P. Beatty. Musical style, psychoaesthetics, and prospects for entropy as an analytic tool. *Computer Music Journal*, 32(4):64–78, 2008.

[16] D. Müllensiefen and M. Pendzich. Court decisions on music plagiarism and the predictive value of similarity algorithms. *Musicæ Scientiæ*, Discussion Forum 4B:257–295, 2009.

[17] M. T. Pearce. *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. Phd thesis, City University, London, 2005.

[18] M. T. Pearce and G. A. Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385, 2004.

[19] I. Quinn. Are pitch-class profiles really "key for key"? *Zeitschrift der Gesellschaft der Musiktheorie*, 7:151–163, 2010.

[20] I. Quinn and P. Mavromatis. Voice-leading prototypes and harmonic function in two chorale corpora. In C. Agon, E. Amiot, M. Andreatta, G. Assayag, J. Bresson, and J. Manderau, editors, *Mathematics and Computation in Music*, pages 230–240. Springer, Heidelberg, 2011.

[21] D. R. W. Sears. *The Classical Cadence as a Closing Schema: Learning, Memory, and Perception*. Phd thesis, McGill University, Montreal, Canada, 2016.

[22] F. Smadja. Retrieving collocations from text: Extract. *Computational Linguistics*, 19(1):143–177, 1993.

[23] P. G. Vos and J. M. Troost. Ascending and descending melodic intervals: Statistical findings and their perceptual relevance. *Music Perception*, 6(4):383–396, 1989.

[24] G. Widmer. Using AI and machine learning to study expressive music performance: Project survey and first report. *AI Communications*, 14(3):149–162, 2001.

[25] G. Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146:129–148, 2003.

[26] J. Williams, P. R. Lessard, S. Desu, E. M. Clark, J. P. Bagrow, C. M. Danforth, and P. S. Dodds. Zipf's law holds for phrases, not words. *Scientific Reports*, 5(12209), 2015.

[27] I. H. Witten and T. C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.

NMED-T: A TEMPO-FOCUSED DATASET OF CORTICAL AND BEHAVIORAL RESPONSES TO NATURALISTIC MUSIC

Steven Losorelli[1,2] **Duc T. Nguyen**[1,2] **Jacek P. Dmochowski**[3] **Blair Kaneshiro**[1,2]
[1]Center for the Study of Language and Information, Stanford University, USA
[2]Center for Computer Research in Music and Acoustics, Stanford University, USA
[3]Department of Biomedical Engineering, City College of New York, USA
{slosorelli, dtn006, blairbo}@stanford.edu jdmochowski@ccny.cuny.edu

ABSTRACT

Understanding human perception of music is foundational to many research topics in Music Information Retrieval (MIR). While the field of MIR has shown a rising interest in the study of brain responses, access to data remains an obstacle. Here we introduce the Naturalistic Music EEG Dataset—Tempo (NMED-T), an open dataset of electrophysiological and behavioral responses collected from 20 participants who heard a set of 10 commercially available musical works. Song stimuli span various genres and tempos, and all contain electronically produced beats in duple meter. Preprocessed and aggregated responses include dense-array EEG and sensorimotor synchronization (tapping) responses, behavioral ratings of the songs, and basic demographic information. These data, along with illustrative analysis code, are published in Matlab format. Raw EEG and tapping data are also made available. In this paper we describe the construction of the dataset, present results from illustrative analyses, and document the format and attributes of the published data. This dataset facilitates reproducible research in neuroscience and cognitive MIR, and points to several possible avenues for future studies on human processing of naturalistic music.

1. INTRODUCTION

Humans possess a unique ability to process music, and many topics in Music Information Retrieval (MIR) involve computational modeling of human perception. Tasks that humans often perform with ease—such as melody extraction, beat detection, and artist identification—remain open topics in MIR. At the same time, a full understanding of the cognitive and perceptual processes underlying human processing of music has yet to be reached.

Greater cross-disciplinary collaboration between MIR and neuroscience has been proposed [14], and a number of studies have incorporated approaches from both fields. For example, neural correlates of short- and long-term features introduced in MIR for genre classification [34] have been sought [1, 6, 10, 20], and brain responses have been used in MIR-related applications including tempo estimation [29, 30] and emotion recognition [5, 21]. Yet even as brain data become more prevalent in MIR research, experimental design, data collection, and data cleaning can present challenges [14]. Therefore, the research community can arguably benefit from curated, ready-to-use datasets of brain responses to real-world musical works.

Aiming to provide an open dataset with which several MIR and neuroscience topics can be explored, we introduce the Naturalistic Music EEG Dataset—Tempo (NMED-T), a dataset of EEG and behavioral responses to commercially available musical works. The dataset contains dense-array EEG responses from 20 adult participants who listened to 10 full-length songs, as well as tapped responses to the beat of shorter excerpts (collected in a separate listen). These responses have been cleaned and aggregated, and are ready to use in Matlab format along with ratings of familiarity and enjoyment, as well as basic demographic information about the participants.

NMED-T contributes to a growing body of publicly available music-related EEG repositories, including the DEAP [17], Music BCI [32], NMED-H [15], and OpenMIIR [31] datasets. It is well suited for MIR research in that the data are cleaned and ready to use but are also made available in raw form; stimuli are complete, naturalistic [1] musical works spanning a wide range of tempos; metadata links to stimulus audio are provided; and behavioral data are included. Moreover, as EEG was recorded while participants listened attentively but did not focus on any particular dimension of the songs, these data are suitable for studying many aspects of music processing.

The remainder of the paper is structured as follows. In § 2 we describe stimulus selection, study design, data collection, and data preprocessing. Illustrative analyses of the preprocessed data, which build upon past music perception and MIR approaches and reveal cortical and behavioral evidence of entrainment to musical beat, are presented in § 3. In § 4 we document the dataset itself. We conclude and discuss potential future uses of the data in § 5.

[1] Denoting real-world music—i.e., music that was created to be consumed in everyday life, as opposed to controlled stimuli created for experimental research.

2. METHODS

2.1 Stimuli

2.1.1 Stimulus Selection

As the present dataset is focused on naturalistic music and tempo, stimuli were ecologically valid, real-world musical works containing steady, electronically produced beats in duple meter at a variety of tempos. The 10 selected songs are all 4:30–5:00 in length, contain vocals (all but one in English), and are in the Western musical tradition. Song information is summarized in Table 1.

To aid in song selection, we computed objective measures of tempo using publicly available Matlab code [8]. The computed tempos were then validated perceptually by four trained musicians. The final set of selected songs range in tempo from 56–150 BPM—a wide enough range to potentially explore octave errors [11, 35]. To facilitate further research on the audio as well as the responses, we purchased digital versions of all songs from Amazon, and include in Table 1 each song's Amazon Standard Identification Number (ASIN).

These real-world stimuli are complex and contain energy at various frequencies—not just those directly related to the beat. We followed the approach of Nozaradan et al. [27] and visualized low-frequency spectra of the stimuli. We extracted the amplitude envelope of each song using the MIR Toolbox, version 1.5 [18] at a sampling rate of 125 Hz (the sampling rate of the preprocessed EEG), and plotted magnitude spectra up to 15 Hz. As can be seen in Fig. 1, spectral peaks often occur at harmonics and subharmonics of the beat—implicating the hierarchical timescale of music—as well as at other frequencies.

2.1.2 Stimulus Preparation

To prepare the stimuli for the EEG experiment, full-length songs were first converted to mono using Audacity, version 2.1.2. [2] We then embedded the second audio channel with an intermittent click that was transmitted directly to the EEG amplifier (not played to participants) to ensure precise time stamping of the stimuli. For the behavioral experiment, we created 35-second excerpts of each song. Using Audacity, we selected the audio from 1:00–1:34 and applied a linear fade-in and fade-out to the first and last 2 seconds, respectively. We then appended 1 second of silence to make the conclusion of each excerpt more obvious to the participant.

2.2 Participants

Twenty right-handed participants, aged 18–29 years (mean age 23 years, 6 female) participated in the experiment. All reported normal hearing, fluency in English, and no cognitive or decisional impairments. We imposed no eligibility criteria related to formal musical training; 17 participants reported having received training (mean 8.4 years among those with training). Participants reported listening to music for 14.5 hours per week on average.

[2] `http://www.audacityteam.org`

2.3 Experimental Specifications & Data Collection

This study was approved by the Stanford University Institutional Review Board. All participants provided written informed consent before participating. Each participant filled out a general demographic and musical background questionnaire, after which the EEG and tapping blocks were completed, with the EEG block always occurring first.

2.3.1 EEG Experiment

First, each participant was informed that the general purpose of the experiment was to study human processing of music, and that he or she would be completing an EEG session and a behavioral test. As the EEG data were collected for the general study of music processing (not limited to beat perception), no explicit mention of beat or tempo was given at this stage of the experiment. Rather, participants were instructed simply to listen attentively to the songs as they played, and to avoid movement of any kind (including stretching, yawning, and tapping or moving to the beat) during the trials. Songs were presented in random order. Following each trial, participants delivered ratings of familiarity and enjoyment for the song just presented, on a scale of 1–9. The EEG experiment was split into two consecutive recording blocks in order to mitigate participant fatigue, limit data size of the EEG recordings, and allow for verification of electrode impedances between recordings. Therefore, a total of 40 EEG recordings were collected across the 20 participants.

The EEG experiment was programmed in Matlab version 2013b [3] with a custom template built on the Psychophysics Toolbox, version 3 [4]. Each participant sat comfortably in a chair at a desk for the duration of the experiment. Stimuli were presented through magnetically shielded Genelec 1030A speakers at a measured loudness level between 73–78 dB. During the trials, the participant viewed a fixation image presented on a computer monitor located 57 cm in front of him or her.

Dense-array EEG was recorded using the Electrical Geodesics, Inc. (EGI) GES300 system [33]. Data were recorded from 128 electrodes with vertex reference using an EGI Net Amps 300 amplifier and Net Station 4.5.7 acquisition software, sampled at 1 kHz with a range of 24 bits. Electrode impedances were verified to be no greater than 50 kΩ—an appropriate level for this system—at the start of each recording.

2.3.2 Behavioral Experiment

Following the EEG recordings, the electrode net was removed from the participant, and the behavioral test began. Here, each participant listened to the 35-second song excerpts, after receiving instructions to "tap to the steady beat of the song as you perceive it." If the participant had questions about tapping to multiple tempos for a given song, he or she was instructed to tap to the steady beat that best reflected his or her perception of it in the moment. Excerpts were presented in random order.

[3] `https://www.mathworks.com`

#	Song Title	Artist	ASIN	Tempo (BPM)	Tempo (Hz)	min:sec
1	"First Fires"	Bonobo	B00CJE73J6	55.97	0.9328	4:38
2	"Oino"	LA Priest	B00T4NHS2W	69.44	1.1574	4:31
3	"Tiptoes"	Daedelus	B011SAZRLC	74.26	1.2376	4:36
4	"Careless Love"	Croquet Club	B06X9736NJ	82.42	1.3736	4:54
5	"Lebanese Blonde"	Thievery Corporation	B000SF16MI	91.46	1.5244	4:49
6	"Canopée"	Polo & Pan	B01GOL4IB0	96.15	1.6026	4:36
7	"Doing Yoga"	Kazy Lambist	B01JDDVIQ4	108.70	1.8116	4:52
8	"Until the Sun Needs to Rise"	Rüfüs du Sol	B01APT6JKA	120.00	2.0000	4:52
9	"Silent Shout"	The Knife	B00IMN40O4	128.21	2.1368	4:54
10	"The Last Thing You Should Do"	David Bowie	B018GS2A46	150.00	2.5000	4:58

Table 1. Stimulus set. Songs were selected on the basis of vocals, electronically produced beats, genre, tempo, and length.

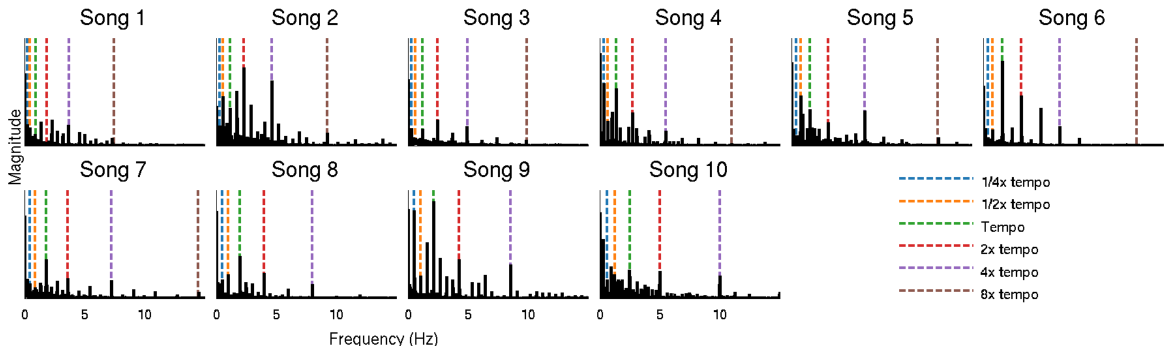


Figure 1. Low-frequency magnitude spectra of stimulus amplitude envelopes. Frequencies related to the musical beat hierarchy, from $1/4x$ the tempo (whole notes) to $8x$ the tempo (32nd notes) are denoted with vertical dashed lines.

Tapping responses were collected using Tap-It, an iOS application that plays audio while simultaneously recording responses tapped on the touchscreen [16]. We note a tap-to-timestamp latency of approximately 15 msec (st. dev. 5 msec) [16]. An Apple iPad 2 was used for this experiment, with stimuli delivered at a comfortable listening level using over-ear Sony MDR-V6 headphones.

2.4 Data Preprocessing

All data preprocessing and analysis was conducted using Matlab, versions 2013b and 2016b.

2.4.1 EEG Preprocessing

The following preprocessing steps were performed on individual EEG recordings that had been exported from Net Station to Matlab cell arrays. First, data from each electrode in the electrodes-by-time data matrix were zero-phase filtered using 8th-order Butterworth highpass (0.3 Hz) and notch (59–61 Hz) filters, and a 16th-order Chebyshev Type I lowpass (50 Hz) filter. Following this, the filtered data were temporally downsampled by a factor of 8 to a final sampling rate of 125 Hz.

We extracted trial labels, onsets, and behavioral ratings, and corrected the stimulus onset times using the click events sent directly from the audio to the EEG amplifier. The data for each trial were epoched, concatenated, and DC corrected (subtracting from each electrode its median value). Bad electrodes were removed from the data matrix, resulting in a reduction in the number of rows. We computed EOG components for tracking vertical and horizontal eye movements, and retained electrodes 1–124 for further analysis, excluding electrodes on the face. We applied a validated approach using Independent Components Analysis (ICA) to remove ocular and cardiac artifacts from the data [2, 13] using the `runica` function from the EEGLAB toolbox [7].

As final preprocessing steps, transients exceeding 4 standard deviations of each electrode's mean power were identified in an iterative fashion and replaced with NaNs. We then reconstituted missing rows corresponding to previously identified bad electrodes with rows of NaNs, ensuring that each data matrix contained the same number of rows. We appended a row of zeros—representing the vertex reference—and converted the data frame to average reference (subtracting from each electrode the mean of all electrodes). All missing values (NaNs) were imputed with the spatial average of data from neighboring electrodes, and a final DC correction was performed. Finally, the epochs were separated once again into single trials. Therefore, after preprocessing, each recording produced a cell array of EEG data, each element of which contained an electrodes-by-time matrix of size $125 \times T$, where T varied according to the length of the stimulus.

After preprocessing all recordings, we aggregated the data on a per-song basis. The data frame for each song is thus a 3D electrodes-by-time-by-participant matrix of size $125 \times T \times 20$.

2.4.2 Preprocessing of Tapping Responses

The Tap-It application stores the timestamps of taps, in seconds, measured from the device touchscreen on a per-trial basis, as well as each participant's randomized stimulus ordering array [16]. We aggregated the tapping responses in a cell array and the ordering arrays in a matrix.

3. ILLUSTRATIVE ANALYSES

The following analyses are presented to illustrate basic properties of the dataset.

3.1 EEG Responses

One approach to studying beat processing using EEG involves low-frequency ($\leq$ 20 Hz) steady-state evoked potentials (SS-EPs). In an SS-EP paradigm, stimuli presented (e.g., flashed or sounded) at a particular frequency elicit brain responses at that same frequency. While SS-EPs are more often used to study vision processing [25], the approach has in recent years been used to study responses to auditory rhythms. Here, SS-EPs have shown evidence of entrainment to musical beat, peaking at beat- and meter-related frequencies even when metrical accents are imagined [26] or when beat frequencies do not dominate low-frequency stimulus spectra [27]. To our knowledge, music SS-EP studies have to date used simple, synthesized rhythmic patterns as stimuli. Our first illustrative analysis extends this approach to complex, naturalistic music.

Spatial filtering is a technique for EEG analysis whereby a weighted sum of electrodes is computed subject to some criterion [3]. Advantages of concentrating activity of interest from many electrodes to a few spatially filtered components include dimensionality reduction, improved SNR, and a reduction in multiple comparisons. For the present analysis we consider two simple spatial filters. The first is simply the mean across all electrodes (ME), which can be thought of as a constant weight applied to each electrode. For the second, we perform Principal Components Analysis (PCA), and analyze the first PC of data.

We first averaged each song's 3D electrodes-by-time-by-participant matrix across participants, producing an electrodes-by-time matrix for each song. Then, so that we analyzed the same amount of data for each song and to account for the time course of listener entrainment to the beat [9], we retained 4 minutes of data from each song, starting 15 seconds into the song.

To compute the spatial filters, we concatenated the participant-averaged data frames across all songs, producing an electrodes-by-aggregated-time matrix. Then, for the ME spatial filter, we computed the mean across electrodes, while for the PCA filter we computed electrode weightings for PC1 using Singular Value Decomposition (SVD). Finally, we reshaped each resulting song-concatenated component vector into a songs-by-time matrix. As our current interest is on SS-EPs, we present the magnitude spectrum of each component on a per-song basis.

The SS-EPs are shown in Fig. 2; y-axis scaling is consistent within each spatial filtering technique. By inspection of the plots, low frequencies (<15 Hz) of ME spectra occasionally contain peaks at frequencies in the musical beat hierarchy (e.g., Song 5). PC1 performs better, eliciting more robust spectral peaks at beat-related frequencies. Moreover, EEG PC1 appears to peak at frequencies directly related to musical beat, while suppressing many of the other spectral peaks that were observed in the magnitude spectra of stimulus amplitude envelopes (Fig. 1).

Spatial filters can be visualized by projecting the filter weights on a 2D scalp topography. While it is common to convert the spatial filter weights to a so-called "forward model," which captures the projection of filtered activity on the scalp, for PCA the spatial filter is equivalent to the forward model [28]. The ME filter, applying a constant weight to all electrodes, would reveal no spatial variation. However, the PC1 filter topography (Fig. 2, bottom right) applies a range of positive and negative weights to the electrodes, which may help to explain why this filter produces more prominent spectral peaks at beat frequencies.

3.2 Behavioral Ratings

Participant ratings of familiarity and enjoyment are shown in Fig. 3. Familiarity with the songs was low overall; ratings of enjoyment tended to be higher, and also varied more across participants.

3.3 Tapped Responses

For each trial of tapping data, we first converted each inter-tap interval to an instantaneous measure of tempo in Hz, mapped it to the midpoint of the interval, and then linearly interpolated the result to a consistent timing grid with a temporal resolution of 2 Hz. We analyze and plot data from a 17-second interval starting 15 seconds into the excerpt (i.e., starting at time 1:15 in the complete song).

The aggregate tapping responses are shown in Fig. 4. We present two visualizations of these results. First, the top figure for each song shows instantaneous tempo over the time of the excerpt for individual participants (gray curves), with the median across participants plotted in black. In bottom figures, we compute the median tempo across time for each individual participant, and summarize with histograms. Beat-related frequencies are shown in the orange ($1/2x$ tempo frequency), green (tempo frequency), and red ($2x$ tempo frequency) lines. To a large extent, participants tended to tap at what we had previously determined to be the tempo frequency. However, there are cases of lower agreement, particularly for the slowest songs (Song 1 and Song 2). Here, the histograms suggest a nearly bimodal distribution of tapped tempos, split between the computational measure and twice that, with the higher measure lying closer to what is considered the preferred tempo region for humans [23].

4. PUBLISHED DATASET

We publish the cleaned EEG data, aggregated behavioral ratings, aggregated tapped responses, and basic demographic data about the participants in Matlab `.mat` format. Example code and helper functions for the illustrative analyses are provided, also in Matlab format. Finally, we publish raw EEG recordings (for researchers who wish to apply their own preprocessing pipelines) as well as individual files of the tapped responses. The dataset is available for download from the Stanford Digital Repository [22], [4] published under a Creative Commons CC-BY license.

[4] `https://purl.stanford.edu/jn859kj8079`

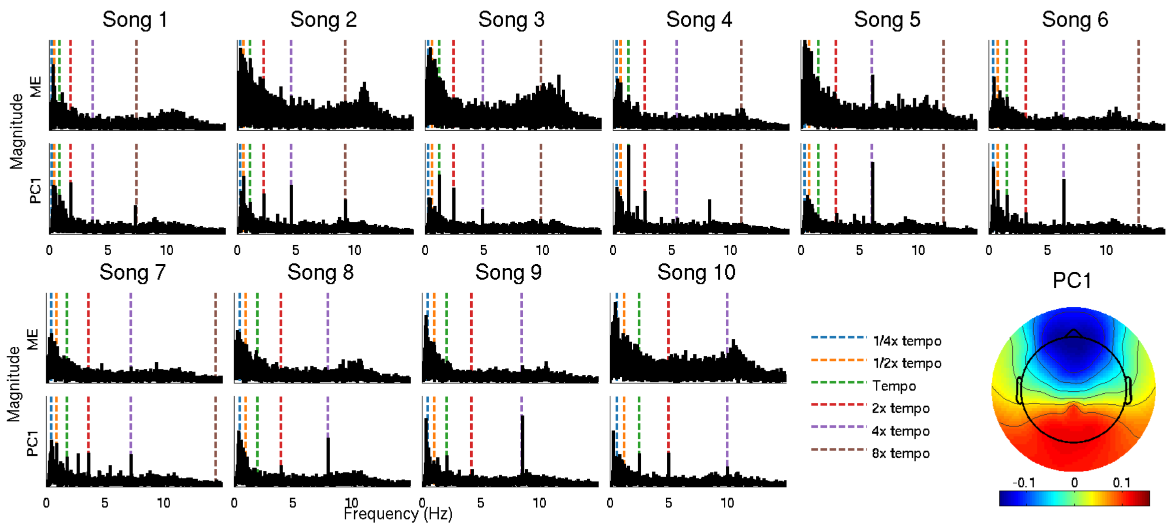


Figure 2. Low-frequency EEG spectra using a mean-electrode spatial filter (top) and PC1 spatial filter (bottom) for each song. Beat-related frequencies are shown with dashed vertical lines. Bottom right: PC1 spatial filter weights.

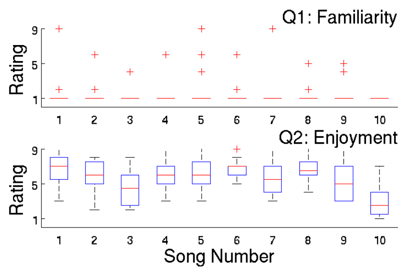


Figure 3. Participant ratings of familiarity and enjoyment.

4.1 Cleaned EEG Data

The `.mat` file `songSS_Imputed.mat` contains the cleaned EEG records, aggregated across participants, for song `SS` (§ 2.4.1). There are 10 such files, one per song. Each `.mat` file contains the following variables:

- *dataSS*: 3D electrodes-by-time-by-participant data frame. The size is $125 \times T \times 20$, with T varying according to the song.
- *subsSS*: Cell array of participant ids. Contents are the same for all songs, but are included in order to link these data to raw EEG files, raw tapping responses, and participant demographics.
- *fs*: Sampling rate, in Hz (always 125).

4.2 Raw EEG Data

We provide the raw EEG records in their exported state before preprocessing. No filtering, epoching, or cleaning has been performed. As each participant underwent two recordings, there are a total of 40 raw EEG files. The file `PP_R_raw.mat` refers to recording `R`$\in 1, 2$ from participant `PP`. Each file contains the following variables:

- *X*: Raw data frame. Size is electrodes-by-time, $129 \times T$, where T is the total length of the recording, including time periods not related to the experimental trials. The vertex reference electrode is row 129.
- *DIN_1*: Cell array containing all event labels (triggers) and times. We provide the helper function `parseDIN.m` to extract the labels and onsets into numeric vectors. Full specification on labels is provided in the `README` file accompanying the dataset.
- *fs*: Sampling rate, in Hz (always 1000).

4.3 Behavioral Ratings

Participants delivered ratings of familiarity (Q1) and enjoyment (Q2) of each song during the EEG session. The file `behavioralRatings.mat` contains a single variable *behavioralRatings*, which is a 3D participant-by-song-by-question ($20 \times 10 \times 2$) matrix.

4.4 Tapping Responses

Aggregated and raw tapping responses are stored in the file `TapIt.zip`. This archive contains the file `TapIt.mat`, which comprises the following variables:

- *allTappedResponses*: Aggregated tapped response times across all participants and songs. This is a participants-by-song (20×10) cell array. Each entry is a column vector of tap times in seconds, recorded from the device touchscreen.
- *allSongOrders*: Song-order vectors, aggregated across all participants. This is a participants-by-trial (20×10) matrix, where each row contains the stimulus presentation order for the respective participant. Numbering starts at 1.

Individual response files are also included in the `.zip` file:

- *PPP_SS.txt*: Single trial of tapped responses, in seconds, for participant `PPP` and song `SS`.
- *PPP_play_order.txt*: Stimulus presentation ordering for participant `PPP`. Numbering starts at 0.

4.5 Participant Demographics

The file `participantInfo.mat` contains a struct array *participantInfo* with participant demographics. Fields

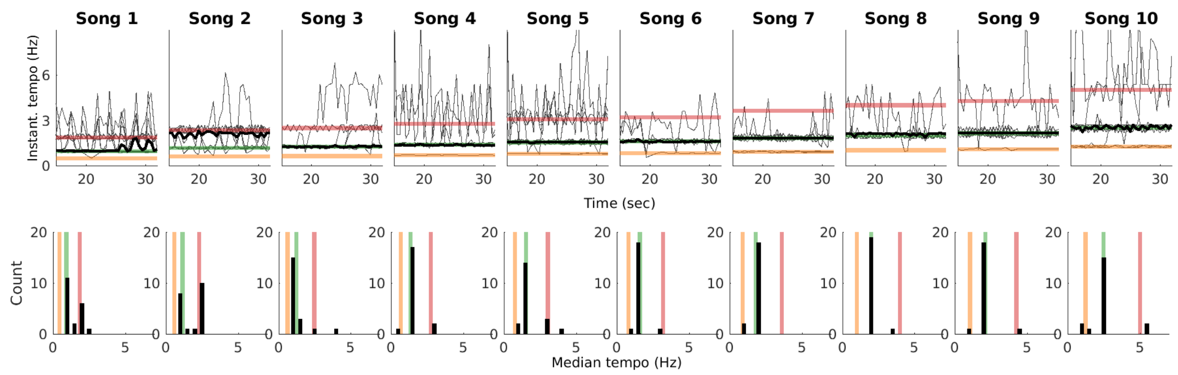


Figure 4. Tapping responses. Top: Instantaneous tempo over time for individual participants (gray), with median across participants in black. Bottom: Histograms of median tempo, over time, for individual participants. Ground-truth tempos are shown with orange ($1/2x$ tempo frequency), green (tempo frequency), and red ($2x$ tempo frequency) lines.

include *age*, *nYearsTraining*, *weeklyListening* (hours), and *id* (participant identifier link to raw filenames).

4.6 Code

The file `Code.zip` contains the Matlab scripts for the analyses performed in § 3. A variety of helper functions and files (e.g., electrode location map, script to parse the *DIN_1* variable in raw EEG files) are also provided here.

5. DISCUSSION

This paper introduces NMED-T, an open dataset of electrophysiological and behavioral responses collected from 20 participants listening to real-world musical excerpts. The published data include both raw and preprocessed dense-array EEG and tapping responses, behavioral ratings of the songs, and basic demographic information.

Our illustrative analyses validate the frequency-tagging, SS-EP approach [26, 27] with responses to complex, naturalistic music (Fig. 2). Even a simple PCA filter computed from trial-averaged responses highlights beat-related frequencies in the EEG spectra. Many PC1 spectra show prominent peaks between 5–10 Hz, regardless of tempo; future research could use this dataset to investigate further the stimulus and response attributes contributing to this phenomenon. The variability in tapping responses (Fig. 4) highlights the challenge of defining a 'ground truth' for tempo and beat identification, particularly for complex music [24]. Here we see various, sometimes conflicting results across and within participants' tapped responses. Past research has suggested that humans inherently prefer certain frequencies related to natural movement [23,35]. This may help to explain why some participants tapped at twice the tempo for the slowest songs, tending toward the postulated 2-Hz natural resonant frequency.

We faced several trade-offs when designing the study. Collection of EEG data, while relatively inexpensive [14], still incurs costs of equipment and time. Participant fatigue must also be taken into account when planning the overall duration of an experiment. As we wished to collect EEG responses to a set of full-length songs from every participant, we were limited in the number of songs we could use, and relegated the secondary tapping task to shorter excerpts. Stimulus selection, too, is often a compromise of breadth and depth. For example, the OpenMIIR dataset [31] uses shorter stimuli from a variety of genres, but at the expense of depth within any one genre; while the NMED-H [15] includes various stimulus manipulations of complete songs, but only four songs from a single genre. Our focus on full-length songs with a steady beat and a variety of tempos limited the range of genres somewhat. We also deliberately avoided massively popular songs in order to minimize possible effects, on the brain responses, of varying familiarity, established personal preferences, and autobiographical associations with the songs [12].

There are shortfalls to the dataset. One potential confound is that the EEG session always preceded the behavioral task; thus, participants were more familiar with the music during the tapping task. As a result, the tapping data may not be suitable for studying the time course of beat entrainment. However, we chose this arrangement so that participants would not be focused specifically on beat while EEG responses were recorded. Second, the tapping data show variations in tapped tempo across participants and within-participant over time. Whether this reflects our participant pool (not all trained musicians), inadequate instruction for the task, or is merely characteristic of this response is not addressed in the present illustrative analyses. Finally, listeners are known to exhibit variations in tempo octave during tapping while largely agreeing on whether a song is fast or slow [19], but we unfortunately did not collect data here to explore this distinction.

Generally speaking, this dataset facilitates research on encoding and decoding of naturalistic music. While the study design and initial analyses focused primarily on beat and tempo, the EEG responses can be analyzed in conjunction with various other stimulus features as well. Investigation of individual differences is also possible (e.g., predicting a particular participant's tapping tempo or preference rating from his or her own EEG). Other researchers might consider augmenting the dataset with complementary responses to the same songs. Ideally, the dataset will find applications in MIR and neuroscience research beyond those envisioned by the authors of this study.

6. ACKNOWLEDGMENTS

This research was supported by the Patrick Suppes Gift Fund and the Wallenberg Network Initiative: Culture, Brain, Learning. The authors thank Anthony Norcia, Dan Ellis, and the anonymous ISMIR reviewers for helpful feedback on the manuscript.

7. REFERENCES

[1] V. Alluri, P. Toiviainen, I. P. Jääskeläinen, E. Glerean, M. Sams, and E. Brattico. Large-scale brain networks emerge from dynamic processing of musical timbre, key and rhythm. *NeuroImage*, 59(4):3677–3689, 2012.

[2] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[3] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K. R. Muller. Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1):41–56, 2008.

[4] D. H. Brainard. The psychophysics toolbox. *Spatial Vision*, 10(4):433–436, 1997.

[5] R. Cabredo, R. S. Legaspi, P. S. Inventado, and M. Numao. An emotion model for music using brain waves. In *ISMIR*, pages 265–270, 2012.

[6] F. Cong, V. Alluri, A. K. Nandi, P. Toiviainen, R. Fa, B. Abu-Jamous, L. Gong, B. G. W. Craenen, H. Poikonen, M. Huotilainen, and T. Ristaniemi. Linking brain responses to naturalistic music through analysis of ongoing EEG and stimulus features. *IEEE Trans. Multimedia*, 15(5):1060–1069, 2013.

[7] A. Delorme and S. Makeig. EEGLAB: An open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134(1):9–21, 2004.

[8] D. P. W. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[9] P. Fraisse and B. H. Repp. Anticipation of rhythmic stimuli: Speed of establishment and precision of synchronization (1966). *Psychomusicology: Music, Mind, and Brain*, 22(1):84, 2012.

[10] N. Gang, B. Kaneshiro, J. Berger, and J. P. Dmochowski. Decoding neurally relevant musical features using Canonical Correlation Analysis. In *ISMIR*, 2017.

[11] J. Hockman and I. Fujinaga. Fast vs slow: Learning tempo octaves from user data. In *ISMIR*, pages 231–236, 2010.

[12] P. Janata. The neural architecture of music-evoked autobiographical memories. *Cerebral Cortex*, 19(11):2579–2594, 2009.

[13] T.-P. Jung, C. Humphries, T.-W. Lee, S. Makeig, M. J. McKeown, V. Iragui, and T. J. Sejnowski. Extended ICA removes artifacts from electroencephalographic recordings. In *NIPS*, pages 894–900, 1998.

[14] B. Kaneshiro and J. P. Dmochowski. Neuroimaging methods for music information retrieval: Current findings and future prospects. In *ISMIR*, pages 538–544, 2015.

[15] B. Kaneshiro, D. T. Nguyen, J. P. Dmochowski, A. M. Norcia, and J. Berger. Naturalistic music EEG dataset—Hindi (NMED-H). In *Stanford Digital Repository*, 2016.

[16] H.-S. Kim, B. Kaneshiro, and J. Berger. Tap-It: An iOS app for sensori-motor synchronization experiments. In *ICMPC12*, 2012.

[17] S. Koelstra, C. Muhl, M. Soleymani, J. S. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt, and I. Patras. DEAP: A database for emotion analysis using physiological signals. *IEEE Trans. Affective Computing*, 3(1):18–31, 2012.

[18] O. Lartillot and P. Toiviainen. A Matlab toolbox for musical feature extraction from audio. In *DAFx*, pages 237–244, 2007.

[19] M. Levy. Improving perceptual tempo estimation with crowd-sourced annotations. In *ISMIR*, pages 317–322, 2011.

[20] Y. P. Lin, J. R. Duann, W. Feng, J. H. Chen, and T. P. Jung. Revealing spatio-spectral electroencephalographic dynamics of musical mode and tempo perception by independent component analysis. *Journal of NeuroEngineering and Rehabilitation*, 11(1), 2014.

[21] Y. P. Lin, C. H. Wang, T. P. Jung, T. L. Wu, S. K. Jeng, J. R. Duann, and J. H. Chen. EEG-based emotion recognition in music listening. *IEEE Transactions on Biomedical Engineering*, 57(7):1798–1806, 2010.

[22] S. Losorelli, D. T. Nguyen, J. P. Dmochowski, and B. Kaneshiro. Naturalistic music EEG dataset—Tempo (NMED-T). In *Stanford Digital Repository*, 2017.

[23] D. Moelants. Preferred tempo reconsidered. In *ICMPC7*, pages 1–4, 2002.

[24] D. Moelants and M. F. McKinney. Tempo perception and musical content: What makes a piece fast, slow, or temporally ambiguous? In *ICMPC8*, pages 558–562, 2004.

[25] A. M. Norcia, L. G. Appelbaum, J. M. Ales, B. R. Cottereau, and B. Rossion. The steady-state visual evoked potential in vision research: A review. *Journal of Vision*, 15(6):4, 2015.

[26] S. Nozaradan, I. Peretz, M. Missal, and A. Mouraux. Tagging the neuronal entrainment to beat and meter. *The Journal of Neuroscience*, 31(28):10234–10240, 2011.

[27] S. Nozaradan, I. Peretz, and A. Mouraux. Selective neuronal entrainment to the beat and meter embedded in a musical rhythm. *The Journal of Neuroscience*, 32(49):17572–17581, 2012.

[28] L. C. Parra, C. D. Spence, A. D. Gerson, and P. Sajda. Recipes for the linear analysis of EEG. *NeuroImage*, 28(2):326–341, 2005.

[29] A. Sternin, S. Stober, J. A. Grahn, and A. M. Owen. Tempo estimation from the EEG signal during perception and imagination of music. In *BCMI/CMMR*, 2015.

[30] S. Stober, T. Prätzlich, and M. Meinard. Brain beats: Tempo extraction from EEG data. In *ISMIR*, pages 276–282, 2016.

[31] S. Stober, A. Sternin, A. M. Owen, and J. A. Grahn. Towards music imagery information retrieval: Introducing the OpenMIIR dataset of EEG recordings from music perception and imagination. In *ISMIR*, 2015.

[32] M. S. Treder, H. Purwins, D. Miklody, I. Sturm, and B. Blankertz. Decoding auditory attention to instruments in polyphonic music using single-trial EEG classification. *Journal of Neural Engineering*, 11(2):026009, 2014.

[33] D. M. Tucker. Spatial sampling of head electrical fields: The geodesic sensor net. *Electroencephalography and Clinical Neurophysiol.*, 87(3):154–163, 1993.

[34] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech and Audio Processing*, 10(5):293–302, 2002.

[35] L. van Noorden and D. Moelants. Resonance in the perception of musical pulse. *Journal of New Music Research*, 28(1):43–66, 1999.

PERFORMANCE ERROR DETECTION AND POST-PROCESSING FOR FAST AND ACCURATE SYMBOLIC MUSIC ALIGNMENT

Eita Nakamura
Kyoto University
enakamura@sap.ist.i.kyoto-u.ac.jp

Kazuyoshi Yoshii
Kyoto University/RIKEN AIP
yoshii@kuis.kyoto-u.ac.jp

Haruhiro Katayose
Kwansei Gakuin University
katayose@kwansei.ac.jp

ABSTRACT

This paper presents a fast and accurate alignment method for polyphonic symbolic music signals. It is known that to accurately align piano performances, methods using the voice structure are needed. However, such methods typically have high computational cost and they are applicable only when prior voice information is given. It is pointed out that alignment errors are typically accompanied by performance errors in the aligned signal. This suggests the possibility of correcting (or *realigning*) preliminary results by a fast (but not-so-accurate) alignment method with a refined method applied to limited segments of aligned signals, to save the computational cost. To realise this, we develop a method for detecting performance errors and a realignment method that works fast and accurately in local regions around performance errors. To remove the dependence on prior voice information, voice separation is performed to the reference signal in the local regions. By applying our method to results obtained by previously proposed hidden Markov models, the highest accuracies are achieved with short computation time. Our source code is published in the accompanying web page, together with a user interface to examine and correct alignment results.

1. INTRODUCTION

To computationally analyse music performances or to construct performance databases, it is needed to match notes in a music performance signal (called an *aligned signal*) to those in a reference musical score or another performance signal (*reference signal*). This process is called music alignment and automating it is a fundamental technique for music information processing and has been a popular field of research [1–16]. This study deals with offline symbolic music alignment, with particular focus on piano performances. Both score-to-MIDI alignment and MIDI-to-MIDI alignment are considered in this paper. We consider Western classical music or similar music styles where musical scores exist behind the performances.

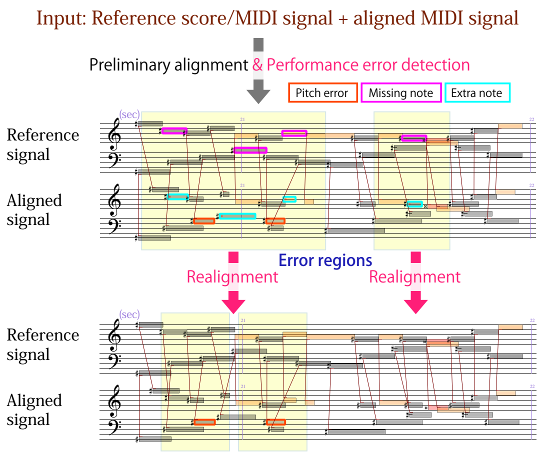


Figure 1. An outcome of the proposed method. Errors in preliminary alignment caused by reordered note pairs in the aligned signal are corrected by the realignment method.

Since music alignment between two identical performances is trivial, the central issue of automatic music alignment is to handle deviations in music performances. Possible deviations include tempo changes, performance errors (e.g. pitch errors, note insertions and deletions), ornamentation, and global structural differences (repeats and skips). To find the optimal alignment, various extensions of sequence matching methods such as hidden Markov models (HMMs) [7–9] and dynamic time warping (DTW) [1,2,6] have been studied. In the method using HMMs [7], for example, an HMM is constructed for each reference signal, in which note insertions and deletions, repeats, and skips are described by transition probabilities, and pitch errors are described by output probabilities. The aligned signal is considered as an output sequence from the HMM and the most probable sequence of latent states is estimated with the Viterbi algorithm for alignment.

It has been found that, in the case of polyphonic piano performances, deviations in performances due to asynchronies between hands/voices require special treatments [4, 5, 7, 9]. Such asynchronies result in reordering of notes with different score times, which is the main cause of alignment errors for HMMs or DTWs that are not specially designed to handle them. Models with explicit voice structure have been proposed and proved effective to solve this [4, 5, 9]. However, prior voice information of the reference signal is needed for applying these methods, which imposes limitations on usability since voice information is not given in single-channel MIDI signals and in some score file formats. Moreover, these methods have high computational cost compared to standard HMMs or DTWs [5–7, 9]. As is empirically known, those reordered notes appear only occasionally and in most cases the standard alignment methods work as accurately as the refined methods using voice information. Thus, the high computational cost would be reduced if parts of aligned signals, for which special treatments are necessary, can be selected.

Because significant deviations in music performances can usually be interpreted as performance errors, alignment errors are often connected with performance errors. For example, a pair of extra and missing notes as in Fig. 1 typically appear as a result of alignment errors. Based on the authors' experience, displaying performance errors enables human annotators to easily find alignment errors and greatly improves the efficiency of examining and correcting automatic alignment results. Likewise, by detecting performance errors in a given result of automatic alignment, it would be possible to select limited regions in the aligned signal that may contain alignment errors.

Based on these observations, this study aims to develop an automatic post-processing method for correcting given symbolic music alignment results. We first develop a performance error detection algorithm that recognises pitch errors, extra notes, and missing notes in a given alignment result. *Error regions* are then defined as segments of aligned and reference signals around performance errors and we investigate how much alignment errors are contained in these regions with various sizes of the regions. Next we develop a post-processing *realignment* method that can handle hand/voice asynchrony based on a voice-structured model. Since both music alignment and recognition of performance errors involve searches for an optimal choice among possible candidate solutions, we formulate them based on statistical models whose parameters can be optimised from data. To construct a realignment method that does not require prior voice information, we combine the method using merged-output HMMs [9] with a voice (hand) separation method [17]. For concreteness, we use as a preliminary alignment method the one based on temporal HMMs [7]. The results of the proposed method are evaluated in comparison with the state-of-the-art methods.

The contributions of this study are as follows. First, our alignment method achieves the highest accuracy and its computational cost is much smaller than previous methods with comparable accuracies. The method works without prior voice information and can be applied for a wide class of performance and score data. The source code for our algorithms and a user interface to examine and correct the results is published in the accompanying web page [18]. To our knowledge, this is currently the only publicly available alignment tool of comparable accuracies. Second, this is the first paper that quantitatively investigates the relation between performance errors and alignment errors, which can be used generally to reduce high computational cost that is typically required in elaborated methods. Lastly, our alignment algorithm of the merged-output HMMs yields better sub-optimisation than a previous one [9].

1.1 Current State-of-the-Art Methods

The method by Gingras and McAdams [5] (*GM algorithm*), which takes into account the voice structure and timing information, is regarded as one of the most accurate methods for symbolic music alignment, with 99.978% of accuracy on their data. The method based on the temporal HMM by Nakamura et al. [7] (*NOSW algorithm*) also uses the timing information but not the voice structure. The method can handle arbitrary repeats and skips, but the accuracy was lower than the GM algorithm in a direct comparison. For online alignment, the method using merged-output HMMs for incorporating the voice structure had better accuracies than the temporal HMMs [9].

Recently, Chen et al. [6] reported a significant lower accuracy ($\leq 91.93\%$) for the GM algorithm on other data and proposed a method based on DTW (*CJL1 algorithm*) with a better accuracy ($\leq 98.51\%$)[1]. Another method (*CJL2 algorithm*) is proposed in the paper, which is less accurate but more efficient than the CJL1 algorithm. The CJL algorithms neither use voice information nor have a special architecture to utilise the voice structure.

2. PERFORMANCE ERROR DETECTION

2.1 Problem Statement

Both reference and aligned signals can be represented as a sequence of musical notes (called *reference notes* and *aligned notes*) with a pitch and an onset time described as physical or score time. For MIDI-to-MIDI alignment, the reference signal can be a performed MIDI signal that has (almost) continuous onset times. In this case, we cluster notes according to onset times to obtain a reference signal with quantised onset times, which enables us to discuss score-to-MIDI and MIDI-to-MIDI alignment in a unified way. Specifically, we put a threshold of 35 ms, which is known to well discriminate chordal notes [19], to form clusters of notes and then quantise onset times (e.g. in units of ms etc.). An *alignment result* is a sequence of labels that indicates for each aligned note the corresponding reference note. If there is no corresponding note (as is the case for extra notes), a distinguished label 'EXTRA' is given.

As performance errors we consider pitch errors, extra notes, and missing notes. Extra notes are aligned notes that are not matched to any of the reference notes and missing notes are reference notes that do not appear in the aligned signal. In this study we consider the *strict alignment*, for which each reference note can be matched to at most one aligned note[2]. For a strict alignment result, performance errors are automatically determined: aligned notes without corresponding reference notes are extra notes; aligned notes with pitches different from the corresponding reference notes have pitch errors; reference notes not appearing

[1] A different evaluation measure was used in Ref. [6] and these upper bounds have been derived as conservative limits.

[2] This condition must be relaxed and apply only locally if we allow global repeats and skips in the aligned signal. In addition, trills and tremolos are exceptions where multiple aligned notes correspond to each reference note. For simplicity and for the lack of space, we concentrate on the case without ornaments, repeats, and skips in this paper.

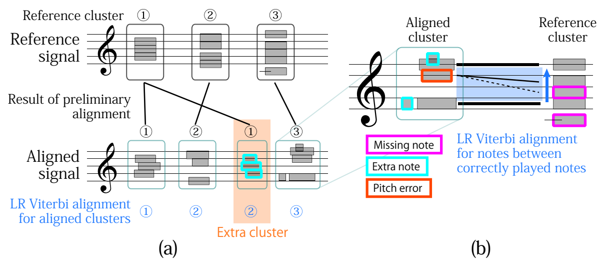


Figure 2. Steps for performance error detection. After (a) cluster-wise LR Viterbi alignment is performed, (b) note-wise LR Viterbi alignment is performed for each cluster.

in the alignment result are missing notes.

Standard alignment methods such as HMMs and DTWs often output alignment results that are not strict (so that a reference note can appear more than once) and performance error indications are not given. Therefore, the aim of performance error detection is to obtain a strict alignment result from a non-strict alignment result, which is equivalent to identifying extra notes in the latter one. In fact, our method uses only the information of matched score times for each note in an input alignment result.

2.2 Model

Our approach for the identification of extra notes is to carry out two left-to-right (LR) Viterbi alignments, first in units of 'chords' and second in units of notes within each 'chord' (Fig. 2). To be precise, we define a *reference cluster* as a set of all notes with the same score time in the reference signal. Aligned notes are clustered so that all successive notes form a cluster (*aligned cluster*) as long as their reference labels are in the same onset cluster. The first LR Viterbi alignment is then performed on the sequence of aligned clusters and those clusters assigned a reference cluster different from the original one are identified as extra clusters. Note that after this procedure each non-extra aligned cluster is matched to a unique reference cluster.

In the next step, extra notes in each (non-extra) aligned cluster are identified. Based on our intuition that aligned notes with correct pitches play a pivot role and the assigned reference labels should respect the pitch order, we first identify aligned notes with correct pitches and then match other notes, which are either extra notes or notes with pitch errors. Since in general there are multiple notes with the same pitch in one aligned cluster, the onset time information should be used here. As a reference point of onset time, the expected onset time $\tilde{t}$ of the reference onset cluster is computed by local averaging, similarly as tempo estimation [19]. If there are multiple candidates with the correct pitch, the one with an onset time nearest to $\tilde{t}$ is chosen and the other candidates are identified as extra notes.

Let $(q_1, \dots, q_C)$ denote an ordered set of notes in the concerned reference cluster, where q_c is the integral pitch of the c-th note and satisfies $q_1 \le q_2 \le \dots \le q_C$, and let $Q^{\rm corr}$ denote the set of reference notes matched to aligned notes with correct pitches. Similarly, let us order notes in the concerned aligned cluster according to pitch first and then onset time. Denoting the pitch and onset time of the b-th note by p_b and t_b, we thus have for all $b \in \{1, \dots, B\}$ (B is the number of notes in the aligned cluster) $p_{b-1} \le p_b$ and $t_{b-1} \le t_b$ if $p_{b-1} = p_b$. Now suppose that a pair of pivot notes (c, c') $(c, c' \in \{1, \dots, C\})$ satisfies that $q_c, q_{c'} \in Q^{\rm corr}$, $q_c < q_{c'}$, and $q_j \notin Q^{\rm corr}$ for each q_j with $c < j < c'$. For such a pair we define $Q = \{q_j \,|\, c < j < c'\}$ and $S = \{b \in \{1, \dots, B\} \,|\, q_c < p_b < q_{c'}\}$. The next step is to match Q and S for each pair (c, c') of pivot notes. For aligned notes with pitches higher or lower than the highest or lowest pivot note, we can similarly define Q and S as half-bounded sets, and for the case with no pivot notes, we define $Q = \{1, \dots, C\}$ and $S = \{1, \dots, B\}$, and carry out the following procedure.

The matching is trivial when $\#Q \le 1$ and $\#S \le 1$. In other cases, multiple interpretations of pitch errors exist and some principle must be introduced to find the optimal choice (Fig. 2(b)). We solve this optimisation problem with a statistical performance model including temporal fluctuations and pitch errors, similar to the model in Ref. [7]. The mapping $z : Q \ni j \mapsto z_j \in S$ is optimised by LR Viterbi alignment with the following probability:

$$P(z_j = b \,|\, z_{j-1} = b') = \theta_{b'b}\, \psi^{\rm pitch}(p_b - q_b)\psi^{\rm time}(t_b - \tilde{t})$$

where θ is a $\#S \times \#S$ LR transition probability matrix,

$$\theta_{b'b} = \begin{cases} 1/\#\{l \in S \,|\, l > b'\}, & b' < b; \\ 0, & \text{otherwise}, \end{cases} \tag{1}$$

and $\psi^{\rm pitch}(\delta p)$ is the probability of pitch errors in δp semitones (given by Eq. (30) of Ref. [19]), and $\psi^{\rm time}(\delta t)$ is the probability of onset time fluctuation given as

$$\psi^{\rm time}(\delta t) = \mathsf{N}(\delta t; 0, \rho^2). \tag{2}$$

Here, $\mathsf{N}(\,\cdot\,; \mu, \Sigma)$ denotes a normal distribution with mean μ and variance Σ. The value of ρ is taken as 100 ms in our implementation. Aligned notes without matched reference notes are classified as extra notes.

2.3 Error Regions and Alignment Errors

Having identified the performance errors, we now define error regions in the aligned signal around them. To do this, we first calculate the *synchronised onset time* for each reference cluster by averaging onset times of corresponding aligned notes, or if there are no such notes, by interpolating/extrapolating neighbouring synchronised onset times. We consider, for each extra note n with onset time t_n, a time interval of the form $[t_n - \Delta, t_n + \Delta)$ with width Δ (called an extra note region) and construct the set $\mathcal{R}_{\rm e}$ of such time intervals for all extra notes. Likewise, the set of pitch error regions $\mathcal{R}_{\rm p}$ is constructed. The set of missing note regions $\mathcal{R}_{\rm m}$ is similarly constructed by using the synchronised onset times to define each time region. Finally, the error region $\mathcal{R}$ is constructed by combining elements in $\mathcal{R}_{\rm e}$, $\mathcal{R}_{\rm p}$, and $\mathcal{R}_{\rm m}$. If there are overlapping time regions, they are expanded/unified to one time region at this step (Fig. 3). Thus, $\mathcal{R}$ is a set $\{[t_r, t'_r)\}_{r=1}^{N_{\mathcal{R}}}$ of non-overlapping

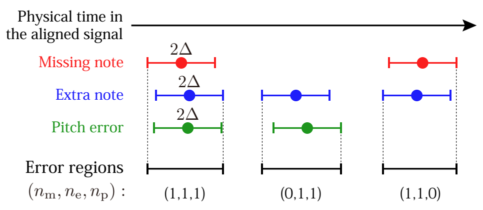

Physical time in the aligned signal
Missing note
Extra note
Pitch error
2Δ
Error regions
$(n_\mathrm{m}, n_\mathrm{e}, n_\mathrm{p})$:
(1,1,1)
(0,1,1)
(1,1,0)

Figure 3. Examples of error regions and their indices.

time regions where we have $t'_{r-1} < t_r$ for all r. The number of missing notes, extra notes, and pitch errors in each region r is denoted by $n_\mathrm{m}(r)$, $n_\mathrm{e}(r)$, and $n_\mathrm{p}(r)$.

Let us discuss the relation between performance errors and alignment errors. An *alignment error* is defined as a reference label in an alignment result that is different from the ground-truth label. We say that an alignment error is contained in the error regions if the onset time of the incorrectly aligned note is contained in one of the regions in $\mathcal{R}$. The proportion of alignment errors contained in the error regions for varying time width Δ, calculated for alignment results of the temporal HMM on the three datasets explained in Sec. 4, is shown in Fig. 4, together with the proportion of aligned notes contained in the error regions. More than 90% of the performance errors are contained in the error regions for Δ as small as 0.1 s, while contained aligned notes remain less than 20% for $\Delta \leq 0.3$ s.

For the alignment errors to be corrected by realignment carried out on each error region, not only incorrectly aligned notes but also their corresponding reference notes must be contained in the region. To be precise, for each time region $[t_r, t'_r)$ in $\mathcal{R}$, we choose segments of the aligned and reference signals and use them as the aligned and reference signals for realignment. For the segment of the aligned signal, the subsequence of aligned notes whose onset times belong to the time region is used. For these aligned notes, we obtain the maximal and minimal score times (τ_max and τ_min) of corresponding reference notes. The subsequence of all reference notes whose onset score times are in the range $[\tau_\mathrm{min}, \tau_\mathrm{max}]$ is used as the segment of the reference signal. We call an alignment error in an error region *correctable* if its ground-truth label is 'EXTRA' or is a reference note in the reference signal segment. We see in Fig. 4 that the proportion of correctable errors increases rapidly for $\Delta < 0.3$ s and gradually for $\Delta > 0.3$ s.

Although it is not always the case, naively we expect that the number of performance errors is reduced when alignment results are corrected, as is evident in the case of correcting a mismatched pair of missing and extra notes. On the other hand, if only one performance error exists or only missing notes exist in an error region, the number of performance errors cannot be reduced by realigning notes. By expanding this idea, we can impose conditions on error regions so that most of the alignment errors remain contained in the selected error regions but the contained alignment notes are reduced significantly. Results in Table 1, where error regions were imposed the condition of containing at least two types of performance errors, show an example of this fact. Such conditions can be used to increase the efficiency of realignment, as we see in Sec. 4.

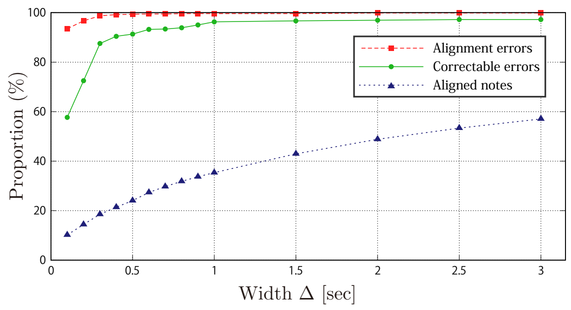


Figure 4. Proportion of alignment errors and aligned notes contained in the error regions.

Conditions	Alignment errors	Correctable errors	Aligned notes
None	98.7%	87.5%	18.5%
$n_\mathrm{m}n_\mathrm{e}+n_\mathrm{e}n_\mathrm{p}+n_\mathrm{p}n_\mathrm{m} > 0$	88.5%	80.3%	9.2%

Table 1. Same as Fig. 4 with and without imposed conditions on the error regions ($\Delta = 0.3$ s).

3. REALIGNMENT

Here we develop a realignment method based on merged-output HMMs, which is applied to the error regions to correct the preliminary alignment result. The overall procedure of realignment is illustrated in Fig. 5. We first apply hand separation for the reference signal segment to estimate the voice structure and then carry out alignment based on the merged-output HMM using the estimated voices.

3.1 Hand Separation

To formulate a method that does not require prior voice information, we apply voice separation to each reference signal segment. Because voice asynchrony in piano performances usually appears between the left- and right-hand parts and a larger number of voices increases the computational cost for realignment, we use a technique that separates a performance signal into two hand parts [17].

Voice information is described with a binary variable s_m for each note m in the reference signal segment. If $s_m = L$ (or R), the m-th note is in the left-hand (or right-hand) part. Let us denote the pitches of the reference signal segment by $\boldsymbol{x} = x_{1:M} = (x_1, \ldots, x_M)$, where the notes are ordered according to the onset score time. (Similar notations appear throughout the paper.) To estimate the sequence $\boldsymbol{s} = s_{1:M}$ from the input $\boldsymbol{x}$, we construct a merged-output HMM. The Markov model for each voice is described with transition probabilities on pitches, denoted by $\chi^L_{yy'}$ and $\chi^R_{yy'}$. Introducing pitch variables for the two voices, y^L_m and y^R_m for each m, the latent state variable for the merged-output HMM is given as $Y_m = (s_m, y^L_m, y^R_m)$ and the transition and output probabilities are given as

$$P(Y_m = Y \,|\, Y_{m-1} = Y') \\ = \tfrac{1}{2}(\delta_{sL}\, \chi^L_{y'^L y^L}\, \delta_{y'^R y^R} + \delta_{sR}\, \chi^R_{y'^R y^R}\, \delta_{y'^L y^L}), \quad (3)$$

$$P(x_m \,|\, Y_m = Y) = \delta_{sL}\delta_{y^L x_m} + \delta_{sR}\delta_{y^R x_m}, \quad (4)$$

where $\delta_{yy'}$ denotes Kronecker's delta. To complete the

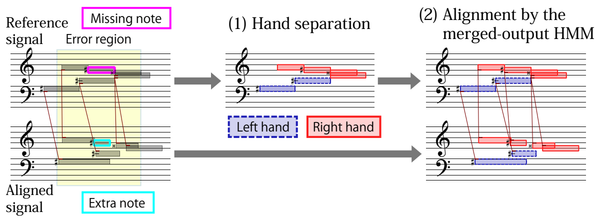


Figure 5. The realignment step consists of hand separation and alignment by the merged-output HMM.

stochastic process, we should specify the initial probability, which is given similarly as in Eq. (3) with initial pitch values denoted by y_0^L and y_0^R. We use as y_0^L and y_0^R the lowest and highest pitch in the reference signal segment. We can estimate $\boldsymbol{s}$ with the maximal posterior probability using an efficient Viterbi algorithm [17].

3.2 Realignment Based on Merged-Output HMM

For realignment we use the merged-output HMM proposed previously [9]. Modifications to the model are introduced to reduce computational cost and an inference algorithm that is more rigorous than the original one is derived.

Let us first briefly review the temporal HMM [7] for music alignment. The aligned signal (segment) can be can be described as a sequence $(\boldsymbol{p}, \boldsymbol{t})$, where $\boldsymbol{p} = (p_1, \dots, p_N)$ denotes the integral pitches and $\boldsymbol{t} = (t_1, \dots, t_N)$ denotes the onset times (N is the number of aligned notes). The reference signal (segment) is represented as a sequence of reference clusters indexed by $i \in \{1, \dots, I\}$ (I is the number of reference clusters), and the corresponding onset score time is denoted by τ_i. Local tempos are denoted by $\boldsymbol{v} = (v_1, \dots, v_N)$. The corresponding reference cluster of the n-th aligned note is denoted by $i_n \in \{1, \dots, I\}$. The latent state of the temporal HMM is indexed by (i_n, v_n) for each $n \in \{1, \dots, N\}$ and the output symbol is the pair (p_n, t_n). Transition and output probabilities are given as

$$\begin{aligned}
&P(i_n, v_n \,|\, i_{n-1}, v_{n-1}) = \pi(i_{n-1}, i_n)\mathsf{N}(v_n; v_{n-1}, \sigma_v^2),\\
&P(p_n | i_n) = \phi(i_n, p_n), \qquad (5)\\
&P(t_n \,|\, t_{n-1}, i_{n-1} = i', i_n = i, v_n)\\
&\quad = (1 - \delta_{ii'})\mathsf{N}(t_n; t_{n-1} + v_n(\tau_i - \tau_{i'}); \sigma_t^2)\\
&\qquad + \delta_{ii'}\mathsf{Exp}(t_n - t_{n-1}; \lambda), \qquad (6)
\end{aligned}$$

where we have assumed the statistical independence for the pairs i_n and v_n, and p_n and t_n. The probability π stochastically describes how the performance proceeds in the reference signal. The standard deviation σ_v represents the amount of tempo variation during the performance. The pitch output probability ϕ stochastically describes pitch errors (similarly as ψ^{pitch} in Sec. 2.2); it depends on the pitch context of reference cluster i. The form of the output probability for onset times reflects the fact that inter-onset intervals between chordal notes obey an exponential distribution (denoted by Exp) and those between onset clusters are approximately given as the product of the local tempo and the score time interval [7]. The scale parameter λ and the standard deviations σ_t and σ_v have been measured [7].

We can now construct the merged-output HMM for music alignment using voice information, by describing each voice by the temporal HMM and merging outputs from the two HMMs [9]. To reduce computational cost, we introduce two simplifications to the model. First, since the error region is considered to span a small time range (less than a few seconds), the variation of tempos should be relatively small. We therefore assume a constant tempo v in each error region, which can be obtained from the preliminary alignment result. This removes the dynamics of tempos and reduces the state space of the temporal HMM to that indexed only by i_n. Second, again because of the locality of error regions, we can assume LR transition probabilities for π. This reduces the number of possible state transition paths and thus reduces the computational cost. With these simplifications, the state space of the merged-output HMM is indexed by $k = (s, i^L, i^R, t^L, t^R)$ ($s \in \{L, R\}$) and the transition and output probabilities are

$$\begin{aligned}
&P(k_n = k \,|\, k_{n-1} = k')\\
&\quad = \tfrac{1}{2}A_s(i^s, t^s \,|\, i'^s, t'^s, v)\\
&\qquad \cdot \left[\delta_{sL}\delta_{i'^R i^R}\delta(t'^R - t^R) + \delta_{sR}\delta_{i'^L i^L}\delta(t'^L - t^L)\right],\\
&A_s(i^s, t^s \,|\, i'^s, t'^s; v) = \pi(i'^s, i^s)P(t'^s \,|\, t^s, i'^s, i^s, v), \qquad (7)\\
&P(p_n \,|\, k_n = k) = \phi(i^s, p_n), \qquad (8)\\
&P(t_n \,|\, k_n = k) = \delta(t_n - t^s), \qquad (9)
\end{aligned}$$

where $\delta(\,\cdot\,)$ is the Dirac delta function. Notating $\boldsymbol{k} = k_{1:N}$, the complete-data probability is given as

$$P(\boldsymbol{k}, \boldsymbol{p}, \boldsymbol{t}) = \prod_{n=1}^{N} P(k_n|k_{n-1})P(p_n|k_n)P(t_n|k_n). \quad (10)$$

The alignment result is obtained by inferring $\boldsymbol{k}$ that maximises $P(\boldsymbol{k}, \boldsymbol{p}, \boldsymbol{t})$. The direct application of the Viterbi algorithm is impossible since the temporal HMM is of autoregressive type, i.e. the output probability of onset times depends on past values. Instead of the rough sub-optimisation method used in Ref. [9], we use a trick of introducing an auxiliary variable that encodes the historical path, as in Ref. [20], which enables almost exact optimisation. Introduce $h_n = 1, 2, \cdots$, which is defined as the smallest $h \geq 1$ satisfying $s_n \neq s_{n-h}$ for each n. We have

$$h_n = \begin{cases} h_{n-1}{+}1, & s_n = s_{n-1};\\ 1, & s_n \neq s_{n-1}, \end{cases} \quad t_n^s = \begin{cases} t_n, & s = s_n;\\ t_{n-h_n}, & s \neq s_n. \end{cases}$$

With a change of variables ($\boldsymbol{h} = h_{1:N}$, $\boldsymbol{i}^L = i_{1:N}^L$, etc.),

$$\begin{aligned}
&P(\boldsymbol{k}, \boldsymbol{p}, \boldsymbol{t}) = P(\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{i}^L, \boldsymbol{i}^R, \boldsymbol{p}, \boldsymbol{t})\\
&= \prod_n \Bigg\{ \tfrac{1}{2}\left[\delta_{s_n L}\delta_{i'^R i^R} + \delta_{s_n R}\delta_{i'^L i^L}\right]\\
&\cdot \left[\delta_{s_n s_{n-1}}\delta_{h_n (h_{n-1}+1)}A_n^{\text{same}} + (1 - \delta_{s_n s_{n-1}})\delta_{h_n 1}A_n^{\text{diff}}\right]\Bigg\},\\
&A_n^{\text{same}} = A_{s_n}(i_n^{s_n}, t_n \,|\, i_{n-1}^{s_n}, t_{n-1}; v),\\
&A_n^{\text{diff}} = A_{s_n}(i_n^{s_n}, t_n \,|\, i_{n-1}^{s_n}, t_{\tilde{n}}; v) \qquad (11)
\end{aligned}$$

where $\tilde{n} = n - h_{n-1} - 1$. It is now possible to derive the Viterbi algorithm for the state space of $(\boldsymbol{s}, \boldsymbol{h}, \boldsymbol{i}^L, \boldsymbol{i}^R)$. A

Algorithm	GM data	CJL data	Our data
Proposed	$\mathbf{0.18 \pm 0.08}$	$\mathbf{0.79 \pm 0.06}$	$\mathbf{0.48 \pm 0.03}$
NOSW+	1.46 ± 0.23	1.81 ± 0.08	0.64 ± 0.04
NOSW [7]	1.78 ± 0.25	2.33 ± 0.10	2.24 ± 0.07
GM [5]	0.28 ± 0.10 [7]	$8.07^{\dagger} \pm 0.18$ [6]	N/A
CJL1 [6]	N/A	$1.49^{\dagger} \pm 0.08$ [6]	N/A
CJL2 [6]	N/A	$2.20^{\dagger} \pm 0.09$ [6]	N/A

Table 2. Alignment error rates (%) with 1σ statistical errors. The best values within 1σ significance are displayed in bold font. Daggers indicate lower bounds (see Sec. 1.1).

cutoff (~ 50) on the maximum value of h_n can be put to reduce the search space with little loss of optimality [20]. Finally, the performance error detection described in Sec. 2 is performed separately on each voice (hand part).

During testing the method, we noticed that alignment errors as simple as a pair of missing and extra notes as in Fig. 1 sometimes remain after applying the described realignment step. This is often because the result of hand separation is not completely correct. To handle this, we carried out a simple processing step (called *pairing step*) of matching trivially corresponding missing and extra note pairs. For each missing note, an extra note with the same pitch is searched within the time region of half width Δ, and if found, they are matched. The pairing step can also be applied before the realignment step to correct trivial alignment errors and thus reduce the cost of realignment.

4. EVALUATION

As explained in Sec. 1.1, the state-of-the-art methods are the NOSW algorithm [7], the GM algorithm [5], and the CJL algorithms (CJL1 and CJL2) [6]. For comparison, we run the performance error detection on the results of the NOSW algorithm (*NOSW+ algorithm*), and the proposed realignment was applied to its results. Since the GM and CJL algorithms were not available from their authors but the used data were provided, we run the proposed method and temporal HMM on their data and directly compared the accuracies. The *GM data* consisted of seven performances of two excerpts of Chopin's piano pieces (total of 2,815 aligned notes). The *CJL data* consisted of 21 pairs of piano MIDI files (total of 25,656 aligned notes), most of which are synthetic (not human-played) performances. We also tested the proposed method on the human-played performance data that we prepared. Our data consisted of 60 excerpts of classical piano pieces each played by three different pianists (total of 43,608 aligned notes) [3] . For the GM data and our data it was score-to-MIDI alignment and for the CJL data it was MIDI-to-MIDI alignment. For the proposed method, $\Delta = 0.3$ s was used, error regions satisfying the condition ($n_{\rm m}n_{\rm e} + n_{\rm e}n_{\rm p} + n_{\rm p}n_{\rm m} > 0$) were selected for realignment, and the pairing step was applied before and after the realignment step.

The rates of alignment errors in Table 2 show that for all data the realignment method significantly improved the preliminary alignment results: in total 47% ($= 369$ aligned notes) of alignment errors made by the NOSW+ algorithm were reduced. The proposed method had the highest accuracies for all datasets. To evaluate computational efficiency, the processing time was measured. Our algorithms were implemented in C++ on a computer with 3.1 GHz CPU and 16 GB memory running Mac OS X 10.11. The measured time for the CJL data was 8.25 s for the NOSW algorithm, 17.76 s for the performance error detection, and 1.12 s for the realignment. Compared to the reported values [6], 342.70 s and 3535.36 s for the CJL1 and GM algorithm, the computational efficiency of the proposed method is evident, although direct comparison is not possible because of different computer environments. Examples demonstrating the effect of the realignment method are shown in the accompanying web page [18].

[3] The data could be provided upon requests to the authors.

	GM data	CJL data	Our data	Time (s)
(a)	0.18	0.79	0.48	5.54 ± 0.07
(b)	0.25	1.17	0.51	6.31 ± 0.07
(c)	0.14	0.85	0.61	6.32 ± 0.05

Table 3. Alignment error rates (%) and processing time (averaged over five trials) for the proposed method with (a) both paring steps and conditions on error regions, (b) only conditions on error regions, and (c) only pairing steps.

To examine the effect of the paring step and the conditions imposed on error regions, the proposed method without these modifications was compared in terms of accuracies and processing time for all data (Table 3). In addition to the expected reduction of computation time, these modifications were also effective in reducing overall alignment errors. This suggests that the realignment by the merged-output HMM increases alignment errors in some error regions and these modifications have effects in avoiding this. Detailed analyses are currently being undertaken.

5. CONCLUSION

We have described a realignment method for symbolic music signals based on merged-output HMMs, which can deal with reordered notes due to voice asynchrony. To reduce the high computational cost, performance errors are detected and the merged-output HMMs are applied to regions around the performance errors rather than to the whole signal. In all tested data and for both score-to-MIDI and MIDI-to-MIDI alignment cases, the proposed realignment method combined with an HMM-based method achieved the highest accuracies, with short computation time.

The principle of using performance errors to select regions in the aligned signals that possibly contain alignment errors is generally applicable to save computation time. For example, when a further refined alignment method is found in the future, we can apply it to the error regions of the results by the proposed method, instead of doing alignment from scratch. In addition, since the realignment can be done locally, it can be applied to performance signals with global repeats and skips [7]. For future work, refinements for the model for performance error detection by examining human-annotated data would be possible to further improve the accuracy and efficiency.

6. ACKNOWLEDGEMENTS

We are grateful to Bruno Gingras and Chun-Ta Chen for providing evaluation data. E. Nakamura is supported by the JSPS Postdoctoral Research Fellowship and this studied was done when he was a visiting researcher at the Queen Mary University of London. This work is in part supported by JSPS KAKENHI Nos. 24220006, 26280089, 26700020, 15K16054, 16H01744, 16H02917, 16K00501, and 16J05486 and JST ACCEL No. JPMJAC1602.

7. REFERENCES

[1] R. Dannenberg, "An On-Line Algorithm for Real-Time Accompaniment," *Proc. ICMC*, pp. 193–198, 1984.

[2] J. Bloch and R. Dannenberg, "Real-Time Computer Accompaniment of Keyboard Performances," *Proc. ICMC*, pp. 279–290, 1985.

[3] P. Desain, H. Honing, and H. Heijink, "Robust Score-Performance Matching: Taking Advantage of Structural Information," *Proc. ICMC*, pp. 337–340, 1997.

[4] H. Heijink, L. Windsor, and P. Desain, "Data Processing in Music Performance Research: Using Structural Information to Improve Score-Performance Matching," *Behavior Research Methods, Instruments, & Computers*, vol. 32, no. 4, pp. 546–554, 2000.

[5] B. Gingras and S. McAdams, "Improved Score-Performance Matching Using Both Structural and Temporal Information from MIDI Recordings," *J. New Music Res.*, vol. 40, no. 1, pp. 43–57, 2011.

[6] C. Chen, J. R. Jang, and W. Liou, "Improved Score-Performance Alignment Algorithms on Polyphonic Music," *Proc. ICASSP*, pp. 1365–1369, 2014.

[7] E. Nakamura, N. Ono, S. Sagayama, and K. Watanabe, "A Stochastic Temporal Model of Polyphonic MIDI Performance with Ornaments," *J. New Music Res.*, vol. 44, no. 4, pp. 287–304, 2015.

[8] B. Pardo and W. Birmingham, "Modeling Form for On-line Following of Musical Performances," *Proc. NCAI*, 2005.

[9] E. Nakamura, Y. Saito, N. Ono, and S. Sagayama, "Merged-Output Hidden Markov Model for Score Following of MIDI Performance with Ornaments, Desynchronized Voices, Repeats and Skips," *Proc. Joint ICMC|SMC 2014*, pp. 1185–1192, 2014.

[10] C. Raphael, "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models," *IEEE TPAMI*, vol. 21, no. 4, pp. 360–370, 1999.

[11] A. Cont, "A Coupled Duration-Focused Architecture for Realtime Music to Score Alignment," *IEEE TPAMI*, vol. 2, no. 6, pp. 974–987, 2010.

[12] C. Fremerey, M. Müller, and M. Clausen, "Handling Repeats and Jumps in Score-Performance Synchronization," *Proc. ISMIR*, pp. 243–248, 2010.

[13] C. Joder, S. Essid, and G. Richard, "A Conditional Random Field Framework for Robust and Scalable Audio-to-Score Matching, *IEEE TASLP*, vol. 19, no. 8, pp. 2385–2397, 2011.

[14] M. Grachten, M. Gasser, A. Arzt, and G. Widmer, "Automatic Alignment of Music Performances with Structural Differences," *Proc. ISMIR*, pp. 607–612, 2013.

[15] A. Maezawa, K. Itoyama, K. Yoshii, and H. G. Okuno, "Bayesian Audio Alignment Based on a Unified Model of Music Composition and Performance," *Proc. ISMIR*, pp. 233–238, 2014.

[16] W. Siying, S. Ewert, and S. Dixon, "Robust and Efficient Joint Alignment of Multiple Musical Performances," *IEEE/ACM TASLP*, vol. 24, no. 11, pp. 2132–2145, 2016.

[17] E. Nakamura, N. Ono, and S. Sagayama, "Merged-Output HMM for Piano Fingering of Both Hands," *Proc. ISMIR*, pp. 531–536, 2014.

[18] E. Nakamura, K. Yoshii, and H. Katayose, *Symbolic Music Alignment Tool*, 2017. `http://anonym9382.github.io/demo.html` [Online]

[19] E. Nakamura, T. Nakamura, Y. Saito, N. Ono, and S. Sagayama, "Outer-Product Hidden Markov Model and Polyphonic MIDI Score Following," *J. New Music Res.*, vol. 43, no. 2, pp. 183–201, 2014.

[20] 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.

PIECE IDENTIFICATION IN CLASSICAL PIANO MUSIC WITHOUT REFERENCE SCORES

Andreas Arzt, Gerhard Widmer
Department of Computational Perception, Johannes Kepler University, Linz, Austria
Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
andreas.arzt@jku.at

ABSTRACT

In this paper we describe an approach to identify the name of a piece of piano music, based on a short audio excerpt of a performance. Given only a description of the pieces in text format (i.e. no score information is provided), a reference database is automatically compiled by acquiring a number of audio representations (performances of the pieces) from internet sources. These are transcribed, preprocessed, and used to build a reference database via a robust symbolic fingerprinting algorithm, which in turn is used to identify new, incoming queries. The main challenge is the amount of noise that is introduced into the identification process by the music transcription algorithm and the automatic (but possibly suboptimal) choice of performances to represent a piece in the reference database. In a number of experiments we show how to improve the identification performance by increasing redundancy in the reference database and by using a preprocessing step to rate the reference performances regarding their suitability as a representation of the pieces in question. As the results show this approach leads to a robust system that is able to identify piano music with high accuracy – without any need for data annotation or manual data preparation.

1. INTRODUCTION

Efficient algorithms for content-based audio retrieval enable systems that allow users to browse and explore music collections (see e.g. [10] for an overview). In this context *audio fingerprinting* algorithms which permit the fast identification of an unknown recording (as long as an almost *exact replica* is contained in the reference database) play an important role. For this task there exist highly efficient algorithms that are in everyday commercial use (see e.g., [3, 6, 13, 15–17]).

However, these algorithms are not able to identify *different performances* of the *same piece* of music, as they are not designed to work in the face of musical variations such as different tempi, expressive timing, differences in instrumentation, ornamentation and other performance aspects. Regarding classical music, the identification of performances that derive from a common musical score is of special interest, as in general there exists a large number of performances of the same piece (and new renditions are performed every day).

This task is generally called *audio matching* (or, mostly in the context of popular music, *cover version identification*, see e.g. [14]). A common approach to solve this problem is to use an *audio alignment* algorithm. This is computationally expensive, as it basically involves aligning the query snippet with every position within every audio file in the database (see [12], and [11] for a indexing method that makes the problem more tractable). Furthermore, due to the coarse feature resolution of these algorithms, relatively large query sizes are needed.

As there exist efficient fingerprinting algorithms, it seems natural to try to adapt them to the problem of cover version identification. A first study towards this is presented in [9], where the authors focused on the suitability of different low-level features as a basis for fingerprinting algorithms, but neglected the problem of tempo differences between performances. In [1] an extension to a well-known fingerprinting algorithm [17] is proposed that makes it invariant to the global tempo. With the help of an *audio transcription* algorithm for piano music (see [5]) a system was built that, given a short audio query, almost instantly returns the corresponding (symbolic) *score* from a reference database – despite the fact that audio transcription is a very hard problem and thus introduces a lot of noise in the process.

In this paper we show how to use this algorithm in the absence of symbolic scores to identify unknown performances, using a reference database based on other *performances* of the pieces in question. As symbolic scores are often not readily available, this increases the applicability of this algorithm in real life systems. The downside of this approach is that now audio transcription is used for both the data contained in the reference database and for the queries, which introduces even more noise. Furthermore, the transcription algorithm we are using is optimised on piano sounds, which for now limits the proposed system to piano music only.

We are going to describe this approach in the context of a system geared towards fully automatic identification of classical piano music, in the sense that even the creation of the collection of audio recordings, which is needed to perform the identification task, is automated. The motivation for this is to reduce the amount of costly manual annotation to a minimum, and instead facilitate available, albeit noisy, web sources like *YouTube* [1] or *Soundcloud* [2]. The main challenge in this setting is the noise introduced into the identification process via multiple processes (automatic retrieval of reference performances, audio transcription of reference performances, and audio transcription of the query). In the paper we will show how to deal with this amount of noise by increasing redundancy in the reference database and by an automatic selection strategy for the reference performances.

The paper is structured as follows. Section 2 gives an overview of the proposed system. Then, in Section 3 the data we are using for our experiments is described. Sections 4, 5, 6 and 7 describe the core experiments of the paper, showing that our approach is robust enough to cope with the multiple sources of noise and performs well in our experiments. A brief outlook on possible improvements and applications is given in Section 8.

2. SYSTEM OVERVIEW

In this section we are going to describe the piece identification system that will be used throughout the paper. The main goals of the system are 1) to automate the process of compiling a reference database, thus making manual annotations obsolete, and 2) based on this reference database, allow for robust and fast piece identification. Figure 1 depicts how the components interact with each other.

The system is based on a **Database Definition** file, which is a list of pieces that are to be included in the database. On this list each piece is represented by an ID, the name of the composer and the name of the piece, including identifiers like the opus number (see Figure 2 for an excerpt of the list). We would like to emphasise once more that this is the only input our system needs (in addition to a source from which the recordings can be retrieved). All the data necessary to perform the identification task is then prepared automatically. This also means that extending the database is as easy as adding a new line to the text file, describing the new piece. The data in this file also defines the granularity of the database. For example, movements of a sonata could be represented as individual pieces or combined as single piece – for our experiments we took the latter approach. For our proof-of-concept implementation we settled for 339 piano pieces of well-known composers (Mozart, Beethoven, Chopin, Scriabin, and Debussy), which already represents a substantial share of the classical piano music repertoire.

A **Web Crawler** takes this list of pieces and retrieves audio recordings of performances of the pieces. In our case we use a simple crawler for *YouTube* (an alternative would be to use *Soundcloud*, amongst others). The queries are constructed by concatenating the name of the composer and the piece, and adding the word "piano", to ensure that mainly piano performances are returned.

Next, the collected recordings are fed into a **Music Transcription Algorithm** that takes the audio files and transcribes them into series of symbolic events. For this step we rely on a well known neural network based method presented in [5], more specifically the version that is available as part of the *Madmom* library [4]. As input it takes a series of preprocessed and filtered STFT frames with two different window lengths. The neural network consists of a linear input layer with 324 units, three bidirectional fully connected recurrent hidden layers with 88 units, and a regression output layer with 88 units, which directly represent the MIDI pitches. The output of the transcription algorithm is a list of detected musical events, represented by their pitches and start times. For details we refer the reader to [5]. This algorithm exhibits state of the art results for the task of piano transcription, as was demonstrated at the MIREX 2014 [3]. Still, polyphonic music transcription is a very hard problem, and thus the output of this transcription algorithm contains a relatively large amount of noise, of which the following components need to be robust to.

The **Automatic Preprocessing** step is concerned with the question of which of the downloaded recordings for each piece should be used in our fingerprint database. In this paper we discuss three setups: take the top match returned by the web crawler (see Section 4), take the top five / fifteen matches returned by the web crawler (see Section 5), and download 30 recordings for each piece, rank them automatically via comparing them to each other and use the top recordings identified via this approach (see Section 6). This means that in the latter two experiments a single piece is represented by *multiple* recordings, adding redundancy to the reference database.

The transcribed sequences of symbolic event information, i.e. sequences of pairs (pitch, onset time), are fed to the **Tempo-invariant Symbolic Fingerprinter**, to build a database of fingerprints that later on can be used to identify queries. The algorithm is used as described in [1], thus it will be summarised here very briefly. The principle idea of the fingerprinting algorithm is to represent an instance (in this case a transcribed performance, representing a piece) via a large number of local, tempo-invariant fingerprint tokens. These tokens are created based on the pitches of three temporally local note events, together with the ratio of their distances in time. Due to the way they are created, the tokens are invariant to the global tempo, and can be stored in a hash table and efficiently queried for.

An incoming **Query** is processed in the same way as above by the **Music Transcription Algorithm**. The resulting sequence of symbolic events is used to query the **Tempo-invariant Symbolic Fingerprinter** for matches. To do so, from the query the same kind of fingerprint tokens are computed, and matching tokens are retrieved from the fingerprint database. Finally, in this result set continuous sequences of matching tokens, which are a strong in-

[1] `https://www.youtube.com`

[2] `https://soundcloud.com`

[3] `http://www.music-ir.org/mirex/wiki/2014:MIREX2014_Results`

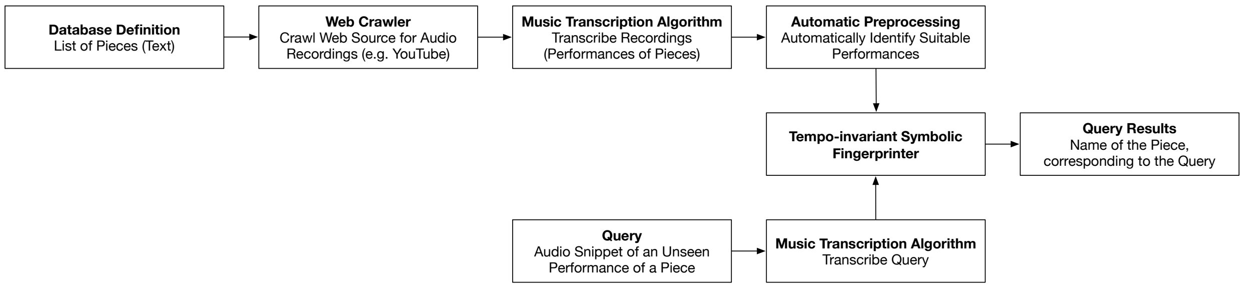

Database Definition
List of Pieces (Text)
Web Crawler
Crawl Web Source for Audio Recordings (e.g. YouTube)
Music Transcription Algorithm
Transcribe Recordings (Performances of Pieces)
Automatic Preprocessing
Automatically Identify Suitable Performances
Tempo-invariant Symbolic Fingerprinter
Query Results
Name of the Piece, corresponding to the Query
Query
Audio Snippet of an Unseen Performance of a Piece
Music Transcription Algorithm
Transcribe Query

Figure 1. System Overview

```
ID;Composer;Piece
...
17;Mozart;Piano Sonata No. 17 in B-flat major K 570
18;Mozart;Piano Sonata No. 18 in D major K 576
19;Mozart;Fantasy No. 1 with Fugue in C major K 394
20;Mozart;Fantasy No. 2 in C minor, K 396
...
41;Beethoven;Piano Sonata No. 14, Op. 27, No. 2 "Moonlight"
42;Beethoven;Piano Sonata No. 15, Op. 28 "Pastoral"
...
168;Chopin;Mazurka Op. 7 No. 5 in C major
169;Chopin;Nocturne Op. 15 No. 1 in F major
170;Chopin;Nocturne Op. 15 No. 2 in F-sharp major
171;Chopin;Nocturne Op. 15 No. 3 in G minor
...
281;Debussy;L 113, Children's Corner, Doctor Gradus ad Parnassum
282;Debussy;L 113, Children's Corner, Jimbo's Lullaby
...
332;Scriabin;Piano Sonata No. 3, Op. 23
333;Scriabin;Piano Sonata No. 4, Op. 30
...
```

Figure 2. An excerpt of the file used for collecting the database.

dication that the query matches a specific part of a piece stored in the fingerprint database, are identified (via a fast, histogram based approach).

The **Query Result** is a list of positions within the reference performances that were inserted into the database (see Table 1). The positions in the result set are ordered by their number of tokens matching the query. As can be seen, the result set is actually more detailed than necessary for our applications scenario, as we are only interested in identifying the respective piece, and not a specific reference performance (or even a position within reference performance). Thus for the experiments in this paper we summarise all occurrences of a piece into one score by summing up the matching scores of all its occurrences in the results set.

3. GROUNDTRUTH DATA AND EXPERIMENTAL SETUP

For the experiments presented in this paper, ground truth data, i.e. performances for which the composer and the name of the piece is known, is needed. We are using commercial recordings of a large part of the pieces contained in our database. This includes e.g. Uchida's recordings of the Mozart Sonatas, Brendel's recordings of the Beethoven Sonatas, Chopin recordings by Arrau, Pires and Pollini, and Debussy recordings by Pollini, Thibaudet, Zimerman. We would like to emphasise that to get realistic results, in our experiments we made sure manually that no exact replicas of these performances are contained in the automatically downloaded data that is used to build the reference database later on. In total 370 tracks were selected and assigned manually to the respective pieces (roughly 30 hours of music, or 665 000 transcribed events). Some of the tracks were assigned to the same piece, as e.g. the movements of the sonatas are typically represented as different audio tracks, but are represented as a single piece in our database.

Piece ID	Performance ID	Time in Ref.	Score
1	0	99	351
1	0	21	292
1	4	16	109
1	4	15	36
1	4	148	36
1	4	150	32
10	48	368	7
1	0	239	7

Table 1. An example of a result returned by the fingerprinting algorithm. This query was performed on a database in which multiple reference performances represent a piece of music, hence for the piece with ID 1 results for two performances are returned. The score is the number of matching fingerprint tokens for the given query at the specific time in the reference recording. For our purposes we summarise the results per piece, i.e. the matching score for the piece with ID 1 is 863, and for the piece with ID 10 it is 7.

The experimental setup is as follows. We are going to use the same set of randomly extracted queries for each experiment. We are using three query lengths of 2, 5 and 10 seconds (we only took queries though which had at least 10 transcribed notes, avoiding to e.g. query for silence), and extract for each length ten queries for each ground truth performance (giving a total of 3 700 queries for each query length). The experiments are based on different strategies to automatically compile the reference database. We start with a simple baseline approach (Section 4) and then gradually improve on it by introducing redundancy and a selection strategy (Sections 5 to 7).

As evaluation measure we use the *Recall at Rank k*[4].

[4] We would like to note that the related measure *Precision at Rank k* is not useful in our experimental setup, as there will only be at most one correct result in the result set.

	Query Length		
	2 s	5 s	10 s
Recall at Rank 1	0.28	0.38	0.46
Recall at Rank 5	0.34	0.45	0.54
Recall at Rank 10	0.35	0.47	0.55
Mean Reciprocal Rank	0.30	0.41	0.48
Mean Query Time	0.13 s	0.41 s	0.92 s

Table 2. Results of the baseline approach. The results are based on 3 700 queries for each query length.

This is the percentage of queries which have the correct corresponding piece in the first k retrieval results. In our experiments we look at the recall at ranks 1, 5 and 10. In addition, we also report the *Mean Reciprocal Rank* (MRR).

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} \quad (1)$$

Here, rank_i refers to the rank position of the correct result for the i^{th} query.

The mean query times (i.e. the mean time it takes to process a single query) given in the tables are based on a desktop computer on a single core [5] . If needed, the computation could easily be sped up by multi-threading the query process.

4. BASELINE APPROACH

The baseline approach is very straightforward. The web crawler is used to download the top result from the web source for each piece on the list. The downloaded audio files are transcribed and then processed by the fingerprinting algorithm to build the reference database, i.e. in the reference database each piece is represented by one performance. Note that due to the automatic process the database can be quite noisy, as some of the pieces might be incomplete (e.g. only a single movement of a piece), represented by more than the actual piece (if e.g. the performance downloaded for the piece also contains other pieces, like a recording of a full concert), or the representation is wrong (if the top result of the web crawler is actually a performance of some other piece).

The generated fingerprint database is queried via the prepared excerpts of the collected ground truth data (see Section 3). The results of this first experiment can be seen in Table 2. As can be seen, already in this scenario and despite the small query sizes the method gives reasonable results. For queries of length ten seconds the algorithm returns the correct name of the piece in close to 50% of the cases. A closer look at the results though showed that the main problem with this simplistic approach is that, as expected, for many pieces the representation in the database is not correct or incomplete. This problem is tackled in the following sections.

[5] Intel Core i7 6700K 4 GHz with 32 GB RAM.

	Query Length		
	2 s	5 s	10 s
Recall at Rank 1	0.58	0.69	0.74
Recall at Rank 5	0.72	0.84	0.90
Recall at Rank 10	0.74	0.86	0.92
Mean Reciprocal Rank	0.64	0.77	0.84
Mean Query Time	0.34 s	0.81 s	2.49 s

Table 3. Results on the reference database based on multiple recordings (the top five results according to the web source) to represent each piece. The results are based on 3 700 queries for each query length.

	Query Length		
	2 s	5 s	10 s
Recall at Rank 1	0.76	0.87	0.91
Recall at Rank 5	0.84	0.94	0.97
Recall at Rank 10	0.86	0.95	0.98
Mean Reciprocal Rank	0.80	0.90	0.94
Mean Query Time	0.82 s	2.85 s	6.08 s

Table 4. Results on the reference database based on multiple recordings (the top fifteen results according to the web source) to represent each piece. The results are based on 3 700 queries for each query length.

5. USING MULTIPLE INSTANCES PER PIECE

A simple way to improve the performance of the system is to increase the redundancy within the reference database. Instead of relying on a single instance (recording) for each piece in the reference base, each piece is represented by multiple recordings. For the first experiment five performances per piece were downloaded using the web crawler. The performances were processed in the same way as for the baseline approach in Section 4 above and inserted into the fingerprint database. Then, on this database the same set of queries were performed. As described in Section 2, the match score of a piece is computed by summing up the scores of the performances representing the piece in question (also see Table 1).

Table 3 shows the results of this experiment. As can be seen, the increased redundancy leads to a substantial increase in identification results, compared to the baseline (see Table 2). The added redundancy increases the chances that for each piece at least one "good" performance (in the sense of corresponding to the piece and relatively easy to transcribe) is contained in the reference database, and thus mitigates the problems caused by noise, at least to some extent.

For an additional experiment we increased the number of performances to fifteen per piece. These results are shown in Table 4. This improved the results even further. The downside of adding more instances to the fingerprint database is a significant increase in computation time.

	Query Length		
	2 s	5 s	10 s
Recall at Rank 1	0.54	0.68	0.74
Recall at Rank 5	0.63	0.76	0.83
Recall at Rank 10	0.64	0.78	0.85
Mean Reciprocal Rank	0.58	0.72	0.78
Mean Query Runtime	0.14 s	0.47 s	0.97 s

Table 5. Results on the reference database based on the top recording selected via the proposed strategy to represent each piece. The results are based on 3 700 queries for each query length.

	Query Length		
	2 s	5 s	10 s
Recall at Rank 1	0.72	0.85	0.89
Recall at Rank 5	0.82	0.92	0.96
Recall at Rank 10	0.84	0.93	0.97
Mean Reciprocal Rank	0.77	0.88	0.92
Mean Query Time	0.49 s	1.71 s	3.83 s

Table 6. Results on the reference database based on multiple recordings (top five recordings selected via the proposed strategy) to represent each piece. The results are based on 3 700 queries for each query length.

6. AUTOMATICALLY SELECTING SUITABLE REPRESENTATIONS

A closer look at the results so far shows that increasing the redundancy in the reference database indeed leads to better results, but also increases the computation time. The main problem with our approach is that in addition to useful data, the process also adds a lot of extra noise to the fingerprint database. The web crawler returns a considerable number of performances of the wrong piece, performances played on a different instrument, and performances recorded in very bad quality. This kind of data increases the runtime and decreases the identification accuracy. In this section we present a method for identifying performances in a given a set of candidates for a piece that most probably are related to the piece in question, which also enables us to discard performances that most probably are noise. In this way we try to reduce the number of stored fingerprint tokens, which generally decreases the computation time, while still achieving good identification performance.

Thus, for each piece we perform the following process to select appropriate representations. First, 30 recordings are downloaded via the web crawler. With a high probability at least some of these are actually piano performances of the piece we are looking for, while the others might have nothing in common. The idea now is to find a homogenous group within this set of candidates. To identify performances which are part of this group, we again employ the symbolic fingerprinting process, but limited to the set of candidate performances. To do so, the performances are transcribed and inserted into a new fingerprint database.

The intuition is that for a query extracted from the same set of candidate performances (that actually matches the piece), the fingerprinter will likely return three kinds of results. Firstly, the top result will be the performance the query was taken from. This is a perfect fit for all tokens, which results in the maximum score. Secondly, a number of other performances will probably also have a high score, identifying them as being based on the same piece and as being transcribed in sufficient quality. Thirdly, performances that actually belong to a different piece, or which are transcribed poorly, will score very low.

Based on these observations, we designed the process of ranking the performances regarding their suitability to represent the piece in question as follows. For each of the performances ten queries are randomly extracted (for our experiments we used a query length of ten seconds) and processed by the fingerprinting algorithm. As in all other experiments, the results are summarised on the performance level (i.e. match scores of positions within the same performance are summed up). Then, for each result the score of the top match (i.e. of the performance the query stems from) is stored, this performance is removed from the result set, and the remaining matching scores are normalised by dividing by the top match score. The reasoning behind this is that the absolute scores depend on the particulars of the query (foremost the length in the sense of the number of notes, but also e.g. if the part in question is normally played in a steady tempo or is subject to expressive tempo changes, which makes it harder to detect and leads to a lower score).

This results in 300 preprocessed and normalised result sets. The suitability of a performance to represent the piece in question is computed by summing up all the scores of all its occurrences in the result sets. The higher this value is for a performance, the more it has in common with the other performances assigned to the piece in question.

Based on this ranking we repeat experiments from Sections 4 and 5, but this time for each piece we select the top one or top five performances, respectively, according to the computed rank within the candidate set for each piece. The results are shown in Tables 5 and 6, which should be compared to Tables 2 and 3, respectively. As can be seen the selection strategy increases the identification performance for both scenarios and for all query lengths.

A comparison of Tables 6 and 4 shows that by using the proposed selection strategy a lower number of performances (5 versus 15) is sufficient to achieve comparable identification accuracy. The decreased number of tokens also results in roughly half the computation time.

The runtime actually depends on a number of factors, most importantly the size of the fingerprint database. But of similar influence is the actual number of tokens that are returned by the fingerprint database for a specific query. The reason is that each of these tokens has to be processed individually to come up with the matching score. This also means that queries for pieces which are represented in the database by a large number of performances will actually take *longer* to compute – a further argument in favour of the selection strategy presented in this section.

7. USING MULTIPLE QUERIES PER PERFORMANCE

So far the assumption was that we only have access to a single short query of two to ten seconds. If instead we have access to a full recording, just querying for one short query would be a suboptimal approach. Thus, we tried an additional query strategy on the reference database based on the performance selection strategy from Section 6 above.

A standard approach for processing long queries (in this case a whole performance) would be to apply shingling [2,7,8], i.e. splitting longer queries into shorter, overlapping ones and track the results of these sub-queries over time. Here, as proof of concept we use an even simpler method: we select ten random queries from the piece we want to identify, process them individually and sum up the results. This can be seen as adding redundancy (relying on multiple queries instead of a single one) on the query side. We perform this experiment on the reference database based on the top five selected recordings via the proposed strategy. The results are shown in Table 7. As can be seen this again considerably improves the results, and we are getting very close to 100%. The main cause for this is that the retrieval precision heavily depends on the quality of the transcription. Some parts of a performance are much harder to transcribe than others (e.g. heavily polyphonic parts with a lot of sustain pedal, which are difficult to transcribe correctly). Using multiple queries, randomly distributed over the whole performance, increases the chances that at least some parts are transcribed in good quality, and that together these queries enable high retrieval accuracy.

Finally, we had a closer look at the few performances that were still misclassified and identified two problems. Our approach does not take care of the problem of recordings of full concerts. If included in the reference database for multiple pieces, these will lead to misclassifications. Furthermore, for some pieces only a small number of performances exists, which causes the crawler to return "similar" but wrong performances (e.g. performances of other pieces of the same composer). We sketch a possible solution to these problems in Section 8 below.

	Querylength		
	2 s	5 s	10 s
Recall at Rank 1	0.92	0.95	0.95
Recall at Rank 5	0.98	0.99	0.99
Recall at Rank 10	0.99	1	1
Mean Reciprocal Rank	0.94	0.97	0.97
Mean Query Time	0.49 s	1.71 s	3.83 s

Table 7. Results for querying for a whole performance via ten random small queries with ten seconds each. The results are based on 3 700 queries for each query length.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach towards piece identification for performances of piano music, based on an automatically compiled reference database using web sources. It is shown that the symbolic fingerprinting method is robust enough to deal with the noise introduced by the transcription algorithms and allows for fast querying in the symbolic domain. Furthermore, increasing the redundancy by using multiple performances to represent a single piece, especially using the proposed selection strategy, largely alleviates the problem of noise introduced by the automatic compilation of the reference database. Additionally, this increases the robustness of the identification process via the fingerprinting algorithm, as 'problematic' sections (e.g. regarding the transcription process) are represented multiple times, thus increasing the chances that the parts in question are well covered by the reference database.

There exist a number of possible improvements regarding the automatic selection of performances for a piece. In our implementation the focus is on increasing the homogeneity within the group of performances for a piece by comparing them to each other. An additional option is to analyse matches on the full reference database and try to find out which performances match well to multiple pieces and exclude them (as they cover multiple songs or were mistakenly assigned to multiple pieces by the crawler).

We are currently in the process of collecting a much larger collection of classical piano music. This dataset will contain a few thousand pieces, covering a large part of the classical piano repertoire[6]. On this dataset we are going to conduct experiments regarding the scalability of our approach in terms of runtime and retrieval accuracy.

In the future, we will also investigate the usefulness of the presented approach for non-classical piano music. Preliminary experiments have shown that this is a much harder task, as compared to classical piano music the pieces are not as strictly defined via a detailed score (e.g. popular songs and jazz standards are mostly described via lead sheets). Thus, performances of the same piece differ more heavily than in classical music. Of course we would also like to lift the restriction to piano music and try our method on other genres, but thus far general music transcription is not robust enough to be used with our approach. Hopefully this will change in the future.

Finally, regarding real-world applications, an automatic method to determine which pieces are well covered by the database, and which ones would benefit from manual intervention, would be desirable. This would help to quickly build a reference database which already covers most pieces well, and then to manually add additional references (based on performances, or even on symbolic score data) for pieces the identification algorithm struggles with.

[6] The reference database is of course compiled automatically (based on the list of pieces), but the preparation of the ground truth for the experiments is a time consuming, manual process.

9. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC Grant Agreement 670035, project CON ESPRESSIONE).

10. REFERENCES

[1] Andreas Arzt, Sebastian Böck, and Gerhard Widmer. Fast identification of piece and score position via symbolic fingerprinting. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 433–438, Porto, Portugal, 2012.

[2] Andreas Arzt, Gerhard Widmer, and Reinhard Sonnleitner. Tempo- and transposition-invariant identification of piece and score position. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 549–554, Taipeh, Taiwan, 2014.

[3] Shumeet Baluja and Michele Covell. Waveprint: Efficient wavelet-based audio fingerprinting. *Pattern Recognition*, 41(11):3467–3480, 2008.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[5] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 121–124, Kyoto, Japan, 2012.

[6] Pedro Cano, Eloi Batlle, Ton Kalker, and Jaap Haitsma. A review of algorithms for audio fingerprinting. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 169–173, St. Thomas, Virgin Islands, USA, 2002.

[7] Michael A. Casey and Malcolm Slaney. Song intersection by approximate nearest neighbor search. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 144–149, Victoria, Canada, 2006.

[8] Peter Grosche and Meinard Müller. Toward characteristic audio shingles for efficient cross-version music retrieval. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Kyoto, Japan, 2012.

[9] Peter Grosche and Meinard Müller. Toward musically-motivated audio fingerprints. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 93–96, Kyoto, Japan, 2012.

[10] Peter Grosche, Meinard Müller, and Joan Serrà. Audio content-based music retrieval. In Meinard Müller, Masataka Goto, and Markus Schedl, editors, *Multimodal Music Processing*, volume 3 of *Dagstuhl Follow-Ups*, pages 157–174. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2012.

[11] Frank Kurth and Meinard Müller. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):382–395, 2008.

[12] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 288–295, London, UK, 2005.

[13] Mathieu Ramona and Geoffroy Peeters. Audioprint: an efficient audio fingerprint system based on a novel cost-less synchronization scheme. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 818–822, Vancouver, Canada, 2013.

[14] Joan Serrà, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: background, approaches, evaluation and beyond. In Z. W. Ras and A. A. Wieczorkowska, editors, *Advances in Music Information Retrieval*, volume 274 of *Studies in Computational Intelligence*, chapter 14, pages 307–332. Springer, Berlin, Germany, 2010.

[15] Joren Six and Marc Leman. Panako - a scalable acoustic fingerprinting system handling time-scale and pitch modification. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 259–264, Taipei, Taiwan, 2014.

[16] Reinhard Sonnleitner and Gerhard Widmer. Robust quad-based audio fingerprinting. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 24(3):409–421, 2016.

[17] Avery Wang. An industrial strength audio search algorithm. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 7–13, Baltimore, Maryland, USA, 2003.

PIPO, A PLUGIN INTERFACE FOR AFFERENT DATA STREAM PROCESSING MODULES

Norbert Schnell
UMR STMS
IRCAM-CNRS-UPMC
schnell@ircam.fr

Diemo Schwarz
UMR STMS
IRCAM-CNRS-UPMC
schwarz@ircam.fr

Joseph Larralde
UMR STMS
IRCAM-CNRS-UPMC
larralde@ircam.fr

Riccardo Borghesi
UMR STMS
IRCAM-CNRS-UPMC
borghesi@ircam.fr

ABSTRACT

We present *PiPo*, a plugin API for data stream processing with applications in interactive audio processing and music information retrieval as well as potentially other domains of signal processing. The development of the API has been motivated by our recurrent need to use a set of signal processing modules that extract low-level descriptors from audio and motion data streams in the context of different authoring environments and end-user applications.

The API is designed to facilitate both, the development of modules and the integration of modules or module graphs into applications. It formalizes the processing of streams of multidimensional data frames which may represent regularly sampled signals as well as time-tagged events or numeric annotations. As we found it sufficient for the processing of incoming (i.e. *afferent*) data streams, *PiPo* modules have a single input and output and can be connected to sequential and parallel processing paths. After laying out the context and motivations, we present the concept and implementation of the *PiPo* API with a set of modules that allow for extracting low-level descriptors from audio streams. In addition, we describe the integration of the API into host environments such as Max, Juce, and OpenFrameworks.

1. INTRODUCTION

1.1 Context and Motivation

Many of the interactive audio applications that we have developed over the past years in collaboration with artists and other researchers rely on signal processing techniques to automatically analyse and annotate audio and motion sensor streams. We often refer to the techniques we deploy in this context as *content-based audio processing* [1]. These techniques generally allows for interactively transforming recorded audio materials as a function of low-level audio descriptions such as pitch, intensity, and timbre descriptions as well as segmentations into temporal units such as notes, syllables, and musical phrases. Similar processing applies in this context to motion capture streams to extract movement qualities and meaningful events and temporal units such as onsets and gestures. In this processing, we frequently reuse a set of algorithms such as filters, projections, extractors, and detectors that may apply in real-time to incoming data streams or, offline, to data streams recorded into files. Since these data streams occur in the overall interactive systems we develop as inputs, we refer to them as *afferent* data streams.

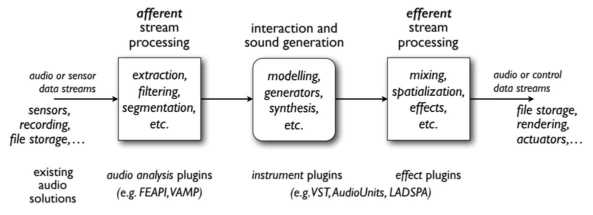

afferent stream processing
interaction and sound generation
efferent stream processing
audio or sensor data streams
sensors, recording, file storage,...
extraction, filtering, segmentation, etc.
modelling, generators, synthesis, etc.
mixing, spatialization, effects, etc.
audio or control data streams
file storage, rendering, actuators,...
existing audio solutions
audio analysis plugins (*e.g. FEAPI,VAMP*)
instrument plugins
effect plugins
(*e.g.VST, AudioUnits, LADSPA*)

Figure 1. An interactive audio system with *afferent* and *efferent* data streams. The labels at the bottom cite existing plugin interfaces for audio applications.

Figure 1 shows the overall structure of such an interactive system. The schema does not distinguish whether the audio and motion data streams actually enter the system in real-time or whether they are read from files. Afferent streams processed in real-time typically originate from a microphone or motion sensors. In some of the systems we have developed, the streams are used to control an interactive system through sound (e.g. voice) or movement. In many of these systems, the same — or very similar — processing that is applied in real-time, applies to data streams read from files. For example for concatenative synthesis [21], audio descriptors are extracted from prerecorded materials. While some interactive systems may generate sound in real-time, the generated description and annotations may be used to create visualizations (e.g. in the context of musicology or education) as well as to support the editing and transformation of recordings in post-production systems.

In general, the processing of afferent streams can be described as *reducing* the data streams in terms of their complexity, dimensionality, and data rate. Typical terms used to characterize this processing include *filtering*, *anal-*

ysis, *extraction*, *description*, *detection*, *recognition*, *scaling*, and *mapping*.

The *PiPo* API (*Plug-in Interface for Processing Objects*) formalizes modules that in this sense transform an incoming data stream into an output data stream allowing for a possibly wide range of streams as well as for modules of arbitrary complexity going from simple scalings to sophisticated machine learning algorithms.

The major motivations for developing the *PiPo* API can be summarized as follows:

- facilitating the integration of algorithms of different origins (i.e. developers) into a given application
- facilitating the use/comparison of different algorithms of similar functionalities in a given context (e.g. applying different filters, extractors or classifiers to the same input stream)
- facilitating the integration of a given algorithm into different contexts and applications
- facilitating the development of applications where the same algorithm applies to data streams in real-time and offline

Ultimately, the motivation for developing the API is the idea of enabling the development of an ecosystem of stream modules and host environments in particular domains as well as across different domains of signal processing.

1.2 Requirements

In this section, we give an overview over the most important general requirements for an afferent data stream processing framework for real-time and off-line use. These requirements concern specific functionalities as well as their efficient implementation in a real-time system (see [22]).

1.2.1 Functional Requirements

Scheduling Processing should run either in batch on sound files and buffers, or on a live audio stream

Segmentation Allow several streams of segmentations in parallel and overlapping segments, or an implicit segmentation, where segments are analysis frames, elementary waveforms, or whole sound files.

Temporal Modeling Any number of temporal modeling algorithms can be integrated, either universal (modeling all descriptors, e.g. mean) or specific (modeling specific descriptors only, e.g. geometric mean for pitch).

Data Type Data can be numeric scalars, vectors, matrices, or strings

Multi-Modality The input data type and rate should allow motion and other data and not be limited to audio only.

User Composability Modules should be composable by the user in the host environment (without having to write and compile code), e.g. chaining feature extractors, smoothing filters, segmentation, and temporal modeling, in order to allow experimentation and rapid prototyping.

1.2.2 Implementation Requirements

Easy Integration and Efficiency It should be easy to integrate the framework in any platform and environment, including real-time and resource-constrained systems (e.g. single-board computers). This basically stipulates that the API be written in C or C++.

Dynamic Plugin Loading It should be possible to add processing modules as plugins to an existing host installation, e.g. by leveraging dynamic linking of shared libraries.

Efficient Modularisation The framework should allow an efficient implementation, notably by *sharing commonly used calculation results*, most of all the FFT representation, between modules, by *avoiding copying* and re-sending data, instead writing them directly to its destination.

External Data External data streams and sources of segmentation, such as a human tapping on attacks oder existing analysis files, must be integratable into the data flow.

Reanalysis A subset of descriptors or only the segmentation and subsequent temporal modeling can be re-run with changed parameters.

Almost all of these requirements are fulfilled by *PiPo*, with the exception of the possibility to pass strings as data elements. This has been avoided to simplify the API and avoid problems of memory-handling. A fixed set of strings (such as class labels for machine-learning) can always be transmitted by their index.

The top-level requirements, that best distinguish *PiPo* from other frameworks are dynamic linking of plugins, multi-modality, and user-composability of modules.

1.3 Related Work

In the rich existing work, we must distinguish audio analysis libraries and toolboxes (see the recent overview [15]) from plugin APIs which impliy a formalization of the input/output formats and the dynamic loading of modules.

Several plugin APIs are commonly used in the world of audio signal processing and virtual instruments, namely LADSPA (Linux Audio Developer's Simple Plugin API),[1] VST (Virtual Studio Technology by Steinberg),[2] and AU (Audio Units by Apple).[3] These APIs are mainly designed for transforming an input audio stream (effect processing) or for generating an audio stream in reaction to incoming MIDI events (virtual instruments). Thus they are not applicable to the demands of general data processing or audio feature extraction.

Many monolithic or collections of analysis modules for popular real-time environments exist, such as

[1] http://www.ladpsa.org

[2] http://ygrabit.steinberg.de

[3] http://developer.apple.com/audio/audiounits.html

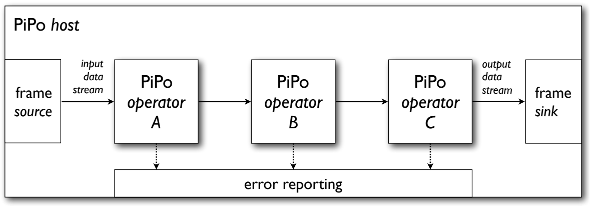


Figure 2. A chain of modules in a host environment.

`analyser~` [11], the patch-based ZSA [13] for MAX, imtr-analysis [22], *TimbreID* [3] for PureData. None of these can be integrated into other environments.

The first descriptor analysis frameworks that would allow the dynamic inclusion of external modules are either plugin frameworks such as the sadly defunct FEAPI [12], and the more lively VAMP [6]. [4] However, the latter does not propose user-composability nor multi-modality (the input is always audio).

There are many existing libraries for audio descriptor analysis (*Yaafe* [5] [14], *Essentia* [6] [2], *OpenSmile* [7] [7, 8], *libXtract*, [8] *IrcamDescriptor* [18]) see this comparison [15]. None of them allow for dynamic linking, easy integration of new algorithms, or user composability without having to code a new module.

The MARSYAS framework, dedicated to music information retrieval, is concerned with scheduling [5] as well as CLAM, [9] but neither is a common environment for real-time sound and music applications.

In summary, no existing API combines all three top-level requirements of dynamic linking of plugins, multi-modality, and user-composability of modules.

2. CONCEPTS AND FORMALIZATION

The *PiPo* API formalizes *modules* as objects that receive a data stream as a succession of *frames* at their input and send a stream as output. As shown in figure 2, modules can be connected to a chain by connecting the input of one module to the output of another. In the simplest case, the processing requires a single module. A *PiPo host*, constructs the modules and connects to the input of the first module of the chain as the source of the stream of frames to be processed. In addition, the host connects to the output of the last module of the chain as the terminating sink that receives the resulting stream.

The data streams received and produced by *PiPo* modules are described by a set of *stream attributes* that are defined before the modules actually receive and produce any frames. This way, the initialization of a module may depend on the attributes of the incoming stream and the module may determine the attributes of the stream it produces as a function of the attributes of the incoming stream.

[4] http://vamp-plugins.org
[5] http://yaafe.sourceforge.net
[6] http://essentia.upf.edu/
[7] http://opensmile.sourceforge.net/
[8] http://jamiebullock.github.io/LibXtract/documentation/
[9] http://clam-project.org

The propagation of the stream parameters and the actual processing of frames are separated into two phases that are both initiated by the host through its connection to the first module. In both phases, each module receives information from its predecessor in the chain and sends information to its successor. In the initialization phase, the host sends out the stream parameters of the input stream to the first module which sends its output stream parameters to the input of the next module, and so forth, until the last module sends the resulting stream parameters back to the host connected to its outlet. Similarly, once the modules are initialized, the host can start sending frames into the input of the first module and receives the resulting frames from the output of the last module. Only in the case of error, as for example when a module cannot accept a stream with a given set of attribute values at its input, a module would report directly to the host, which in turn can output the error message to the host environment.

2.1 Streams of Frames

Each frame of a data stream is composed of a *time-tag* and a two-dimensional matrix of numeric values. A data stream is described by the following set of attributes:

- *frame rate* of the stream
- whether the frames of the stream are *time-tagged*
- *dimensions* of the frames' two-dimensional matrix
- *labels* describing the columns of the data matrix
- whether the frames' data matrices have a *variable number of rows*

In case of streams of time-tagged frames of an irregular rate, the *frame rate* attribute should announce the worst case (the fastest) rate, so that succeeding modules — or the host — can take this parameter into account (e.g. for allocating memory).

This formalization of data streams allows for representing a large spectrum of different signals, event streams, and numeric annotations. For example:

- *mono or multi-channel audio signals* are represented as scalars or multi-dimensional vectors of a constant frame rate
- *real or complex frames of spectral data* are represented as single column vectors or matrices of two columns (i.e. labeled *'real'* and *'imag'*), usually of a constant frame rate
- *multi-dimensional motion capture data streams* are represented as multi-dimensional row-vectors (e.g. labeled *'x'*, *'y'*, *'z'*) that may be time-tagged or of a constant frame rate
- *onset markers* are represented as time-tagged frames without numeric data (i.e. an empty matrix)
- *segments* are represented as time-tagged frames with a row-vector of data including a *'duration'* column and, optionally, multiple columns of values describing the segment (e.g. *'pitch'*, *'intensity'*, *'category'*)

- *harmonics* are represented as two-dimensional matrices with variable number of rows, one row for each harmonic, with multiple columns (*'frequency'*, *'amplitude'*, *'phase'*) of a constant frame rate

From the host's point of view, once constructed, an arbitrary chain of modules is defined by the stream it produces as a function of the input stream provided by the host. Before starting the actual stream processing by sending frames into the chain, the host retrieves the attributes of the output stream that can be used, for example, to allocate memory or bandwidth and automatically determine display options as well as to configure and generate other interactions with connected sub-systems or users.

2.2 Chains of Modules

As mentioned above, *PiPo* modules have a single input and output and can be connected to chains. Hereby, a chain of modules — conceptually as well as by implementation — may appear as a single module communicating with its environment (i.e. a host or connected modules) through a single in- and output and an error channel.

Apart from the *stream attributes* of its incoming stream, each module is configured and controlled by a set of typed *module parameters* that are explicitly declared through the *PiPo* API. Possible parameter types are single values of 64-bit float, 32-bit integer, string, and declared enumerated types as well as fixed or variable sized arrays of values of these types and heterogeneous variable sized arrays. In addition to its type, a module parameter is declared with a name, a short description, and a flag whether changing a given module parameter requires the reinitialization of the module — and consequently of the following modules in a chain.

An important feature of the design of the API is that it allows for implementing modules of virtually any complexity and for composing chains of modules of any granularity. An extractor of MEL cepstrum coefficients (MFCCs), for example, may be implemented as a single monolithic module or composed of a chain of modules that include the successive calculation of STFT frames, MEL coefficients, and DFT coefficients.

2.3 Graphs Beyond Chains

The construction of certain algorithms from basic modules requires more complex graphs of modules. For example, the extraction of a set of basic audio descriptors shown in figure 3 requires to split and merge the processing of the implied data streams. While the first split allows for processing the same audio frames in time and frequency domain, the second applies the calculation of a loudness descriptor and a spectral centroid to the same frequency domain frames produced by the STFT. The final set of estimated descriptor values (i.e. pitch, periodicity, AC1, loudness, and spectral moments) is merged to a single vector at the output of the sub-graph.

As described in section 2.2, any chain (or sequence) of modules can be considered as a single module. In the formalization of graphs in the *PiPo* API, parallel modules can also be reduced to a single module. These two rules, illustrated by figure 4, provide a consistent basis to build a large variety of complex *PiPo* graphs.

Figure 5 shows the structure of the `pipo.descr` module expressed in terms of sequence and parallel components.

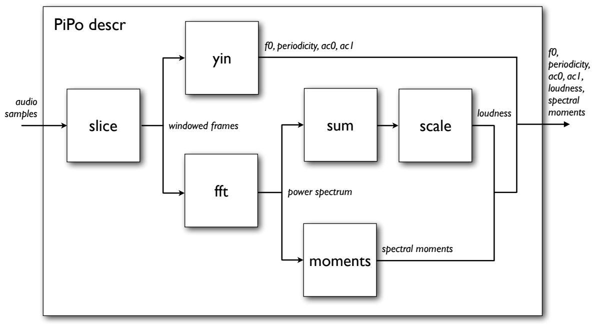


Figure 3. A complex graph of *PiPo* modules for calculating 9 basic audio descriptors.

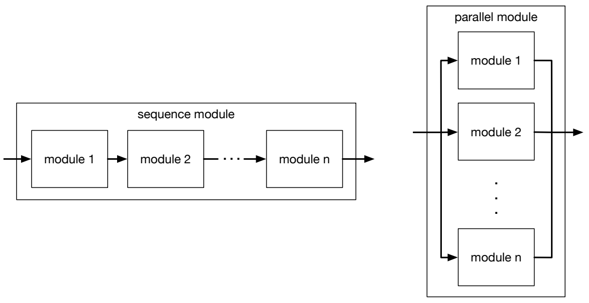


Figure 4. Any number of modules connected in sequence or in parallel can be reduced to a single module.

2.4 Hosts

In summary, a *PiPo* host has to provide the following functionalities:

- constructing a single or a graph of modules
- parametrizing the modules

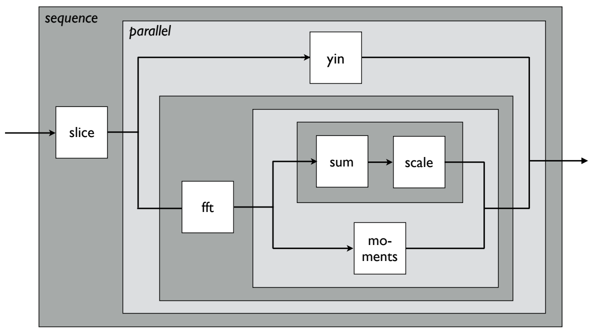


Figure 5. Decomposition of the `pipo.descr` module into sequence and parallel elements.

- connecting a terminating sink to the output of the chain
- acquiring the input stream
- initializing the modules by sending the input stream attributes into the chain
- handling initialization errors emitted by the modules
- sending the frames of the input stream into the chain and handling the frames of the output stream
- allowing for real-time parametrization of the modules (if applicable)

The *PiPo* API includes abstractions that support the implementation of hosts.

3. IMPLEMENTATION

PiPo is an API that essentially consists of a single C++ header file. This file defines the base *PiPo* class, and its declared parameters. [10] .

3.1 The PiPo API

The minimal module must inherit from the class `PiPo` and implement at least the `streamAttributes` and `frames` methods:

In `streamAttributes`, all initialisation can be done, as all input stream attributes are known. The output stream attributes are passed on to the receiving module via `propagateStreamAttributes`. In `frames`, only data processing and, when needed, buffering should be done. Output frames are passed on with `propagateFrames`.

If the module can produce additional output data after the end of the input data (e.g. filters), it must implement `finalize`, from within which more calls to `propagateFrames` can be made, followed by a mandatory call to `propagateFinalize`.

If the module keeps internal state or buffering, it should implement the `reset` method to put itself into a clean state.

A segmentation module calls the method `propagateSegment` to signal the onset, offset and exact boundaries of a new segment to following temporal modeling modules (which implement `segment`).

The utility function `signalError` can be used to pass an error message to the host.

3.2 Module Parameters

The template class `PiPo::Attr` permits to define scalar, enum, or variable or fixed size vector parameters of a pipo module that are exposed to the host environment.

They are initialised in the module constructor with a short name, a description, a flag if a change of value means the fundamental stream parameters must be reset (if true, `streamAttributes` will be called again for the whole chain), and a default value.

Their value can be queried in `streamAttributes` or `frames` (in real-time hosts, a parameter's value can change over time) with `PiPo::Attr::get()`.

[10] https://github.com/Ircam-RnD/pipo-sdk/tree/master/include

3.3 Example of a Minimal PiPo Module

```
class PiPoGain : public PiPo
{
private:
  std::vector<PiPoValue> buffer;

public:
  PiPoScalarAttr<double> factor;

  PiPoGain (Parent *parent, PiPo *receiver = NULL)
  : PiPo(parent, receiver),
    factor(this, "factor", "Gain Factor", false, 1.0) { }

  ~PiPoGain (void) { }

  int streamAttributes (bool hasTimeTags, double rate,
    double offset, unsigned int width, unsigned int height,
    const char **labels, bool hasVarSize,
    double domain, unsigned int maxFrames)
  { // can not work in place, create output buffer
    buffer.resize(width * height * maxFrames);

    return propagateStreamAttributes(hasTimeTags, rate,
             offset, width, height, labels,
             hasVarSize, domain, maxFrames);
  }

  int frames (double time, PiPoValue *values,
              unsigned int size, unsigned int num)
  { // get gain factor here, it could change while running
    double f = factor.get();
    PiPoValue *ptr = &buffer[0];

    for (unsigned int i = 0; i < num; i++)
    {
      for (unsigned int j = 0; j < size; j++)
        ptr[j] = values[j] * f;

      ptr    += size;
      values += size;
    }

    return propagateFrames(time, &buffer[0], size, num);
  }
};
```

3.4 Existing Modules

The list of existing *PiPo* modules can be organized into the following categories:

Stream Processing `slice` (windowing), `scale`, `sum`, `select` (get columns),

Filtering `biquad` (biquad filter), `mvavrg` (moving average filter), `median` (median filter), `delta` (derivative), `finitedif` [9], `bayesfilter` [10]

Segmentation `onseg` (segments starting at signal onset), `chop` (segments of regular intervals), `gate` (segments excluding weak sections), `sylseg` [16]

Temporal Modeling `mean`, `std`, `meanstd`, `min`, `max`

Analysis `descr` (basic audio descriptors), `yin` (pitch extractor), `moments` (centroid, spread, skewness, kurtosis), `lpc` (linear predictive coding), `lpcformants` (formant extraction), `psy` (pitch synchronous markers), `ircamdescriptor` [18]

Frequency Domain Processing `fft` (FFT from pre-windowed frames), `dct` (discrete cosine transform), `bands` (MEL bands and similar from power or amplitude spectrum), `mel` (MEL bands from audio stream), `mfcc` (MFCC from audio stream), `wavelet` (wavelet transform from audio stream)

They can be instantiated from C++ code using the precompiled *libpipo* library, thanks to the `PiPoCollection` class defined in the *PiPo* SDK, or be used in one of the environments described in 3.6. The `PiPoCollection` class acts as a module factory. It is able to instantiate *PiPo* graphs from a simple syntax, which can then be used as simple *PiPo*s in a host environment. It also allows users of the API to add their own *PiPo* modules to the original collection. Once added, more complex graphs combining modules from *libpipo* and these new modules can be instantiated and run inside a *PiPo* host.

3.5 PiPo Graph Construction

Graphs of *PiPo* modules can be either constructed in C++ code or — within a given host environment — through expressions of a very simple syntax. For the first case, the API defines a set of primitives that can be used to construct graphs of any complexity by arranging modules in sequence and in parallel. In the latter case, these primitives are instantiated by a parser function provided by the API.

3.5.1 Specific Graph Construction Modules

Additional to the data processing modules listed above, there are two internal modules that handle the connection of processing modules in sequence or in parallel.

The `sequence` module simply connects the upstream module to the downstream one (i.e. setting the latter as receiver, so that the API calls get propagated through the chain). The `parallel` module essentially consists of an ordered set of modules that receive the same input stream and output towards an internal `merge` module. Each incoming frame is processed by each of the parallel modules in the given order, whereby the `merge` module concatenates the output data column-wise into a single matrix that is output towards the receiver of the `parallel` module.

3.5.2 Graph Construction Syntax

The construction of sequences and parallel modules is also available at the user level via a simple syntax inspired by FAUST [17], with the following operators:

: (sequence)

< (branch)

, (parallel)

> (merge)

For example, the `pipo.descr` module described in section 2.3 would be written like this: `slice<yin,fft<sum:scale,moments>>`

A fifth operator, _ (identity), allows the propagation of intermediate analysis results to the end of the graph. Following the sequence and parallel reduction rules from section 2.3, any *PiPo* graph is equivalent to a *PiPo*, and as a consequence must have a single input and a single output, which implies that the graph syntax must contain the exact same number of branch and merge operators.

3.6 Bindings

3.6.1 Max

The *PiPo* modules are available within the MAX visual programming environment via the MuBu package, [19] where they can run in real-time using the `pipo~` and `pipo` MAX objects, or offline using the `mubu.process` object.

3.6.2 Juce, OpenFrameworks, OpenMusic, Unity3D

PiPo has been integrated into the JUCE[11] framework, the creative coding framework OPENFRAMEWORKS,[12] the computer-aided composition environment OPENMUSIC [4] and the game development environment UNITY3D.[13] Most if these developments are based on the IAE (Interactive Audio Engine) library. [20] The library allows for loading a sound file as input of a user-specified *PiPo* chain and to retrieve the result at the output.

4. CONCLUSIONS AND FUTURE WORK

The *PiPo* API and modules are in production use in our department, and with research project partners, and artists in interactive gesture and music installations and digital instruments. We feel it could help a wider community for easy prototyping and transfer to developed products.

The *PiPo* API is currently in the process of being integrated in the RAPIDMIX API, a wider C++ software ecosystem including machine learning, signal feature extraction and audio processing libraries, as a standardized way of building modular signal descriptors and machine learning algorithms, integrating them into a global workflow and allowing users of this ecosystem to build sustainable code on top of a base collection of algorithms, by providing a flexible mean of interaction between its software components.

We made first steps to add an API similar to *PiPo* to also integrate the iterative training of machine learning and data processing easily into the same host environments.

The *PiPo* SDK that supports the development of modules as well as hosts, has been published under the BSD 3-Clause license at https://github.com/Ircam-RnD/pipo-sdk.

5. ACKNOWLEDGMENTS

The development of the *PiPo* API has received support from the RAPID-MIX project (H2020-ICT-2014-1 Project ID 644862), funded by the European Union's Horizon 2020 research and innovation programme.

[11] https://www.juce.com
[12] http://openframeworks.cc
[13] http://unity3d.com

6. REFERENCES

[1] X. Amatriain, J. Bonada, A. Loscos, J. Arcos, and V. Verfaille. Content-based transformations. *Journal of New Music Research*, 32(1):95–114, 2003.

[2] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José Zapata, and Xavier Serra. Essentia: An open-source library for sound and music analysis. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 855–858, New York, NY, USA, 2013. ACM.

[3] W Brent. A Timbre Analysis and Classification Toolkit for Pure Data. In *International Computer Music Conference*, New York City, NY, 2010.

[4] Jean Bresson, Carlos Agon, and Gérard Assayag. OpenMusic – Visual Programming Environment for Music Composition, Analysis and Research. In *ACM MultiMedia (MM'11)*, Scottsdale, United States, 2011.

[5] Neil Burroughs, Adam Parkin, and George Tzanetakis. Flexible scheduling for dataflow audio processing. In *Proceedings of the International Computer Music Conference (ICMC)*, New Orleans, Louisiana, USA, August 2006.

[6] Chris Cannam. The VAMP Audio Analysis Plugin API: A Programmers Guide. http://vamp-plugins.org/guide.pdf, 2008.

[7] Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 835–838, New York, NY, USA, 2013. ACM.

[8] Florian Eyben, Martin Wöllmer, and Björn Schuller. Opensmile: The munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1459–1462, New York, NY, USA, 2010. ACM.

[9] B. Fornberg. Finite difference method. *Scholarpedia*, 6(10):9685, 2011. revision #91262.

[10] Jules Françoise. *Motion-Sound Mapping by Demonstration*. PhD thesis, Université Pierre et Marie Curie, 2015.

[11] Tristan Jehan. Musical signal parameter estimation. Master's thesis, IFSIC, Université de Rennes, France, and Center for New Music and Audio Technologies (CNMAT), University of California, Berkeley, USA, 1997.

[12] Alexander Lerch, Gunnar Eisenberg, and Koen Tanghe. FEAPI: A Low Level Feature Extraction Plugin API. In *8th International Conference on Digital Audio Effects (DAFx05*, 2005.

[13] M. Malt and E. Jourdan. Zsa. Descriptors: a library for real-time descriptors analysis. In *5th Sound and Music Computing Conference*, pages 134–137, Berlin, Germany, August 2008.

[14] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard. YAAFE, an easy to use and efficient audio feature extraction software. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, 2010.

[15] David Moffat, David Ronan, Joshua D Reiss, et al. An evaluation of audio feature extraction toolboxes. In *Proceedings of the COST-G6 Conference on Digital Audio Effects (DAFx)*, Trondheim, Norway, 2015.

[16] Nicolas Obin, François Lamare, and Axel Roebel. Syll-o-matic: an adaptive time-frequency representation for the automatic segmentation of speech into syllables. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[17] Yann Orlarey, Dominique Fober, and Stéphane Letz. Faust: an efficient functional approach to dsp programming. *New Computational Paradigms for Computer Music*, 290, 2009.

[18] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the Cuidado project. Technical Report version 1.0, Ircam – Centre Pompidou, Paris, France, April 2004.

[19] Norbert Schnell, Axel Röbel, Diemo Schwarz, Geoffroy Peeters, and Ricardo Borghesi. MuBu & friends – assembling tools for content based real-time interactive audio processing in Max/MSP. In *Proceedings of the International Computer Music Conference (ICMC)*, Montreal, Canada, August 2009.

[20] Norbert Schnell, Diemo Schwarz, Roland Cahen, and Victor Zappi. IAE & IAEOU. In Roland Cahen, editor, *Topophonie research project : Audiographic cluster navigation (2009-2012)*, Les Carnets d'Experimentation de l'Ecole Nationale Superieure de Creation Industrielle, pages 50–51. ENSCI - Les Ateliers / Paris Design Lab, December 2012.

[21] Diemo Schwarz. Corpus-based concatenative synthesis. *IEEE Signal Processing Magazine*, 24(2):92–104, March 2007. Special Section: Signal Processing for Sound Synthesis.

[22] Diemo Schwarz and Norbert Schnell. A modular sound descriptor analysis framework for relaxed-real-time applications. In *Proceedings of the International Computer Music Conference (ICMC)*, New York, NY, 2010.

RANKING-BASED EMOTION RECOGNITION FOR EXPERIMENTAL MUSIC

Jianyu Fan, Kıvanç Tatar, Miles Thorogood, Philippe Pasquier
Simon Fraser University
Vancouver, Canada
`jianyuf, ktatar, mthorogo, pasquier@sfu.ca`

ABSTRACT

Emotion recognition is an open problem in Affective Computing the field. Music emotion recognition (MER) has challenges including variability of musical content across genres, the cultural background of listeners, reliability of ground truth data, and the modeling human hearing in computational domains. In this study, we focus on experimental music emotion recognition. First, we present a music corpus that contains 100 experimental music clips and 40 music clips from 8 musical genres. The dataset (the music clips and annotations) is publicly available at: http://metacreation.net/project/emusic/. Then, we present a crowdsourcing method that we use to collect ground truth via ranking the valence and arousal of music clips. Next, we propose a smoothed RankSVM (SRSVM) method. The evaluation has shown that the SRSVM outperforms four other ranking algorithms. Finally, we analyze the distribution of perceived emotion of experimental music against other genres to demonstrate the difference between genres.

1. INTRODUCTION

The research in MER proposes computational approaches to recognize the emotion of music. The increasing numbers of MER studies in recent years have been focusing on particular musical genres, such as classical music, pop, rock, jazz, and blues [41]. So far, to our knowledge, MER in experimental music has yet to be explored.

The definition and use of the term experimental music have been an ongoing discussion within the last century. John Cage [15] clarifies the action of experimentalism as “the outcome of which is not foreseen”. Demers [17] defined experimental as “anything that has departed significantly from norms of the time…” [p.7] and continues by the two assumptions of “…that experimental music is distinct from and superior to a mainstream-culture industry and that culture and history determine aesthetic experience" [p.139]. Experimental music does not only rely on harmony and melody [6]. Experimental music explores the continuum between rhythm, pitch, and noise; the notion of organized sound; the expansion of temporal field; and the morphologies of sound. In this study, our definition of experimental music encompasses experimental electronic music such as acousmatic music, electroacoustic music, noise music, soundscape compositions as well as experimental music with acoustic instruments such as free improvisation or improvised music. We also include Contemporary Art practices that use sound as a medium in our definition of experimental music.

There are many applications in which a computational model of MER for experimental music would be beneficial. MER computational models can be used in the system architecture of Musical Metacreation (MuMe) systems for experimental music. MuMe is the partial or complete automation of musical tasks [34]. A variety of MuMe systems apply machine listening. Machine listening is the computational modeling of the human hearing. In that sense, a computational model for MER in experimental music can be useful to design a machine listening algorithm for a MuMe system. Moreover, we can use computational MER models in the analysis of experimental music works. Also, we can design mood enabled recommendation systems for experimental music albums using a MER model for experimental music.

Still, MER has several challenges. First, music perception can be dramatically different if listeners are from different regions of the world and have various unique cultural backgrounds [5,18]. Second, it is difficult for researchers to collect ground truth data to cover a wide range of population that well distributed in different parts of the world [5]. Third, in the previous studies, researchers designed listening tests that asked participants to annotate the music pieces by rating their emotion perception of the music pieces [41,49]. However, the cognitive load of rating emotion is heavy for participants [9]. This causes the low-reliability of the annotations [19,44]. Fourth, the level of participant’s agreement on the emotion of a music clip varies because the perception of music is subjective. Even for one individual, the ratings can change during a day [49]. Fifth, in the case of experimental music emotion recognition, there is no annotated dataset available. The current MIREX MER task is the case of pop music emotion recognition.

To overcome these difficulties, we designed a ranking-based experiment to collect ground truth annotations based on a crowdsourcing method. Crowdsourcing method is to elicit a large amount of data from a large group of people from online communities [8]. Our ground truth annotations were gathered from 823 annotators from 66 countries, which covers diverse cultural backgrounds. Then, to reduce the cognitive load, we used a ranking-based method to ask participants to do pairwise comparisons between experimental music clips. The ranking based approach only needs relative comparisons instead of absolute ratings. This improves the objectiveness of the ground truth data. We applied the Quicksort algorithm to select comparisons during the data collection stage to reduce the workload (see Section 4.1). Then, we proposed a SRSVM method and compared it with other ranking algorithms. The results show that SRSVM is better than four other ranking algorithms regarding experimental music emotion recognition.

The database, containing the 140 music clips and the annotations, can be freely downloaded at http://metacreation.net/project/emusic/. We believe that public release of such a dataset will foster research in the field and benefit MER communities. The main contributions of this paper are thus four-fold:

- We provide a music corpus, EMusic. The corpus includes 100 experimental music clips and 40 mainstream music clips.
- We use a crowdsourcing method to collect the pairwise ranking data for experimental music clips, and share an annotated experimental music dataset.
- We proposed the SRSVM method for experimental music emotion recognition and compared our approach with other ranking algorithms.
- We compared the annotations of experimental music with that of other music genres.

2. RELATED WORKS

The Music Information Research Evaluation eXchange (MIREX) community evaluates systems for Audio Music Mood Classification every year. Studies have been classified into two major categories based on the model of emotion: categorical and dimensional approaches.

2.1 Categorical Approaches in MER

Categorical MER approaches use discrete affect models to estimate emotion. Discrete affect models propose that we can describe all emotions using a set of basic emotions. These basic emotion categories are happiness, sadness, fear, anger and disgust [22, 33], shame, embarrassment, contempt and guilt [3], as well as exuberance, anxious/frantic and contentment [32]. There is still no consensus on the discrete emotion categories of music [32].

In the previous studies with categorical MER approaches, researchers conducted experiments to collect the ground truth annotations. Then, researchers used the audio features of music clips with classification methods to model the relationship between audio features and emotion categories [23, 45, 46].

2.2 Dimensional Approaches in MER

Dimensional affect models use a Cartesian space with continuous dimensions to represent emotions [7,14,40,48]. The simplest dimensional affect model has two dimensions: valence and arousal. Other dimensional affect models with additional dimensional such as tension, potency, and dominance have also been proposed in the literature [32]. MER studies use dimensional affect models to compute continuous values that represent the emotion of audio samples. These studies focus on continuous machine learning models such as regression models. Researchers gather the ground truth data by conducting an evaluation experiment in which the participants label the emotion music clips on a dimensional affect grid.

2.3 Rating or Ranking

Affective ratings instruments have been used for collecting affective annotations. Researchers have used such tools in video emotion recognition [27, 30], music emotion recognition [11], speech emotion recognition [35], soundscape emotion recognition [20] and movement emotion recognition [43]. However, recent studies show that rating based experiments have limitations and fundamental flaws [13]. Rating-based experiments neglect the existence of interpersonal differences on the rating process. In addition, rating emotion in a continuum is difficult because annotators tend to score the samples based on the previous ratings instead of their non-biased feelings [44]. Yang and Lee indicated that the rating-based approach imposes a heavy cognitive load on the subjects [48]. Moreover, the contextual situation of annotators can affect the consistency of ratings [12].

Ranking has been an alternative approach for eliciting responses from subjects [9, 39, 48]. Metallinou and Narayanan found that there is a higher Inter-annotator reliability when people were asked to describe emotions in relative terms rather than in absolute terms [2]. Yannakakis et al. also showed that the inter-rater agreement of the ordinal data is significantly higher than that of the nominal data [12].

Yang and Chen designed a ranking-based experiment to collect ground truth data and build a ranking model recognize the perceived emotion of pop music [9]. The result showed that the ranking-based approach simplifies the annotation process and enhances the Inter-annotator reliability. Hence, we designed a ranking-based method to for experimental music emotion recognition, where annotators made pairwise comparisons between two audio clips based on valence and arousal.

2.4 Emotion Taxonomy

According to previous studies [1, 24], two types of emotions are at play when listening to music.

- Perceived emotion: Emotions that are communicated by the source.
- Induced emotion: Emotional reaction that the source provokes in listeners.

The perceived emotion is more abstract and objective. It is the emotion the source conveys. The perceived emotion of happy songs is always “happy”. However, the induced emotion is more subjective. The same happy music may not necessarily induce happiness in the listener. In this study, we focus on the perceived emotion of music clips because it is more objective.

3. DATA COLLECTION

To build a MER system for experimental music, we first built an experimental music corpus: EMusic. Then, we collected emotion annotations using a crowdsourcing method.

3.1 Corpus Construction

In EMusic corpus, there are 100 experimental music clips and 40 music clips from 8 musical genres, including blues, classical, country, electronic, folk, jazz, pop and rock. The 100 experiment music clips are extracted from 29 experimental music pieces, which are high quality works of Electroacoustic music. The 40 music clips are selected from 1000 songs database [29]. We segmented these compositions using multi-granular novelty segmentation [31] provided in the MIRToolbox [32]. Using this automatic segmentation method, we ensure that each segment is consistent. Then, we manually chose novel clips to create a homogeneous and consistent corpus that would not disturb the listeners. A 0.1 seconds fade in/out effect has been added to each audio clip.

Music clips are converted to a format in wav (44100 Hz sampling frequency, 32 bits precision and mono channel). All the audio samples are normalized. Regarding the duration, Xiao et al. [50] showed that the use of six to eight seconds is good for presenting stable mood for classical music segments. Fan et al. [19] indicated that the duration of six seconds is long enough for soundscape emotion recognition. Following the previous study, we aimed for the average duration of 6 seconds in this experiment (Mean: 6.20s, Std: 1.55s). The duration of clips varies because of the automatic segmentation by novelty.

3.2 Select Comparisons

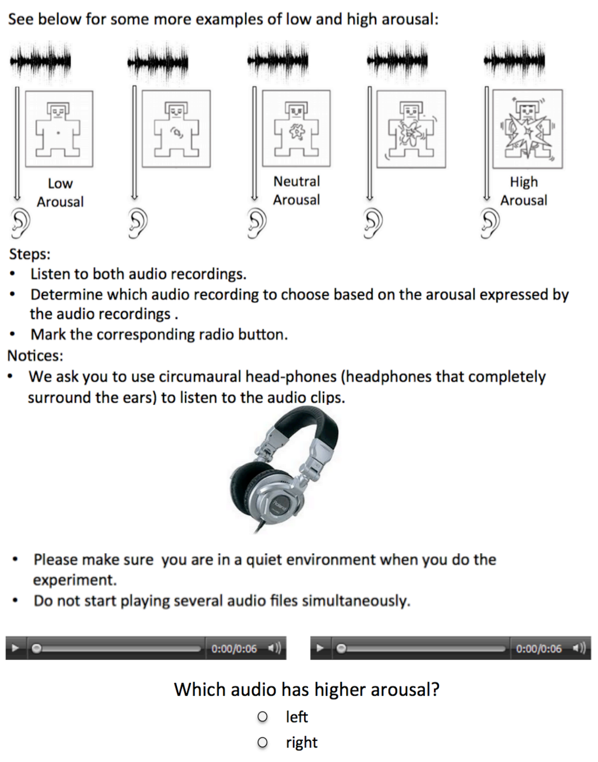
To create a robust set of annotations, we need multiple annotations per pairwise comparison of audio clips. Baveyes et al. [44] found that collecting three annotations per comparison is a good compromise between the cost and the accuracy of the experiment. Therefore, we follow this approach for its feasibility within our experiment.

To efficiently create pairwise comparisons presented to the listeners, we use a Quicksort algorithm [44]. For the first iteration of the algorithm, we select one audio sample as the pivot. All remaining clips are to be compared with the pivot so that the algorithm generates 139 comparisons. We then collect three annotations for each comparison and determine the result to be the one that provided by at least two annotators. In the case that we did not select a pivot that has the lowest or the highest valence or arousal, we end up with two separate sets after the first iteration. Therefore we repeatedly select a new pivot in each set until each audio clip received a rank of valence and a rank of arousal from 1 to 140. The computational complexity of the Quicksort algorithm is O(*NlogN*).

3.3 Online Experiment

We conduct an online experiment to annotate our corpus of experimental music clips with affective labels. We used the CrowdFlower[1] platform to crowd source annotations from people online. To sort the 140 music clips based on valence and arousal independently, we launched one task for valence and another task for arousal.

[1] https://www.crowdflower.com/


See below for some more examples of low and high arousal:

Steps:
- Listen to both audio recordings.
- Determine which audio recording to choose based on the arousal expressed by the audio recordings .
- Mark the corresponding radio button.

Notices:
- We ask you to use circumaural head-phones (headphones that completely surround the ears) to listen to the audio clips.
- Please make sure you are in a quiet environment when you do the experiment.
- Do not start playing several audio files simultaneously.

Which audio has higher arousal?
- left
- right

Figure 1. The interface of crowdsourcing study.

At the beginning of the annotation process, subjects are provided with the terminology of arousal and v dalence. In our experiment, we used valence to describe perceived
pleasantness of the sound. We provided subjects with the Self-Assessment Manikin [28] at the beginning of the task to make sure the task was understood. The Self-Assessment Manikin is a pictorial system used in experiments to represent emotional valence and arousal axes. Its non-verbal design makes it easy to use regardless of age, educational or cultural background. We modified the pictorial system by adding arrows to inform annotators that we were collecting perceived emotion.

We requested annotators to follow a tutorial to get familiar with the annotation interface. Annotators were notified that they were required to use headphones to listen to the audio clips. We asked them to turn the volume up to a comfortable level given a test signal. Annotators were then presented with a quiz, where 5 gold standard comparisons were provided. These comparisons were easily comparable regarding valence and arousal, which were carefully selected by experts. The annotators could continue to the task only if they achieve an 80% of accuracy in the quiz.

To ensure the quality of the annotations, we tracked annotators' performance by inserting gold standard comparisons throughout the tasks. Similar to the comparisons in the quiz, these 5 comparisons were easily comparable regarding valence and arousal. If their answers were not the same as the default answer, they would be noticed by a pop out window. If they had strong reason to explain their answer, they could message the reason to us. This also affects annotators' reputation on the CrowdFlower.

Annotators could listen repeatedly to an audio clip. After an annotator had listened to both audio clips, the option to enter the response was presented in the form of an input button. For easing the fatigue that increases naturally during manual data annotation [2], they could pause the annotation process at any time and continue at a later stage. The volume control bar was disabled so that annotators could not adjust the individual volumes themselves. An annotator had to rank 5 pairs of clips before being paid US$0.05 and was able to exit the task at any time.

3.4 Annotation Results

A total of 823 annotators performed the task from 66 different countries. Most of the workers are Venezuelans (31.71%), Brazilian (6.93%), Serbian (6.44%), Russian (5.95%) and Bosnians (5.10%). The annotators were from the world population and it is unlikely they have a background in experimental music. This avoids the potential bias brought by experts.

Each pair was displayed to annotators until three annotations are collected for this pair. 823 annotators provided 2817 comparisons for arousal and 2445 comparisons for valence. The 823 trusted annotators had an average accuracy of 91.81% in the quiz. Annotators took approximately 13s to perform a task. This also proves that annotators carefully listened to both music clips.

Categories	Arousal	Valence
Percent Agreement	0.839	0.801
Krippendorff's α	0.360	0.222

Table 1. Inter-annotator reliability.

We evaluate the Inter-annotator reliability based on percent agreement and Krippendorff's α. Percent agreement calculates the ratio between the number of annotations that are in agreement and the total number of annotations. However, percent agreement overestimates inter-annotator reliability because it does not consider the agreement expected by chance. Krippendorff's α is more flexible and allows missing data (comparisons can be annotated by any number of workers). Thus, no comparisons are discarded to compute this measure. Their values can range from 0 to 1 for Percent agreement and from -1 to 1 for Krippendorff's alpha.

In Table 1, the inter-annotator reliability is similar to other emotion studies [30, 44]. The percent agreement indicates that annotators agreed on 83.9% and 80.1% of comparisons. The value of Krippendorff's α is between 0.21 to 0.40, which indicates a fair level of agreement.

4. LEARN TO RANK

4.1 Standard Ranking Algorithms

The state-of-the-art ranking algorithms can be three categories: the pointwise approach [42], the pairwise approach [36] and the listwise approach [10]. The pointwise approach learns the score of the samples directly. The pointwise approach takes one train sample at a time and trains a classifier/regressor based on the loss of the single sample. The pairwise approach solves the ranking problems by using a pair of samples to train and provides an optimal ordering for the pair. Listwise methods try to minimize the listwise loss by evaluating the whole ranking list. Each ranking algorithm assigns a ranking score to each sample, and rank the sample based on the score.

In the following, we introduce five ranking algorithms: ListNet, Coordinate Ascent, RankNet, RankBoost and RankSVM. ListNet is a listwise ranking algorithm [10], which uses neural networks to predict the ranking score. The algorithm calculates the probability of the sample ranking within top-k, and computes the difference between the probability distribution of predicted ranks and ground truth data based on cross entropy. Coordinate Ascent algorithm is a gradient-based listwise method for multi-variate optimization [16]. It directly optimizes the mean of the average precision scores for each ranking. RankNet is a pairwise ranking algorithm, which predicts the ranking probability of a pair of samples <A, B>. If sample A receives a higher ranking score than that of sample B, then the object probability $\bar{P}_{AB}$ equals 1, otherwise, $\bar{P}_{AB}$ equals 0. The loss function of RankNet is the cross-entropy between the predicted probability and the object probability. RankBoost is another pairwise ranking algorithm [47]. It replaces training samples with pairs of samples to learn the association between samples. RankSVM is a common pairwise method extended from support vector machines [36]. The difference between features vectors of a pair of training samples can be transformed to a new feature vector to represent the pair. RankSVM converts a ranking task to a classification task.

4.2 Searching Strategies

Given a test sample, a ranking model provides a ranking score regarding valence/arousal. A ranking score is a real number. To obtain the predicted rank of the test sample based on the ranking score, we used two search strategies: one-by-one search and smoothed binary search.

4.2.1 One-by-One Search

First, we obtain predicted ranking scores of the entire training set and the test sample. Then, we sorted all clips by ranking score to obtain the predicted ranking of the test sample. Ties are unlikely to happen since we set the value of the score retains 6 digits after the decimal point.

4.2.2 Smoothed Binary Search

Smoothed binary search compares the ranking score of a test sample with the ranking scores of pivots selected from the training set to find the rankings of a test sample along the valence/arousal axis. We add a smoothed window to traditional binary by selecting a group of pivots instead of one pivot. Following is the description of the smoothed binary search:

- Given a test sample, pick an odd number of clips from the training set that are consecutive on the valence/arousal axis as pivots. The odd number of clips avoids the ties. The group of pivots has the medium value of valence/arousal among the subset.
- Predict the ranking score for the group of pivots and the test sample, and compare their ranking score. The test sample with a score of less than half of the pivots comes before the pivots, while the test

sample with a score greater than half of the pivots comes after pivots.

- Recursively apply the above steps until the size of subsets is 2. The average ranking of these two training samples is the predicted rankings.

4.3 SRSVM

We propose the SRSVM for experimental music emotion recognition. The training of SRSVM is the same as standard RankSVM. During the testing/ranking stage, SRSVM finds the predicted ranking of the test sample based on the smoothed binary search.

5. PERFORMANCE ANALYSIS

5.1 Features Selection

We began with a feature set including rms, brightness, loudness, spectral slope, spectral flux, spectral rolloff, attack leap, regularity, pulse clarity, hcdf, inharmonicity, perceptual sharpness, pitch, key, tempo, and 12 MFCCs. We used 23-ms analysis windows and calculated the mean and standard deviation to represents signals as the long-term statistical distribution of local spectral features, which ended up with a 56-dimension feature vector [21]. We used MIRToolbox [32] and YAAFE [4] libraries to extract audio features.

Selected Features
Mean of Root Mean Square
Standard deviation of Root Mean Square
Standard deviation of Brightness
Mean of MFCC 1
Standard deviation of MFCC 2
Standard deviation of MFCC 8
Mean of MFCC 12
Mean of Hcdf
Mean of Loudness
Standard deviation of Loudness
Mean of Regularity

Table 2. Selected features for predicting valence/arousal

Before training the model, we build a feature selector that removes all low-variance features over the entire corpus to select a subset of discriminative features. The threshold of variance is 0.02, which is chosen as a heuristic value. This step kept 43 features out of 56 features. Then, we used a random forests method, which has ten randomized decision trees to evaluate the importance of features based on the Gini impurity index. We ended up having an 11-dimensional feature vector (see Table. 2). Because our dataset includes 100 experimental music clips and 40 clips belong to other genres, we tested the ranking algorithms using the whole dataset and the subset of experiment music separately.

5.2 Comparing with Ranking Algorithms

We evaluate the ranking algorithms of experimental MER using Goodman-Kruskal gamma (G). Goodman-Kruskal gamma measures the association between the predicted rankings and the ground truth annotations [37, 38]. G depends on two measures: the number of pairs of cases ranked in the same order on both variables (number of concordant, N_S) and the number of pairs of cases ranked in reversed order on both variables (number of discordant, N_D). G ignores ties. In our experiment, we had no ties. G is close to 1 indicate strong agreement, -1 for total disagreement, and 0 if the rankings are independent.

$$G = \frac{N_S - N_D}{N_S + N_D} \tag{1}$$

We used the leave-one-out validation method to compare the SRSVM with ListNet, RankNet, Coordinate Ascent, and RankBoost. For a given test sample, ranking algorithms output a predicted valence/arousal score. To obtain the predicted rankings of the whole test set, we used one-by-one searching strategy and smoothed binary search strategy. Then, we measured the gamma between the predicted rankings and the ground truth annotation.

As we can see from Table 3, when we use SRSVM, we obtain the best performance when the windows size is three samples (G: 0.733, $p < 0.001$). When the window size is 1, the test sample will be compared with one pivot iteratively until it falls into a small interval. This becomes a standard binary search. After adding a smoothed window, the test sample is compared with a group of pivots. This increases the accuracy of predicting whether the test sample is larger or smaller than the pivots.

Algorithm	One-by-One Search	Smoothed Binary Search (Number of samples)		
		1	3	5
ListNet	0.044	0.088	0.057	0.022
RankNet	0.096	0.386	0.269	0.255
Coordinate Ascent	0.191	0.436	0.387	0.486
Rank-Boost	0.619	0.679	0.697	0.717
RankSVM	0.398	0.690	**0.733** **SRSVM**	0.697 **SRSVM**

Table 3. Goodman-Kruskal gamma of ranking algorithms for arousal recognition using the whole dataset

Method	One-by-One Search	Smoothed Binary Search (Number of samples)		
		1	3	5
ListNet	0.015	0.002	0.049	0.002
RankNet	0.063	0.155	0.055	0.260
Coordinate Ascent	0.016	0.130	0.195	0.254
Rank-Boost	0.438	0.467	0.345	0.440
RankSVM	0.333	0.490	**0.573** **SRSVM**	0.556 **SRSVM**

Table 4. Goodman-Kruskal gamma of ranking algorithms for valence recognition using the whole dataset

When using the whole dataset, the valence recognition is harder than arousal recognition. However, the SRSVM still obtains the best performance (G: 0.573, $p < 0.001$).

Method	One-by-One Search	Smoothed Binary Search (Number of samples)		
		1	3	5
ListNet	0.001	0.037	-0.013	0.013
RankNet	0.110	0.096	0.242	0.299
Coordinate Ascent	0.237	0.515	0.519	0.556
Rank-Boost	0.698	0.741	0.740	0.748
RankSVM	0.300	0.776	**0.801** **SRSVM**	0.776 **SRSVM**

Table 5. Goodman-Kruskal gamma of ranking algorithms for arousal recognition using the subset that only contains experimental music clips.

As Table 5 shows, when we only consider experimental music, the Gamma statistic of SRSVM for arousal recognition has the best result (G: 0.801, $p < 0.001$). The results of the experimental music case are better than the results of the case including clips of all genres.

Method	One-by-One Search	Smoothed Binary Search (Number of samples)		
		1	3	5
ListNet	0.115	0.037	-0.012	0.036
RankNet	0.058	0.116	0.246	0.277
Coordinate Ascent	0.067	0.100	0.131	0.106
Rank-Boost	0.167	0.236	0.279	0.346
RankSVM	0.434	0.570	**0.795** **SRSVM**	0.628 **SRSVM**

Table 6. Goodman-Kruskal gamma of ranking algorithms for valence recognition using the subset that only contains experimental music clips.

Table 6 shows that when we only consider experimental music, the Gamma statistic of SRSVM for valence recognition (G: 0.795, $p < 0.001$) is significantly higher than using the whole dataset.

From Table 3-6, we can see the best performing model is SRSVM with 3 samples as the smoothed window. The second best performing model is SRSVM with 5 samples as the smoothed window. This result implies that a good emotion-recognition can be obtained by using SRSVM.

5.3 Comparing between Experimental Music and Other Genres

We convert the rankings to ratings to visualize the distribution of the ranking data. This illustration has two assumptions. First, the distances between two successive rankings are equal. Second, the valence and arousal are in the range of [-1.0, 1.0].

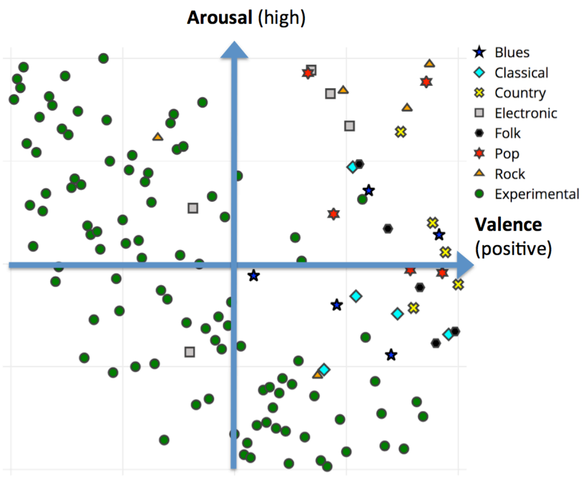


Figure 2. The distribution of the ground truth annotations, the green dots represent experimental music clips

From Figure 2, it can be observed that other genres have both higher perceived valence and arousal comparing to experimental music. Because we have only 5 samples per genre, we need to have a large ground truth dataset to prove that assumption. The figure also shows the negative correlation between valence and arousal of experimental music clips. To test this, we run a Pearson correlation test on the ground truth data. Our Pearson correlation coefficient is -0.3261, which indicates there is a weak negative correlation between the two dimensions.

6. CONCLUSIONS AND FUTURE WORKS

We present an annotated dataset for experimental music emotion recognition. 140 music clips are ranked along the valence and arousal axis through a listening experiment. It is available at http://metacreation.net/project/emusic/. We presented a SRSVM method to predict rankings of experimental music clips regarding valence/arousal and compared SRSVM with other ranking method. We also compared the valence and arousal of experimental music with that of the music of other genres, which shows other genres of music have both higher perceived valence and arousal than experimental music.

Even with the smaller number of clips, we found other genres have both higher perceived valence and arousal comparing to experimental music. In the future, we plan to compare the perceived emotion of different genres by collecting a larger dataset.

7. REFERENCES

[1] A. Kawakami, K. Furukawa, K. Katahira and K. Okanoya, “Sad music induces pleasant emotion,” Front Psychol Vol. 4, No. 311, 2013.

[2] A. Metallinou and S. Narayanan, “Annotation and processing of continuous emotional attributes: challenges and opportunities,” *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition*, pp. 1–8, 2013.

[3] A. Ortony and T. J. Turner, “What’s basic about basic emotions?” *Psychological review.* Vol. 97, No. 3, pp. 315-331, 2014.

[4] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard, “Yaafe, an Easy to Use and Efficient Audio Feature Extraction Software,” *Proceedings of the International Symposium on Music Information Retrieval*, pp. 441–446. 2010.

[5] 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, 2012.

[6] C. Palombini, “Pierre Schaeffer. 1953: Towards an Experimental Music,” *Music and Letters*, Vol. 74, No. 4, pp. 542–57, 1993.

[7] D. Liu, L. Lu, and H.-J. Zhang, “Automatic mood detection from acoustic music data,” *Proceedings of the International Symposium Music Information Retrieval,*, pp. 81–87, 2003.

[8] D. McDuff, “Crowdsourcing affective responses for predicting media effectiveness,” *Ph.D. Dissertation*. Massachusetts Institute of Technology, 2014.

[9] D. Yang and W.-S. Lee, “Disambiguating music emotion using software agents,” *Proceedings of the International Conference on Music Information* Retrieval, 2004.

[10] F. Xia, T.-Y. Liu, J. Wang, W.-S. Zhang, and H. Li, “Listwise approach to learning to rank: Theory and algorithm,” *Proceedings of the IEEE International Conference on Machine. Learning*, pp. 1192–1199, 2008.

[11] F. Weninger, F. Eyben, and B. Schuller, “On-line continuous-time music mood regression with deep recurrent neural networks,” *Proceedings of the IEEE International Conference Acoustics, Speech and Si*gnal Processing, 2014.

[12] G. N. Yannakakis and H. P. Matínez, “Grounding Truth via Ordinal Annotation,” *Proceedings of the International Conference on Affective Computing and Intelligent Interaction*, 2015.

[13] G. N. Yannakakis, H. P. Mart´ınez, “Ratings are overrated!” *Frontiers on Human-Media Interaction*, Vol. 2, No. 13, 2015.

[14] J. A. Sloboda and P. N. Juslin, “Psychological perspectives on music and emotion,” in *Music and Emotion: Theory and Research*, Oxford University Press, 2001.

[15] J. Cage, *Silence: Lectures and Writings*, Wesleyan, 1961.

[16] J. Chen, C. Xiong, and J. Callan, “An empirical study of learning to rank for entity search,” *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2016.

[17] J. Dermers, *Listening through the Noise: The Aesthetics of Experimental Electronic Music*, Oxford University Press, 2010.

[18] J. Fan and M. Casey, “Study of Chinese and UK hit songs prediction,” *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pp. 640–652, 2013.

[19] J. Fan, M. Thorogood, P. Pasquier, “Automatic Soundscape Affect Recognition Using A Dimensional Approach,” *Journal of the Audio Engineering Society*, Vol. 64, No. 9, pp. 646–653, 2016.

[20] J. Fan, M. Thorogood, and P. Pasquier, “Automatic Recognition of Eventfulness and Pleasantness of Soundscape,” *Proceedings of the 10th Audio Mostly*, 2015.

[21] J. J. Aucouturier and B. Defreville, “Sounds like a park: A computational technique to recognize soundscapes holistically, without source identification,” *Proceedings of the International Congress on Acoustics*, pp. 621–626, 2009.

[22] J. Panksepp: *Affective Neuroscience: The Foundation of Human and Animal Emotions*, Oxford University Press, 1998.

[23] K. Bischoff, C. S. Firan, R. Paiu, W. Nejdl, C. Laurier, and M. Sordo, “Music mood and theme classification—a hybrid approach,” *Proceedings of the International Conference on Music Information Retrieval*, pp. 657-662, 2009.

[24] K. Kallinen and N. Ravaja, N, “Emotion perceived and emotion felt: Same and different,” *Musicae Scientiae*, Vol. 5, No. 1, pp. 123-147, 2006.

[25] K. Svore, L. Vanderwende, and C. Burges, “Enhancing single-document summarization by combining RankNet and third-party sources,” *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 448–457, 2007.

[26] L. A. Goodman and W. H. Kruskal, “Measures of Association for Cross Classifications,” *Journal of the American Statistical Association*. Vol. 49, No. 268, pp-732–764, 1954.

[27] L. Devillers, R. Cowie, J.-C. Martin, E. Douglas-Cowie, S. Abrilian, and M. McRorie, “Real life emotions in French and English TV video clips: an integrated annotation protocol combining continuous and discrete approaches,” *Proceedings of the International conference on Language Resources and Evaluation*, 2006.

[28] M. M. Bradley and P. J. Lang, "Measuring emotion: the self- assessment manikin and the semantic differential," *Journal of behavior therapy and experimental psychiatry*, Vol. 25, No. 1, pp. 49–59, 1994.

[29] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," *Proceedings of the 2nd ACM International Workshop on Crowdsourcing for Multimedia*, pp. 1–6, 2013.

[30] N. Malandrakis, A. Potamianos, G. Evangelopoulos, and A. Zlatintsi, "A supervised approach to movie emotion tracking," *Proceedings of the IEEE International Conference Acoustics, Speech and Signal Processing*, pp. 2376–2379, 2011

[31] O. Lartillot, D. Cereghetti, K. Eliard, and D. Grandjean, "A simple, high-yield method for assessing structural novelty," *Proceedings of the 3rd International Conference on Music & Emotion,* 2013

[32] O. Lartillot, P. Toiviainen, and T. Eerola, "A Matlab Toolbox for Music Information Retrieval," in: C. Preisach, H. Burkhardt, L. Schmidt-Thieme, P. D. R. Decker, (Eds), *Data Analysis, Machine Learning and Applications. Studies in Classification, Data Analysis, and Knowledge Organization,* pp. 261–268, Springer, Berlin, Heidelberg, 2008.

[33] P. Ekman, "An argument for basic emotions," *Cognition and Emotion*. Vol. 6, No. 3, pp.169–200, 1992.

[34] P. Pasquier, A. Eigenfeldt, O. Bown, and S. Dubnov, "An Introduction to Musical Metacreation," *ACM Computers In Entertainment, Special Issue: Musical Metacreation*, Vol. 14, No. 2, 2016.

[35] R. Elbarougy and M. Akagi, "Speech Emotion Recognition System Based on a Dimensional Approach Using a Three-Layered Model," *Proceedings of the Signal & Information Processing Association Annual Summit and Conference*, 2012.

[36] R. Herbrich, T. Graepel, and K. Obermayer, "Support vector learning for ordinal regression," *Proceedings of International Conference on. Artificial* Neural *Network*, 1999.

[37] R. Morris, "Crowdsourcing workshop: The emergence of affective crowdsourcing," *Proceedings of the Annual Conference Extended Abstracts on Human Factors in Computing Systems*, 2011.

[38] R. Morris and D. McDuff, "Crowdsourcing techniques for affective computing," in R.A. Calvo, S.K. DMello, J. Gratch and A. Kappas (Eds). *Handbook of Affective Computing*, Oxford University Press, 2014.

[39] S. Ovadia, "Ratings and rankings: Reconsidering the structure of values and their measurement," *International Journal of Social Research Methodology*, Vol. 7, No. 5, pp. 403–414, 2004.

[40] T. Eerola, O. Lartillot, and P. Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models," *Proceedings of the International Symposium Music Information Retrieval*, pp. 621–626, 2009.

[41] T. Eerola, Tuomas, and J. K. Vuoskoski. "A Review of Music and Emotion Studies: Approaches, Emotion Models, and Stimuli," *Music Perception: An Interdisciplinary Journal*, Vol. 30, No. 3, pp. 307–340, 2013.

[42] T. Y. Liu, "The Pointwise Approach," in *Learning to rank for information retrieval*. Berlin: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2011.

[43] W. Li, P. Pasquier, "Automatic Affect Classification of Human Motion Capture Sequences in the Valence-Arousal Model," *Proceedings of the International Symposium on Movement and Computing*, 2016.

[44] Y. Baveye, E. Dellandrea, C. Chamaret, and L. Chen, "LIRIS-ACCEDE: A video database for affective content analysis," *IEEE Transactions on Affective Computing*, Vol. 6, No. 1, pp. 43–55, 2015.

[45] Y. Feng, Y. Zhuang, and Y. Pan, "Popular music retrieval by detecting mood," *Proceedings of the International Conference on Information Retrieval*, pp. 375–376, 2013.

[46] 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," *Proceedings of the International Conference on Music Information Retrieval*, 2010.

[47] Y. Freund, R. Iyer, R. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Proceedings of the International Conference on Machine Learning*, pp. 170–178, 1998.

[48] Y.-H. Yang and H. Chen, "Ranking-based emotion recognition for music organization and retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 19, No. 4, pp. 762– 774, 2011.

[49] Y.-H. Yang and H.-H. Chen, "Machine recognition of music emotion: A review," *ACM Trans. Intel. Systems & Technology*, Vol. 3, No. 3, 2012.

[50] Z. Xiao, E. Dellandrea, W. Dou, and L. Chen, "What is the best segment duration for music mood analysis?" *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, pp. 17–24, 2008.

SCALE- AND RHYTHM-AWARE MUSICAL NOTE ESTIMATION FOR VOCAL F0 TRAJECTORIES BASED ON A SEMI-TATUM-SYNCHRONOUS HIERARCHICAL HIDDEN SEMI-MARKOV MODEL

Ryo Nishikimi[1] **Eita Nakamura**[1] **Masataka Goto**[2] **Katsutoshi Itoyama**[1] **Kazuyoshi Yoshii**[1,3]
[1]Graduate School of Informatics, Kyoto University, Japan [3]RIKEN AIP, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Japan
{nishikimi, enakamura, itoyama, yoshii}@sap.ist.i.kyoto-u.ac.jp, m.goto@aist.go.jp

ABSTRACT

This paper presents a statistical method that estimates a sequence of musical notes from a vocal F0 trajectory. Since the onset times and F0s of sung notes are considerably deviated from the discrete tatums and pitches indicated in a musical score, a score model is crucial for improving time-frequency quantization of the F0s. We thus propose a hierarchical hidden semi-Markov model (HHSMM) that combines a score model representing the rhythms and pitches of musical notes with musical scales with an F0 model representing time-frequency deviations from a note sequence specified by a score. In the score model, musical scales are generated stochastically. Note pitches are then generated according to the scales and note onsets are generated according to a Markov process defined on the tatum grid. In the F0 model, onset deviations, smooth note-to-note F0 transitions, and F0 deviations are generated stochastically and added to the note sequence. Given an F0 trajectory, our method estimates the most likely sequence of musical notes while giving more importance on the score model than the F0 model. Experimental results showed that the proposed method outperformed an HMM-based method having no models of scales and rhythms.

1. INTRODUCTION

Singing voice analysis is important for music information retrieval because a singing voice usually forms a large part of the melody line of popular music, and provides much information about music. Singing voice analysis techniques such as vocal F0 estimation [1, 3, 7, 9, 14] and singing voice separation [8, 12] have actively been studied and applied to singer identification [10, 22], karaoke generation [19], query-by-humming [8], and active music listening [6]. To leverage musical information conveyed by singing voices, it is helpful to convert a vocal F0 trajectory to a musical score containing only discrete symbols.

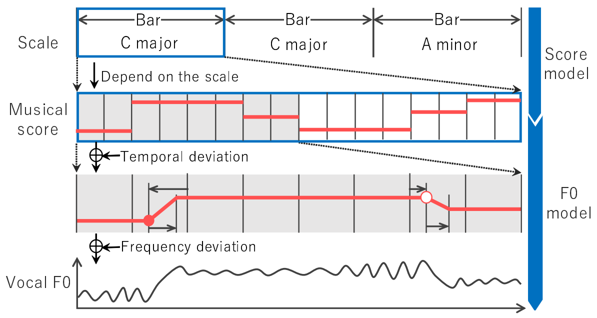


Figure 1: The generative process of a vocal F0 trajectory based on a hierarchical hidden semi-Markov model involving a score model and an F0 model.

In this study, we tackle musical note estimation for vocal F0 trajectories that tend to have large deviations from original musical scores. The pitches and onset times of musical notes in a musical score can take only discrete values, whereas an F0 trajectory is a continuous signal that can dynamically and smoothly vary over time. F0 trajectories are modulated by vibrato and changes smoothly from one note to another by a portamento. Naive time-frequency quantization of an F0 trajectory therefore outputs a note sequence that often includes statistically-rare chromatic note progressions with unlikely rhythms.

To solve this problem, we propose a statistical method of scale- and rhythm-aware musical note estimation based on integration of a score model describing the process of generating a note sequence and an F0 model describing the process of generating an F0 trajectory from the note sequence (Fig. 1). In the score model, a sequence of musical scales (local keys) is determined by a Markov process and the semitone-level pitch of each note is then determined according to both a scale of the note position and the pitch of a previous note. The onset position of each note on a tatum grid is determined according to that of a previous note to make rhythmic structures. In the F0 model, the time-frequency deviations are added to a step-function-shaped F0 trajectory corresponding to a musical score given by the score model. The integrated model is thus formulated as a hierarchical hidden semi-Markov model (HHSMM). Given a vocal F0 trajectory with a tatum grid, the scales, musical notes, and F0 deviations, which are all latent variables of the proposed model, are jointly estimated by using a Markov chain Monte Carlo algorithm. A key feature of our method is that musical scales and rhythms work as self-organizing constraints on time-frequency quantization of vocal F0 trajectories.

2. RELATED WORK

In this section, we introduce related work on the analysis of singing voices.

2.1 Vocal F0 Estimation for Music Audio Signals

Estimation of vocal F0 trajectories for music audio signals has actively been studied [1, 3, 7, 9, 14], and the outputs of these methods can be used as inputs of our method. One of the most basic method is subharmonic summation (SHS) [7] that calculates the sum of the harmonic components of each candidate F0. Ikemiya *et al.* [9] improved F0 estimation based on SHS and singing voice separation based on robust principle component analysis (RPCA) [8] by using the mutual dependency of those two tasks. Salamon *et al.* [21] estimated contours of the melody F0 candidates by calculating a salience function and then recursively removed contours which do not form a melody line by using the characteristics of each contour. Durrieu *et al.* [3] extracted a main melody by representing accompaniments with a model inspired by non-negative matrix factorization (NMF) and leading voices with a source-filter model. Mauch *et al.* [14] modified the YIN [1] in a probabilistic way so that the modified system could determine multiple candidate fundamental frequencies and then select one at each frame by using an HMM.

2.2 Musical Note Estimation for Singing Voices

Estimation of musical notes from sung melody has been a hot research topic [6, 11, 13, 15, 17, 18, 20, 23]. A naive method is to take the majority of vocal F0s in each interval of a regular grid [6]. Paiva *et al.* [17] proposed a cascading method based on multipitch detection, multipitch trajectory construction, segmentation of multipitch trajectory, elimination of irrelevant notes, and extraction of notes that form a main melody. Raphael [18] proposed an HMM-based method that estimates pitches, rhythms, and tempos when the number of notes is given. The rhythm and onset deviation models used in [18] are similar to those used in our method. Laaksonen *et al.* [11] divided audio data into segments corresponding to scales and notes by focusing on the boundaries of chords given as input, and independently estimated the notes based on a score function. Ryynänen *et al.* [20] proposed a method based on a hierarchical HMM in order to capture the different kinds of vocal fluctuations (e.g., vibrato and portamento) within one note. In this model, the transition between pitches is represented in the upper-level HMM and the transition between the vocal fluctuations is represented in the lower-level HMM. Molina *et al.* [15] focused on the hysteresis characteristics of vocal F0s. Nishikimi *et al.* [16] proposed a method based on an HHM that represents the generative process of a vocal F0 trajectory considering the time and frequency deviations. Yang *et al.* [23] proposed a method based on a hierarchical HMM that represents the generative process of the f_0-Δf_0 plane. Mauch *et al.* [13] developed a software tool called Tony for extracting pitches. In this tool, a vocal F0 trajectory is estimated by PYIN [14], and musical notes are estimated by a modified version of Ryynänen's method [20].

3. PROPOSED METHOD

This section explains the proposed method of estimating a sequence of musical notes from a vocal F0 trajectory. The method is based on an HHSMM (Fig. 1) that stochastically generates the F0 trajectory with time-frequency deviations from a sequence of musical notes depending on musical scales. The upper part of the proposed model is an HMM that stochastically generates a sequence of musical notes according to the scales that are assigned to bars. The lower part is an HSMM that represents the musical notes and temporal deviations as latent variables and the frequency deviations as F0 emission probabilities.

3.1 Problem Specification

The problem we tackle is defined as follows:

Input: A vocal F0 trajectory $\boldsymbol{X}=\{x_t\}_{t=1}^{T}$ and 16th-note-level tatums $\boldsymbol{Y}=\{(u_n, v_n)\}_{n=0}^{N}$,
Output: A sequence of notes $\boldsymbol{Z}=\{z_j=(p_j, l_j)\}_{j=0}^{J}$,

where T is the number of frames in a vocal F0 trajectory, x_t is a log frequency at time t, and N is the number of 16th-note-level tatums. $u_n \in \{1, \ldots, T+1\}$ is the time of tatum n and the beginning and end of music are represented as $u_0 = 1$ and $u_N = T+1$, respectively. $v_n \in \{0, \ldots, 15\}$ is the relative position of tatum n in a bar. J is the number of musical notes estimated by proposed methods, and the j-th note z_j is represented as a pair consisting of an pitch $p_j \in \{1, \ldots, K\}$ and a note length $l_j \in \{1, \ldots, L\}$ in the unit of tatums, where K is the number of kinds of semitone-level pitches, and p_j indicates any one in $\{\mu_1, \ldots, \mu_K\}$, which is a set of log frequencies corresponding to semitone-level pitches. For convenience we introduce the initial note z_0 that does not appear in the actual score.

3.2 Probabilistic Modeling of Musical Scores

This section describes the score model constructed with an HMM that represents rhythms and pitches of musical notes under musical scales.

3.2.1 Modeling Scale Transitions

Scales are represented as $\boldsymbol{S} = \{s_m\}_{m=0}^{M}$, where M denotes the number of bars in the musical piece and s_m denotes the scale at the m-th bar. For convenience, we introduce the initial bar s_0 to which the initial note z_0 belongs. Instead of fixing one scale for the whole piece, the scale is allowed to change at bar lines. Each scale s_m takes one of the 24 values of {C, C#, $\cdots$, B} × {major, minor}. The latent variables $\boldsymbol{S}$ are described by a Markov chain as

$$p(s_0|\boldsymbol{\pi}) = \pi_{s_0}, \tag{1}$$

$$p(s_m|s_{m-1}, \boldsymbol{\xi}_{s_{m-1}}) = \xi_{s_{m-1}s_m}, \tag{2}$$

where $\boldsymbol{\pi}\in\mathbb{R}_{\geq 0}^{24}$ is a set of initial probabilities and $\boldsymbol{\xi}_s\in\mathbb{R}_{\geq 0}^{24}$ is a set of transition probabilities.

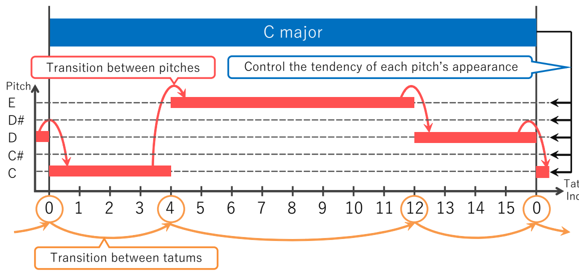
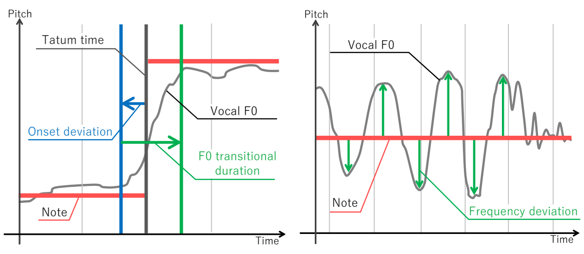

C major
Transition between pitches
Control the tendency of each pitch's appearance
Pitch
E
D#
D
C#
C
Tatum Index
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 0
Transition between tatums

Figure 2: Overview of the score model.



(a) Temporal deviations (b) Frequency deviations

Figure 3: Deviations in a vocal F0 trajectory.

3.2.2 Modeling Pitch Transitions

The sequence of pitches $\boldsymbol{P}$ is generated by a Markov chain depending on scales $\boldsymbol{S}$ as follows (Fig. 2):

$$p(p_0|s_0, \boldsymbol{\phi}_{s_0}) = \phi_{s_0 p_0}, \tag{3}$$

$$p(p_j|p_{j-1}, s_m, \boldsymbol{\psi}_{s_m p_{j-1}}) = \psi_{s_m p_{j-1} p_j}, \tag{4}$$

where $\boldsymbol{\phi}_s \in \mathbb{R}^K_{\geq 0}$ is a set of initial probabilities, $\boldsymbol{\psi}_{sp} \in \mathbb{R}^K_{\geq 0}$ is a set of transition probabilities, and m is the index of a bar to which the note z_j belongs. Moreover, $\phi_{s_0 p_0}$ and $\psi_{s_m p_{j-1} p_j}$ are defined as

$$\phi_{s_0 p_0} = \frac{\hat{\phi}_{\hat{s}_0 \deg(p_0; s_0)}}{\sum_{p=1}^{K} \hat{\phi}_{\hat{s}_0 \deg(p; s_0)}}, \tag{5}$$

$$\psi_{s_m p_{j-1} p_j} = \frac{\hat{\psi}_{\hat{s}_m \deg(p_{j-1}; s_m) \deg(p_j; s_m)}}{\sum_{p=1}^{K} \hat{\psi}_{\hat{s}_m \deg(p_{j-1}; s_m) \deg(p; s_m)}}, \tag{6}$$

where $\hat{s} \in \{\text{major}, \text{minor}\}$ is the mode of scale s and $\deg(p; s) \in \{0, \ldots, 11\}$ is the degree of pitch p in scale s (defined as the relative pitch class of p from the tonic of scale s). $\hat{\phi}_*$ and $\hat{\psi}_*$ are the initial and transition probabilities of pitch classes, given the scales.

3.2.3 Modeling Onset Transitions

Considering the transition between onset positions of adjacent notes, the model makes $\boldsymbol{Z}$ have the plausible rhythm. Let $r_{j-1} \in \{v_n\}_{n=1}^N$ be the onset position of the j-th note z_j. The transition probability is given by

$$p(r_j|r_{j-1}, \boldsymbol{\zeta}_{r_{j-1}}) = \zeta_{r_{j-1} r_j}, \tag{7}$$

where the distance between r_{j-1} and r_j indicates the note value l_j of note z_j. We assume that $r_0 = v_0$ and $r_J = v_N$.

3.3 Probabilistic Modeling of F0 Trajectories

The section describes the F0 model based on an HSMM that represents the generative process of a vocal F0 trajectory. In our model, the pitches, onsets, and temporal deviations are represented as latent variables, and the frequency deviations are represented as emission probabilities.

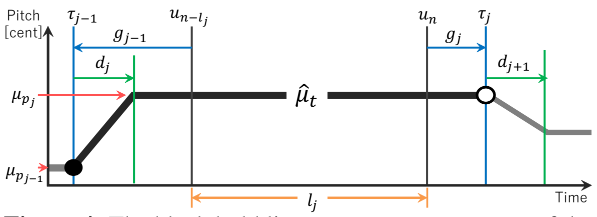


Figure 4: The black bold line represents a sequence of the location parameters of the Cauchy distributions.

3.3.1 Modeling Temporal Deviations

We assume that vocal F0 trajectories include the following two types of temporal deviations (Fig. 3a):

Onset deviation: the gap between the vocal onset time and the note onset time.

F0 transitional duration: the time it takes for singing voices to finish transitioning from one pitch to the next.

The onset deviations $\boldsymbol{G} = \{g_j\}_{j=0}^J$ accompanying with $\boldsymbol{Z}$ are represented as discrete latent variables. Each g_j can take an integer value between $-G$ and G. As with the onset position model, g_{j-1} denotes the onset deviation of note z_j. We assume that each g_j is independently generated by

$$p(g_j|\boldsymbol{\rho}) = \rho_{g_j}, \tag{8}$$

where $\boldsymbol{\rho} \in \mathbb{R}^{2G+1}_{\geq 0}$ is a set of onset deviation probabilities. We assume that there are no deviations for the onset of the first note and the offset of the last note, *i.e.*, $g_0 = g_J = 0$.

The F0 transitional durations $\boldsymbol{D} = \{d_j\}_{j=1}^J$ accompanying with $\boldsymbol{Z}$ are also represented as discrete latent variables. Each d_j can take a value from 1 to D. The continuous transition of vocal F0s between notes z_{j-1} and z_j is represented by a slanted line spanning d_j frames. We assume that each d_j is independently generated as follows:

$$p(d_j|\boldsymbol{\eta}) = \eta_{d_j}, \tag{9}$$

where $\boldsymbol{\eta} \in \mathbb{R}^D_{\geq 0}$ is a set of duration probabilities.

3.3.2 Modeling Frequency Deviations

The vocal F0 trajectory $\boldsymbol{X} = \{x_t\}_{t=1}^T$ is generated by imparting probabilistic frequency deviations to the sequence of notes to which probabilistic temporal deviations have already been imparted (Fig. 3b). Assuming that x_t is independently generated at each frame, the emission probability of the j-th note z_j is given by

$$\begin{aligned} &p(x_{\tau_{j-1}:\tau_j-1}|p_{j-1}, p_j, l_j, g_{j-1}, g_j, d_j, \hat{\mu}_t, \lambda) \\ &= \prod_{t=\tau_{j-1}}^{\tau_j-1} \{\delta_{x_t,\text{voiced}} \text{Cauchy}(x_t|\hat{\mu}_t, \lambda) + \delta_{x_t,\text{unvoiced}}\} \\ &= e_{p_{j-1} p_j l_j g_{j-1} g_j d_j}, \end{aligned} \tag{10}$$

where $x_{\tau':\tau-1}$ indicates $x_{\tau'}, \ldots, x_{\tau-1}$, λ is a scale parameter that represents the scale of the frequency deviations, δ is Kronecker's delta, and $\hat{\mu}_t$ (Fig. 4) is a location parameter given by

$$\hat{\mu}_t = \begin{cases} \frac{\mu_{p_j} - \mu_{p_{j-1}}}{d_j}(t - \tau_{j-1}) + \mu_{p_{j-1}} & (\tau_{j-1} \leq t < \tau_j + d_j) \\ \mu_{p_j} & (\tau_{j-1} + d_j \leq t < \tau_j) \end{cases}. \tag{11}$$

When the onset of note z_{j+1} is located at the n-th tatum, $\tau_j = u_n + g_j$ and $\tau_{j-1} = u_{n-l_j} + g_{j-1}$.

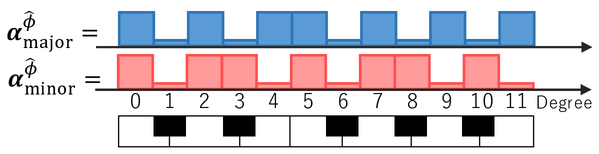

$\boldsymbol{a}^{\hat{\phi}}_{\text{major}}$ =
$\boldsymbol{a}^{\hat{\phi}}_{\text{minor}}$ =
0 1 2 3 4 5 6 7 8 9 10 11 Degree

Figure 5: The configuration of the hyperemarameter $\boldsymbol{a}^{\hat{\phi}}_{\hat{s}}$.

3.4 Prior Distributions

We put conjugate Dirichlet priors on categorical model parameters $\boldsymbol{\pi}$, $\boldsymbol{\xi}$, $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\psi}}$, $\boldsymbol{\zeta}$, $\boldsymbol{\rho}$, and $\boldsymbol{\eta}$ as follows:

$$\begin{aligned}&\boldsymbol{\pi} \sim \text{Dirichlet}\big(\boldsymbol{a}^{\pi}\big),\ \ \boldsymbol{\xi}_s \sim \text{Dirichlet}\big(\boldsymbol{a}^{\xi}_s\big),\\ &\hat{\boldsymbol{\phi}}_{\hat{s}} \sim \text{Dirichlet}\big(\boldsymbol{a}^{\hat{\phi}}_{\hat{s}}\big),\ \ \hat{\boldsymbol{\psi}}_{\hat{s}\deg(p;s)} \sim \text{Dirichlet}\big(\boldsymbol{a}^{\hat{\psi}}_{\hat{s}\deg(p;s)}\big),\\ &\boldsymbol{\zeta}_r \sim \text{Dirichlet}\big(\boldsymbol{a}^{\zeta}_r\big),\\ &\boldsymbol{\rho} \sim \text{Dirichlet}\big(\boldsymbol{a}^{\rho}\big),\ \ \boldsymbol{\eta} \sim \text{Dirichlet}\big(\boldsymbol{a}^{\eta}\big),\end{aligned} \tag{12}$$

where $\boldsymbol{a}^{\pi}\in\mathbb{R}^{26}_{+}$, $\boldsymbol{a}^{\xi}_s\in\mathbb{R}^{26}_{+}$, $\boldsymbol{a}^{\hat{\phi}}_{\hat{s}}\in\mathbb{R}^{12}_{+}$, $\boldsymbol{a}^{\hat{\psi}}_{\hat{s}\deg(p;s)}\in\mathbb{R}^{12}_{+}$, $\boldsymbol{a}^{\zeta}_r\in\mathbb{R}^{16}_{+}$, $\boldsymbol{a}^{\rho}\in\mathbb{R}^{2G+1}_{+}$, and $\boldsymbol{a}^{\eta}\in\mathbb{R}^{D}_{+}$ are hyperparameters. The probability distribution over the 12 pitch classes under a scale is estimated using the priors on the initial and transitional probabilities of those classes. As illustrated in Fig. 5, we set the hyperparameters $\boldsymbol{a}^{\hat{\phi}}_{\hat{s}}$ and $\boldsymbol{a}^{\hat{\psi}}_{\hat{s}\deg(p;s)}$ so that the probability distributions represent the diatonic scales, respectively. Since the Cauchy distribution does not have a conjugate prior, we put a Gamma prior on λ as

$$\lambda \sim \text{Gamma}\big(a^{\lambda}_0, a^{\lambda}_1\big), \tag{13}$$

where a^{λ}_0 and a^{λ}_1 are shape and rate hyperparameters.

3.5 Bayesian Inference

Given an F0 trajectory $\boldsymbol{X}$, we aim to calculate the posterior distribution $p(\boldsymbol{Q}, \boldsymbol{S}, \boldsymbol{\Theta}|\boldsymbol{X})$, where $\boldsymbol{Q} = \{\boldsymbol{P}, \boldsymbol{L}, \boldsymbol{G}, \boldsymbol{D}\}$ (latent variables) and $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \boldsymbol{\xi}, \hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\psi}}, \boldsymbol{\zeta}, \boldsymbol{\rho}, \boldsymbol{\eta}\}$ (model parameters). Since this calculation is analytically intractable, we use Markov chain Monte Carlo (MCMC) methods. To get samples of the latent variables $\boldsymbol{S}$ and $\boldsymbol{Q}$, forward filtering-backward sampling algorithms are used. To get samples of $\boldsymbol{\Theta}$ except for λ, a set of parameters with conjugate priors, a Gibbs sampling algorithm is used. Since there is no conjugate prior for the parameter λ, we use the Metropolis-Hastings (MH) algorithm. Since $\boldsymbol{S}$ and $\boldsymbol{Q}$ share the sequence of notes $\boldsymbol{Z}$ and are mutually dependent, each variable is updated as follows:

1. Initialize notes $\boldsymbol{Z}$ with a majority-vote method.
2. Update the sequence of scales $\boldsymbol{S}$ based on given $\boldsymbol{Z}$.
3. Update $\boldsymbol{Q}$ based on given $\boldsymbol{S}$.
4. Update the model parameters $\boldsymbol{\Theta}$.
5. Return to 2.

3.5.1 Inferring Latent Variables $\boldsymbol{S}$

Given the sequence of notes $\boldsymbol{Z}$, each s_m is sampled in accordance with the probability given by

$$\beta^{S}_{s_m} = p(s_m|s_{m+1:M}, \boldsymbol{Z}), \tag{14}$$

where $s_{m+1:M}$ represents $s_{m+1}, \ldots, s_M$. The calculation of Eq. (14) and sampling of scales $\boldsymbol{S}$ are performed by the forward filtering-backward sampling method.

In forward filtering, we recursively calculate the probability $\alpha^{S}_{s_m}$ as follows:

$$\alpha^{S}_{s_0} = p(p_0, s_0) = p(p_0|s_0)p(s_0) = \phi_{s_0p_0}\pi_{s_0}, \tag{15}$$

$$\begin{aligned}\alpha^{S}_{s_m} &= p(p_{0:j_{m+1}-1}, s_m)\\ &= \prod_{j=j_m}^{j_{m+1}-1} \psi_{s_mp_{j-1}p_j} \sum_{s_{m-1}} \xi_{s_{m-1}s_m}\alpha^{S}_{s_{m-1}},\end{aligned} \tag{16}$$

where j_m is the index of the first note whose onset belongs to the m-th bar. j_m can be calculated from given note values $\boldsymbol{L}$.

In backward sampling, Eq. (14) is calculated by using the values calculated in forward filtering, and scales are sampled recursively as follows:

$$\beta^{S}_{s_M} = p(s_M|\boldsymbol{Z}) \propto \alpha^{S}_{s_M}, \tag{17}$$

$$\beta^{S}_{s_m} = p(s_m|s_{m+1:M}, \boldsymbol{Z}) \propto \alpha^{S}_{s_m}\xi_{s_ms_{m+1}}. \tag{18}$$

3.5.2 Inferring Latent Variables $\boldsymbol{Q}$

The latent variables $\boldsymbol{Q}$ can be estimated in a way similar to that in which the latent variables $\boldsymbol{S}$ are inferred. In forward filtering, we recursively calculate the probability $\alpha^{Q}_{p_nl_n,g_nd_n}$ as follows:

$$\alpha^{Q}_{p_0l_0g_0d_0} = p(p_0|\boldsymbol{S}) = \phi_{y_0p_0}, \tag{19}$$

$$\begin{aligned}\alpha^{Q}_{p_nl_ng_nd_n} &= p(x_{1:\tau_n-1}, p_n, l_n, g_n, d_n|\boldsymbol{S})\\ &= \begin{cases} 0 & (l_n > n)\\ \rho_{g_n}\eta_{d_n}\zeta_{r_0r_n} & \\ \quad \cdot \sum_{p_0} \psi_{s_1p_0p_n} e_{p_0p_nl_n0g_nd_n}\alpha^{Q}_{p_0l_0g_0d_0} & (l_n = n)\\ \displaystyle\sum_{p_{n'},g_{n'}} \sum_{l_{n'}}^{\min(n',L)} \sum_{d_{n'}} \rho_{g_n}\eta_{d_n}\zeta_{r_{n'}r_n}\psi_{s_{m(n')}p_{n'}p_n} & \\ \quad \cdot e_{p_{n'}p_nl_ng_{n'}g_nd_n}\alpha^{Q}_{p_{n'}l_{n'}g_{n'}d_{n'}} & (l_n < n)\end{cases},\end{aligned} \tag{20}$$

where $\tau_n = u_n + g_n$, $n' = n - l_n$, and $m(n')$ is the index of the bar that the n'-th tatum belongs to. p_n, l_n, g_n, and d_n are the variables of forward messages that correspond to the note whose offset position is at the n-th tatum u_n. Note that these variables are different from j-indexed variables p_j, l_j, g_j, and d_j. Since the onset and offset times of the note $z_n = (p_n, l_n)$ are respectively the $(n{-}l_n)$-th tatum and the n-th tatum, the probability $p(l_n)$ which appears in the recursive calculation of Eq. (20) is replaced by $p(r_n|r_{n-l_n})$.

In backward sampling, the posterior distribution of the latent variables is calculated by using the values calculated in forward filtering, and notes and temporal deviations are sampled recursively as follows:

$$\beta_{p_Nl_Ng_Nd_N} = p(p_N, l_N, g_N, d_N|\boldsymbol{X}, \boldsymbol{S}) \propto \alpha^{Q}_{p_Nl_Ng_Nd_N},$$

$$\begin{aligned}&\beta_{p_{n'}l_{n'}g_{n'}d_{n'}}\\ &= p(p_{n'}, l_{n'}, g_{n'}, d_{n'}|p_{n:N}, l_{n:N}, g_{n:N}, d_{n:N}, \boldsymbol{X})\\ &\propto \begin{cases} 0 & (l_n > n)\\ e_{p_{n'}p_nl_ng_{n'}g_nd_n}\psi_{s_{m(n')}p_{n'}p_n} & \\ \quad \cdot \zeta_{r_{n'}r_n}\rho_{g_n}\eta_{d_n}\alpha^{Q}_{p_{n'}l_{n'}g_{n'}d_{n'}} & (l_n \leq n)\end{cases}.\end{aligned} \tag{21}$$

3.5.3 Learning Model Parameters $\boldsymbol{\Theta}$

The posterior distributions of the model parameters with the prior distributions are calculated using $\boldsymbol{S}$ and $\boldsymbol{Q}$ obtained in the backward sampling steps, and these parameters are sampled according to the posterior distributions as follows:

$$\boldsymbol{\pi} \sim \text{Dirichlet}\big(\boldsymbol{a}^{\pi} + \boldsymbol{b}^{\pi}\big),\ \ \boldsymbol{\xi}_s \sim \text{Dirichlet}\big(\boldsymbol{a}^{\xi}_s + \boldsymbol{b}^{\xi}_s\big), \tag{22}$$

$$\hat{\boldsymbol{\phi}}_{\hat{s}} \sim \mathrm{Dirichlet}\left(\boldsymbol{a}^{\hat{\phi}}_{\hat{s}} + \boldsymbol{b}^{\hat{\phi}}_{\hat{s}}\right), \tag{23}$$

$$\hat{\boldsymbol{\psi}}_{\hat{s}\deg(p;s)} \sim \mathrm{Dirichlet}\left(\boldsymbol{a}^{\hat{\psi}}_{\hat{s}\deg(p;s)} + \boldsymbol{b}^{\hat{\psi}}_{\hat{s}\deg(p;s)}\right), \tag{24}$$

$$\boldsymbol{\zeta}_r \sim \mathrm{Dirichlet}\left(\boldsymbol{a}^{\zeta}_r + \boldsymbol{b}^{\zeta}_r\right), \tag{25}$$

$$\boldsymbol{\rho} \sim \mathrm{Dirichlet}\left(\boldsymbol{a}^{\rho} + \boldsymbol{b}^{\rho}\right), \quad \boldsymbol{\eta} \sim \mathrm{Dirichlet}\left(\boldsymbol{a}^{\eta} + \boldsymbol{b}^{\eta}\right), \tag{26}$$

where $\boldsymbol{b}^{\pi} \in \mathbb{R}^{26}_{\geq 0}$ is a unit vector whose s_0-th element is 1. $\boldsymbol{b}^{\xi}_{s} \in \mathbb{R}^{26}_{\geq 0}$ is a vector whose s'-th element indicates the number of transitions between adjacent scales s and s' in the sequence of latent variables $\boldsymbol{Y}$. $\boldsymbol{b}^{\rho} \in \mathbb{R}^{2G+1}_{\geq 0}$ is a vector whose g-th element indicates the number of vocal onset deviations of g in sampled $\boldsymbol{Q}$, and $\boldsymbol{b}^{\eta} \in \mathbb{R}^{D}_{\geq 0}$ is a vector whose d-th element represents the number of F0 transitional durations of d in sampled $\boldsymbol{Q}$. $\boldsymbol{b}^{\zeta}_{r} \in \mathbb{R}^{16}_{\geq 0}$ is a vector whose r'-th element represents the number of transitions between adjacent note onset positions r and r' in $\boldsymbol{R} = \{r_j\}_{j=0}^{J}$ that can be calculated from the note values $\boldsymbol{L}$ sampled in backward sampling. Regarding the vector $\boldsymbol{b}^{\hat{\phi}}_{\hat{s}} \in \mathbb{R}^{12}_{\geq 0}$, when the scale of the initial bar and the pitch of the initial note are $s_0 = s$ and $p_0 = p$, the value of the element $b^{\hat{\phi}}_{\hat{s}\deg(p;s)}$ is 1, and the other elements are 0. Regarding the vector $\boldsymbol{b}^{\hat{\psi}}_{\hat{s}\deg(p;s)} \in \mathbb{R}^{12}_{\geq 0}$, the value of $b^{\hat{\psi}}_{\hat{s}\deg(p;s)\deg(p';s)}$ is increased by one when there is a transition from a pitch p to a pitch p' under a scale s in the sampled latent variables.

To apply the MH sampling to the parameter λ, we define a random-walk proposal distribution as follows:

$$q(\lambda^* | \lambda) = \mathrm{Gamma}(\gamma\lambda, \gamma), \tag{27}$$

where λ^* is a proposal, λ is the current sample, and γ is a hyperparameter. The proposal λ^* is accepted as the next sample according to the probability given by

$$A(\lambda^*, \lambda) = \min\left\{\frac{\mathcal{L}(\lambda^*) q(\lambda | \lambda^*)}{\mathcal{L}(\lambda) q(\lambda^* | \lambda)}\right\}, \tag{28}$$

where $L(\lambda)$ is the complete joint likelihood of λ given by

$$\mathcal{L}(\lambda) = \mathrm{Gamma}\left(\lambda | a^{\lambda}_0, a^{\lambda}_1\right) \prod_{j=1}^{J} e_{p_{j-1} p_j l_j g_{j-1} g_j d_j}, \tag{29}$$

$\{p_j, l_j, g_j, d_j\}_{j=0}^{J}$ are the values sampled in the backward sampling. The value of λ is updated by λ^* only when the value of $A(\lambda^*, \lambda)$ is larger than a random number sampled from the uniform distribution $\mathcal{U}(0, 1)$.

3.6 Viterbi Decoding

The sequence of latent variables $\boldsymbol{S}$ and $\boldsymbol{Q}$ are estimated with the Viterbi algorithm with the model parameters that maximize the joint distribution $p(\boldsymbol{X}, \boldsymbol{Q}, \boldsymbol{S}, \boldsymbol{\Theta} | \boldsymbol{\Phi})$ in the learning process. As in the inference of latent variables, we initialize $\boldsymbol{Z}$ by the majority-vote method, $\boldsymbol{S}$ is estimated based on $\boldsymbol{Z}$, and then $\boldsymbol{Q}$ is estimated depending on the $\boldsymbol{S}$ estimated in the previous step.

In the Viterbi decoding on scales $\boldsymbol{S}$, the value ω^{S}_{s} is recursively calculated as follows:

$$\omega^{S}_{s_0} = \ln \phi_{s_0 k_0} + \ln \pi_{s_0}, \tag{30}$$

$$\omega^{S}_{s_m} = \sum_{j=j_m}^{j_{m+1}-1} \ln \psi_{s_m p_{j-1} p_j} + \max_{s_{m-1}} \left\{\ln \xi_{s_{m-1} s_m} + \omega^{S}_{s_{m-1}}\right\}. \tag{31}$$

In the recursive calculation of ω^{S}_{s}, the previous state s_{m-1} that maximizes the value of $\omega^{S}_{s_m}$ is memorized as $c^{S}_{s_m}$, and the scales $\boldsymbol{S}$ are recursively estimated as follows:

$$s_M = \arg\max_{s_M} \alpha^{S}_{s_M}, \quad s_{m-1} = c^{S}_{s_m}. \tag{32}$$

In the Viterbi decoding on variables $\boldsymbol{Q}$, the value ω^{Q}_{plgd} is recursively calculated as follows:

$$\omega^{Q}_{p_0 l_0 g_0 d_0} = \mathrm{w}^{\phi} \ln \phi_{s_0 p_0}, \tag{33}$$

$$\omega^{Q}_{p_n l_n g_n d_n} = \begin{cases} -\inf & (l_n > n) \\ \mathrm{w}^{\rho} \ln \rho_{g_n} + \mathrm{w}^{\eta} \ln \eta_{d_n} + \mathrm{w}^{\zeta} \ln \zeta_{r_n r_0} \\ \quad + \max_{p_0} \Big\{ \mathrm{w}^{\psi} \ln \psi_{s_1 p_0 p_n} \\ \quad + \mathrm{w}^{e} \ln e_{p_0 p_n l_n 0 g_n d_n} + \omega^{Q}_{p_0 l_0 g_0 d_0} \Big\} & (l_n = n) \\ \mathrm{w}^{\rho} \ln \rho_{g_n} + \mathrm{w}^{\eta} \ln \eta_{d_n} + \mathrm{w}^{\zeta} \ln \zeta_{r_n r_{n'}} \\ \quad + \max_{(p_{n'}, l_{n'}, g_{n'}, d_{n'})} \Big\{ \mathrm{w}^{\psi} \ln \psi_{s_{m(n')} p_{n'} p_n} \\ \quad + \mathrm{w}^{e} \ln e_{p_{n'} p_n l_n g_{n'} g_n d_n} + \omega^{Q}_{p_{n'} l_{n'} g_{n'} d_{n'}} \Big\} & (l_n < n) \end{cases}, \tag{34}$$

where w^{ϕ}, w^{ψ}, w^{ρ}, w^{η}, w^{ζ}, and w^{e} are the weight parameters that control the balance between probabilities. In the recursive calculation of ω^{Q}_{plgd}, the previous states $p_{n'}$, $l_{n'}$, $g_{n'}$, and $d_{n'}$ which maximize the value of $\omega^{Q}_{p_n l_n g_n d_n}$ are memorized as $c^{Q}_{p_n l_n g_n d_n}$, and the variables $\boldsymbol{Q}$ are recursively estimated as follows:

$$(p_N, l_N, g_N, d_N) = \arg\max_{p_N, l_N, g_N, d_N} \alpha^{Q}_{p_N l_N g_N d_N}, \tag{35}$$

$$(p_{n'}, l_{n'}, g_{n'}, d_{n'}) = c^{Q}_{p_n l_n g_n d_n}. \tag{36}$$

4. EVALUATION

We report comparative experiments conducted to evaluate the performance of the proposed method in musical note estimation from vocal F0 trajectories.

4.1 Experimental Conditions

Among the 100 pieces of popular music in the RWC music database [5], we used 63 pieces that do not include 32nd notes, triplets, harmonizing parts, and overlaps of adjacent notes, which the proposed method cannot deal with. The input F0 trajectories were obtained from the annotation data [4] or automatically estimated by using the state-of-the-art melody extraction method proposed in [9]. The annotation data contain unvoiced regions and the estimation data do not. The tatum times and onset positions were obtained from the annotation data.

The Bayesian inference and Viterbi decoding were independently conducted for each song. The onset transition probabilities were learned in advance from a corpus of rock music [2] without Bayesian learning. The hyperparameters were $\boldsymbol{a}^{\pi} = \boldsymbol{1}$, $\boldsymbol{a}^{\xi}_{s} = \mathbb{1}$, $\boldsymbol{a}^{\zeta}_{r} = \boldsymbol{1}$, $\boldsymbol{a}^{\rho} = \boldsymbol{a}^{\eta} = a^{\lambda}_0 = a^{\lambda}_1 = \gamma = 1$, where $\mathbb{1}$ and $\boldsymbol{1}$ respectively represent the matrix and vector whose elements are all ones. The elements of $\boldsymbol{a}^{\hat{\phi}}_{\hat{s}}$ and $\boldsymbol{a}^{\hat{\psi}}_{\hat{s}\deg(p;s)}$ corresponding to musical notes on the scale of $\hat{s}$ were 10 and the others were 1. The weight parameters of the Viterbi algorithm were empirically set as $\mathrm{w}^{\phi} = \mathrm{w}^{\psi} = 29.4$, $\mathrm{w}^{\rho} = 2.4$, $\mathrm{w}^{\eta} = 2.9$, $\mathrm{w}^{\zeta} = 48.5$, and $\mathrm{w}^{e} = 3.8$. To obtain musically-consistent sequences of

Model	Input F0s	Tatum level	Note level
Proposed method	Ground-truth	**72.4** ± 1.7	28.1 ± 2.1
	Estimated	68.7 ± 1.3	30.7 ± 1.8
With only rhythms	Ground-truth	71.5 ± 1.6	26.3 ± 2.1
	Estimated	67.7 ± 1.3	29.1 ± 1.8
With only scales	Ground-truth	67.8 ± 1.6	10.6 ± 1.2
	Estimated	65.6 ± 1.2	13.8 ± 1.1
Without scales & rhythms	Ground-truth	67.2 ± 1.5	9.8 ± 1.2
	Estimated	64.6 ± 1.2	12.9 ± 1.1
Majority vote	Ground-truth	54.1 ± 1.5	20.1 ± 1.4
	Estimated	61.0 ± 1.4	22.0 ± 1.5
HMM [16]	Estimated	68.0 ± 1.2	14.8 ± 1.3

Table 1: Average matching rates [%] and their standard errors in tatum and note levels.

musical notes, we put more emphasis on the score model than the F0 model.

For comparison, we tested the majority-vote method as a baseline and the latest conventional method based on a semi-beat-synchronous HMM [16]. Since the conventional method cannot deal with unvoiced regions in a vocal F0 trajectory given as input, we only tested the method for the estimation data. To evaluate the effectiveness of the score model, we tested four versions of the proposed method; a method that does not consider scales (scale transition probabilities) and rhythms (onset transition probabilities), a method considering only scales, a method considering only rhythms, the full method considering both scales and rhythms. To accelerate the inference, the search range of pitches was limited around the pitches estimated by the majority-vote method.

To evaluate the performance of each method, we calculated tatum-level and note-level matching rates by comparing the estimated sequences of musical notes with the ground-truth data. The tatum-level matching rate is the rate of the number of tatum units whose pitches were estimated correctly to the total number of tatum units whose pitches exist in the ground-truth scores. The note-level matching rate is the rate of the number of musical notes whose pitches, onsets, and offsets were estimated correctly to the total number of musical notes in the ground-truth scores. If adjacent notes in the ground-truth scores have the same pitch or are connected by a tie, those notes were regarded as a single note. Since the compared method [16] outputs a pitch in a 16th-note-wise manner, a sequence of the same pitches was regarded as a single note.

4.2 Experimental Results

The experimental results are shown in Table 1[1] . The proposed method outperformed the majority-vote method and the conventional method in terms of both measures. Comparing the tatum-level matching rates obtained by the four versions of the proposed method, we confirmed that the score model improved the performance of musical note estimation. The use of the onset transition probabilities

[1] The results of music note estimation by the proposed method are available online: http://sap.ist.i.kyoto-u.ac.jp/members/nishikimi/demo/ismir2017/

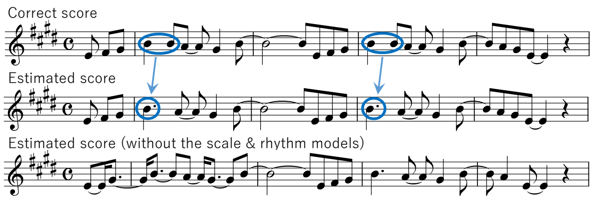


Figure 6: Musical scores estimated from a ground-truth F0 trajectory by the proposed method and its variant without scale and rhythm constraints.

(rhythm constraints) was found to be more effective than that of the scale transition probabilities (scale constraints). Although the tatum-level matching rate obtained the proposed method (68.7%) was close to that obtained by the conventional method (68.0%), the note-level matching rate obtained the proposed method (30.7%) was better than that obtained by the conventional method (14.8%), This is a remarkable advantage of the proposed HHSMM that can directly represent both the pitches and durations (onsets and offsets) of musical notes on symbolic musical scores, not on continuous-time piano rolls.

Examples of estimated musical scores are illustrated in Fig. 6. The proposed method yielded the almost accurate musical score except that some notes were merged. To correctly recognize two adjacent notes with the same pitch, it is necessary to refer to original singing voices or music audio signals. The score estimated without considering the score model, on the other hand, included a lot of wrong notes that were inconsistent with music theory. This result also shows the effectiveness of using the score model as musical constraints on musical note estimation.

5. CONCLUSION

This paper presented a statistical method for musical note estimation from a vocal F0 trajectory. Our method is based on an HHSMM that combines a score model (HMM) representing the generative process of a musical score from musical scales with an F0 model (HSMM) representing the generative process of a vocal F0 trajectory with time-frequency deviation from the musical score. We confirmed that the proposed method can yield more musically-consistent sequences of musical notes.

One of the most interesting directions of this research is to use the proposed model as a musically-meaningful prior distribution on a vocal F0 trajectory in vocal F0 estimation for music audio signals. We plan to integrate the proposed "language" model that generates an F0 trajectory from a musical score with an acoustic model that generates a spectrogram from the F0 trajectory in a hierarchical Bayesian manner. This enables us to jointly learn the vocal F0 trajectory and musical score from music audio signals. Joint estimation of beat times and F0s is worth investigating to overcome the problem of estimation-error accumulation in the cascaded estimation approach.

Acknowledgement: This study was partially supported by JSPS KAKENHI Grant Numbers 26700020, 16H01744, and 16J05486 and JST ACCEL No. JPMJAC1602.

6. REFERENCES

[1] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[2] T. De Clercq and D. Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, 2011.

[3] J.-L. Durrieu, G. Richard, B. David, and C. Févotte. Source/filter model for unsupervised main melody extraction from polyphonic audio signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):564–575, 2010.

[4] M. Goto. Aist annotation for the RWC music database. In *The 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 359–360, 2006.

[5] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *The 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 287–288, 2002.

[6] M. Goto, K. Yoshii, H. Fujihara, M. Mauch, and T Nakano. Songle: A web service for active music listening improved by user contributions. In *Proc. of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 311–316, 2011.

[7] Dik J. Hermes. Measurement of pitch by subharmonic summation. *The journal of the acoustical society of America*, 83(1):257–264, 1988.

[8] P.-S. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012)*, pages 57–60, 2012.

[9] Y. Ikemiya, K. Yoshii, and K. Itoyama. Singing voice analysis and editing based on mutually dependent F0 estimation and source separation. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, pages 574–578, 2015.

[10] Y. E. Kim and B. Whitman. Singer identification in popular music recordings using voice coding features. In *3rd International Conference on Music Information Retrieval (ISMIR 2002)*, volume 13, page 17, 2002.

[11] A. Laaksonen. Automatic melody transcription based on chord transcription. In *Proc. of the 15th International Society for Music Information Retrieval (ISMIR 2014)*, pages 119–124, 2014.

[12] Y. Li and D. Wang. Separation of singing voice from music accompaniment for monaural recordings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(4):1475–1487, 2007.

[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. of the 1st International Conference on Technologies for Music Notation and Representation (TENOR 2015)*, pages 23–30, 2015.

[14] 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 2014)*, pages 659–663, 2014.

[15] 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)*, 23(2):252–263, 2015.

[16] 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. of the 17th International Society for Music Information Retrieval Conference (ISMIR 2026)*, pages 461–467, 2016.

[17] R. P. Paiva, T. Mendes, and A. Cardoso. On the detection of melody notes in polyphonic audio. In *6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 175–182, 2005.

[18] C. Raphael. A graphical model for recognizing sung melodies. In *6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 658–663, 2005.

[19] M. Ryynänen, T. Virtanen, J. Paulus, and A. Klapuri. Accompaniment separation and karaoke application based on automatic melody transcription. In *2008 IEEE International Conference on Multimedia and Expo*, pages 1417–1420, 2008.

[20] M. P. Ryynänen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[21] J. Salamon and E. Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[22] W.-H. Tsai and H.-M. Wang. Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1):330–341, 2006.

[23] L. Yang, A. Maezawa, J. B. L. Smith, and E. Chew. Probabilistic transcription of sung melody using a pitch dynamic model. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, pages 301–305, 2017.

SCORE-INFORMED SYLLABLE SEGMENTATION FOR A CAPPELLA SINGING VOICE WITH CONVOLUTIONAL NEURAL NETWORKS

Jordi Pons*, Rong Gong* and Xavier Serra
Music Technology Group, Universitat Pompeu Fabra, Barcelona
jordi.pons@upf.edu, rong.gong@upf.edu, xavier.serra@upf.edu
* contributed equally

ABSTRACT

This paper introduces a new score-informed method for the segmentation of jingju a cappella singing phrase into syllables. The proposed method estimates the most likely sequence of syllable boundaries given the estimated syllable onset detection function (ODF) and its score. Throughout the paper, we first examine the jingju syllables structure and propose a definition of the term "syllable onset". Then, we identify which are the challenges that jingju a cappella singing poses. Further, we investigate how to improve the syllable ODF estimation with convolutional neural networks (CNNs). We propose a novel CNN architecture that allows to efficiently capture different time-frequency scales for estimating syllable onsets. Besides, we propose using a score-informed Viterbi algorithm – instead of thresholding the onset function–, because the available musical knowledge we have (the score) can be used to inform the Viterbi algorithm to overcome the identified challenges. The proposed method outperforms the state-of-the-art in syllable segmentation for jingju a cappella singing. We further provide an analysis of the segmentation errors which points possible research directions.

1. INTRODUCTION

The ultimate goal of our research project is to automatically evaluate the jingju a cappella singing of a student in the scenario of jingju singing education – see Figure 1. Jingju, a traditional Chinese performing art form also known as Peking or Beijing opera, is extremely demanding in the clear pronunciation and accurate intonation for each syllabic or phonetic singing unit. To this end, during the initial learning stages, students are required to completely imitate tutor's singing. Therefore, the automatic jingju singing evaluation tool we envision is based on this training principle and measures the intonation and pronunciation similarities between the student's and the tutor's singings. Before measuring the similarities, the singing phrase should be automatically segmented into syllabic or phonetic units in order to capture the temporal details.
In this paper we tackle the problem of score-informed automatic syllable segmentation for a cappella singing (bold rectangle in Figure 1). Jingju music scores, which contain the duration information for each singing syllable, will be a helpful hint for the segmentation method.

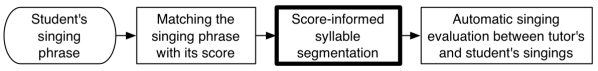

Student's singing phrase
Matching the singing phrase with its score
Score-informed syllable segmentation
Automatic singing evaluation between tutor's and student's singings

Figure 1. Framework of the entire research project. The module with bold border is addressed in this paper.

The syllable segmentation task consists of determining the time positions of the syllable boundaries – onset and offset. In this research, we consider the onset of the subsequent syllable to be the offset of the current one. Therefore, we treat the segmentation task as an onset detection problem.

Most segmentation methods rely on first estimating an ODF. For example, Klapuri [7] utilizes a band-wise processing principle inspired by psychoacoustics. He divides the audio signal into 21 non-overlapping bands, then detects onset components in each band ODF and finally combines them to yield onsets. Obin *et al.* [10] obtain a syllable ODF by fusing mel-frequency intensity profiles and voicing profiles. One shortcoming of these band-wise methods has been already pointed out by Klapuri [7]: *"they are unable to deal with strong amplitude modulations"* – what is very common in singing voice recordings.

On the other hand, some methods are based on features and supervised learning. For example, Toh *et al.* [17] use two Gaussian Mixture Models (GMMs) to classify singing audio frames into onset or non-onset classes. Three timbral features (MFCCs, LPCCs and ERB-bands) are chosen as input. Their results show that this supervised method is superior to many band-wise ODF-based methods.

Neural networks have also been successfully explored for the task of musical onset detection. Eyben *et al.* [4] trained a bidirectional long-short term memory neural network on mel-scale magnitude spectrograms to estimate an ODF. Schlüter *et al.* [16] proposed to use CNNs to estimate the ODF, which defines the current state-of-the-art for onset detection. The advantage of applying deep learning methods in musical onset detection is that one does not

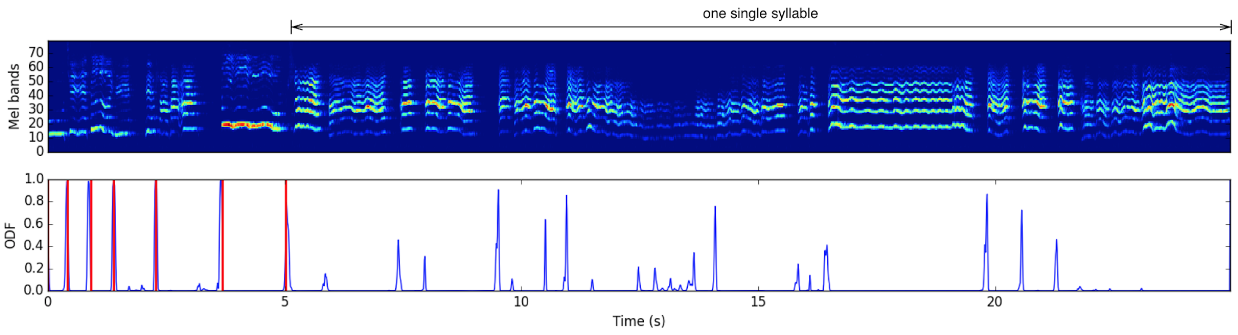


Figure 2. A *dan* role-type singing phrase with its last syllable prolongated. **Top:** log-mel bands spectrogram. **Bottom:** syllable ODF estimated with Schlüter's method [16] (blue), and ground truth syllable onsets (red).

need to design rules or handcraft features to capture the relevant facets for detecting onset/non-onset frames – which are very difficult to design. Besides, if context (more than one frame) is input into the network, spectro-temporal features can also be learned – what might be useful to learn slow-transient features defining some onsets.

Thresholding or peak-picking operations are normally executed over the ODF in order to determine the final onsets [4, 7, 16, 17]. However, we will argue that these operations are not suitable for selecting jingju singing syllable onsets. Probabilistic models –which allow incorporating *prior* domain knowledge for decision making– usually result in a performance gain compared to simple thresholding and peak-picking methods. For example, Böck *et al.* [1] tracked the beat/downbeat by using a dynamic bayesian network (DBN) observing a beat ODF estimated by a recurrent neural network (RNN). Or Obin *et al.* [10] applied a segmental Viterbi algorithm over a syllable ODF to detect speech syllable onsets. These two approaches are closely related to ours: former one relates with our work because it uses deep learning to estimate the ODF, and latter one because we also consider using the Viterbi algorithm for estimating the onset sequence.

This paper introduces a new score-informed method for the segmentation of jingju singing into syllables. We first define what a "syllable" and "syllable onset" is in the context of jingju music. By doing so, in section 2 we introduce the challenges we aim to address. The audio dataset we use is described in section 3. Section 4 explains the CNN architecture used for estimating the syllable ODF, and the Viterbi decoding algorithm that exploits the prior syllable duration information extracted from the score. Evaluation and error analysis are conducted in section 5, and section 6 concludes this work.

2. BACKGROUND AND CHALLENGES

Jingju singing is the most precise articulated rendition of the spoken Mandarin language. Although certain special pronunciations in jingju theatrical language differ from their normal Mandarin pronunciations –due to: firstly, the adoption of certain regional dialects; and secondly, the ease or variety in pronunciation and projection of sound– the mono-syllabic pronouncing structure of the standard Mandarin doesn't change [18].

A syllable/character of jingju singing is composed of three distinct parts in most of the cases: the "head" (*tou*), the "belly" (*fu*) and the "tail" (*wei*). The "head" consists of an initial consonant or semi-vowel – and a medial vowel, if the syllable includes one. The "head" is not normally prolonged in its pronunciation except when there is a medial vowel. The "belly" follows the "head" and consists of the central vowel and it is normally prolonged. The "belly" is the most sonorous part of a jingju singing syllable and can be analogous to the nuclei of a speech syllable. The "tail" is composed of the terminal vowel or consonant [18]. However, there are syllables in jingju singing where "head" is absent – only "belly" is a necessary element. To avoid ambiguity, we define the syllable onset as *"the start of the initial consonant if the syllable includes one or the start of the central vowel otherwise"*. This definition is slightly different from that agreed by most of the phonological theories [5]: "*any consonants that precede the nuclear element (the vowel)*". It is also worth stressing that our notion of singing syllable onset is different from the singing onset defined in Toh *et al.*'s paper [17]: "*the start of a new human-perceived note, taking into account contextual cues*", of which the latter emphasizes on the intonational aspect instead of the phonological aspect of the singing voice.

2.1 Challenges

Figure 2 shows an example of a *dan* role-type singing phrase in which the last syllable lasts approximately 20s. This singing method is more common in *dan* singing than in *laosheng* singing. However, both role-types use it as a way of improving artistic expression and showing off their singing skills. These skills include breathing techniques, intonational techniques and dynamic control techniques, among others. The syllable ODF (blue curve in Figure 2) is generated using a CNN model based on Schlüter *et al.*'s work [16], which is considered the state-of-the-art. This CNN model is trained with the jingju dataset presented in section 3. The resulting syllable ODF is quite robust to pitch variations and continuous dynamic change since no prominent peaks are observed in these regions. However, many peaks can be found in the start and end positions of each articulation throughout this long syllable ($\approx$ 20s). Note that thresholding or peak-picking would choose these peaks (false syllable onsets) since they have similar amplitude level than real onsets. For that reason, we propose using a score-informed decoding method as an alternative to naive thresholding.

Stealing breath (*tou qi*) is one of the primary methods of taking the breath in jingju singing. Breath is stolen when a sound is too long to be delivered in one breath – and no vocal pauses are desired [18]. However, this is not the only technique which can lead to pauses within a syllable. Another singing technique (*zu yin* – literal translation: block sound), provokes also pauses without occurring exhalation or inhalation. This kind of pause can be very short in duration and can be easily found in jingju singing syllables, see Figure 3 for example. These silences can be another source of false positives if thresholding or peak-picking is used. However, these can also be avoided by using a score-informed decoding method.

A priori syllable duration information is often easy to obtain from the score and this is an advantage which we exploit to avoid previously described issues. The repertoire of jingju includes around 1400 plays [18], and most used teaching pieces are transcribed into scores. We use the score to guide the Viterbi decoding throughout the onset detection process.

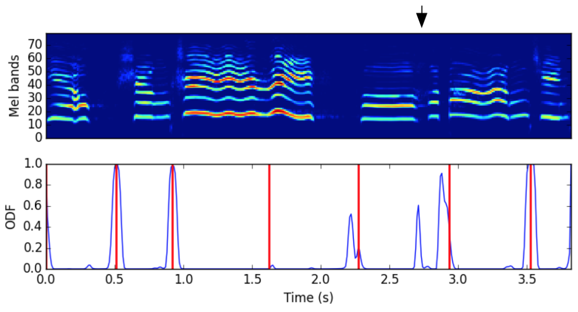


Figure 3. Another *dan* role-type singing phrase. A *zu yin* (block sound) is located by the arrow within a normal length syllable. Schlüter's syllable ODF (blue curve). Ground truth syllable onset positions (red vertical lines).

3. DATASET

The *a cappella* singing audio dataset [1] used for this study focuses on two most important jingju role-types [14]: *dan* (female) and *laosheng* (old man). It contains 39 interpretations of 31 unique arias sung by 11 jingju singers. Audio is sampled at 44.1 kHz and is is pre-segmented into phrases. The syllable onset ground truth is manually annotated in Praat [2] – with 291 phrases and 2641 syllables (including padding written-characters [18]). Two Mandarin native speakers and a jingju musicologist have been devoted to this annotation work. Syllable durations are manually transcribed from music scores considering as unit duration the quarter note. The whole dataset is randomly split into training, validation and test sets (60%, 20% and 20%, respectively). The percentage of presence of each role-type in a split is kept constant throughout sets.

In order to highlight the differences between jingju music and popular western music, we compare the vowel's duration between Kruspe's dataset [8] (a cappella singing of commercial pop songs) and ours. In Kruspe's dataset, the standard deviation duration of voiced phonemes is of 0.31s, whereas this duration is more than doubled in our dataset: 0.75s. Since voiced phonemes are the main component of a syllable, it thus becomes clear that jingju singing syllable durations show huge variations. Therefore, it is impossible to model syllable durations with a single distribution. For that reason the proposed model must be able to handle different syllable lengths depending on the case. To this end, syllable durations extracted from scores will be a valuable information.

4. APPROACH

A score-informed syllable boundary detection approach is explained in this section. We first propose some improvements to the current state-of-the-art CNN onset detection model proposed by Schlüter *et al.* The syllable ODF output from the CNN serves as observation probability for the syllable boundary decoding process. Then, an *a priori* syllable duration model based on the score is proposed. This model guides the Viterbi algorithm by informing it in which time-positions is likely to occur a syllable boundary. Therefore, the syllable boundaries sequence is decoded by taking advantage of a score-informed Viterbi algorithm.

4.1 CNN syllable onset detection function

Studied CNNs are inspired by Schlüter *et al.*'s [16] and Pons *et al.*'s work [11, 12]. Schlüter *et al.* have shown that CNNs fed with spectrograms can achieve state-of-the-art performance for the onset detection task [16]. On the other hand, Pons *et al.* [11] have recently proposed a novel design strategy for spectrograms-based CNNs. They propose using different filter shapes in the first layer so that local stationarities in spectrograms (present at different time/frequency-scales) can be efficiently captured.

Explored models are based on Schlüter *et al.*'s architecture [16], which consists of two convolutional layers, and a dense layer of 256 units connected to an output softmax layer – with two output units standing for onset and non-onset. First CNN layer uses 20x 3×7 filters [2] and is followed by a 3×1 max pool layer. Second CNN layer has 20x 3×3 filters and is followed by a 3×1 max-pool layer. Input is set to be a log-mel spectrogram [3] of size 80×21 – note that the network takes a decision for every frame given its context: ±10 frames, 21 frames in total.

[1] https://goo.gl/y0P7BL

[2] First number denote the number of filters (*ie.* 20x). Second and third respectively denote the frequency and temporal size of the filter (*ie.* 3×7).

[3] Original Schlüter *et al.*'s model inputs three channels with different resolution spectrograms to the network. However, preliminary results showed that a single channel was performing better than three.

Proposed architectures follow Pons *et al.* [11,12] design strategy and several filter shapes in the first layer are used. This approach allows to efficiently capture different time-frequency scales, what might be interesting for the task at hand because onsets can exhibit different time-frequency patterns. For example, Figure 4 depicts two onsets: *middle* onset is expressed as an abrupt time-frequency change corresponding to an onset starting with a consonant, and *right* onset is expressed as a gradual timbral change that corresponds to an onset starting with a central vowel. Note that these examples are representative of the proposed definition of syllable onset. Moreover, using different filter shapes in the first layer promotes a much richer representation out of the first layer. In the following, several CNNs are designed to efficiently capture the relevant time-frequency contexts and scales for onset detection.

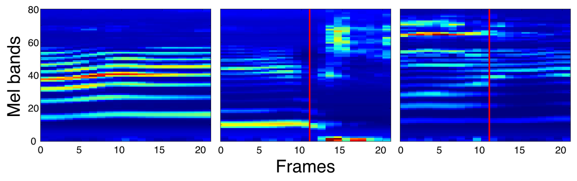


Figure 4. Log-mel spectrograms (80 bands, 21 frames). *left*: Non-onset spectrogram. *middle* and *right*: Onset spectrograms. Red line denotes the onset frame.

Schlüter *et al.*'s architecture and proposed ones only differ in the way the first convolutional layer and its following max-pool layer are set up. Input and remaining layers are kept intact – unless it is explicitly stated:

$\rightarrow$ Temporal architecture. Note that the small filters proposed by Schlüter *et al.* might have difficulties on learning the longer time-scales required to model the gradual changes that some onsets express – see Figure 4, *right*. Several filter shapes are designed to efficiently capture different and longer time-scales [11] than Schlüter *et al.*:

· 12x 1×7, 6x 3×7 and 3x 5×7

· 12x 1×12, 6x 3×12 and 3x 5×12

These CNNs are followed by a 3×5 max-pool layer.

$\rightarrow$ Timbral architecture. Since our syllable onset definition considers phonological aspects, we consider the timbre to be an important feature for our task. Filter shapes are designed to learn timbral representations, note that these filters span throughout the frequency domain [12]:

· 12x 50×1, 6x 50×5 and 3x 50×10

· 12x 70×1, 6x 70×5 and 3x 70×10

These CNNs are followed by a 5×3 max-pool layer.

We also explore combining temporal and timbral architectures. We propose a late-fusion approach where we multiply the estimated output probabilities from temporal and timbral (independently trained) architectures [4].

In addition, we also explore increasing the number of filters in the second layer from 20x to 32x.

For concatenating several feature maps resulting of CNNs with different filter shapes, it is required to use zero padding. We apply *same* padding in the first CNN layer, so that all resulting feature maps have the same length and these can be concatenated. STFT was performed using a 25ms window (2048 samples with zero-padding) with a hop size of 10ms. The 80 log-mel bands energies are calculated on frequencies between 0Hz and 11000Hz and spectrograms are standardized to have zero mean and unit variance. We use L2 weight decay regularization, ELU activation functions [3] and 30% dropout for each layer. The model parameters are learned with mini-batch training (batch size 128) using the ADAM update rule [6] and early stopping – if validation loss (categorical cross-entropy) was not decreasing after 10 epochs. In order to allow a fair comparison between Schlüter's and Pons' architectures, the original Schlüter architecture is modified to meet above described hyper-parameters.

Finally, syllable ODFs estimated with CNNs are smoothed by convolving those with a 5 frames Hanning window [16] – since estimated syllable ODFs are typically very spiky.

[4] Using temporal and timbral filters in first layer yield to worse results.

4.2 A priori duration model

The *a priori* duration model is shaped with a Gaussian function $\mathcal{N}(x;\mu_l,\sigma_l^2)$ whose mean μ_l represents the l-th relative syllable duration – according to the score. Its standard deviation σ_l is proportional to μ_l: $\sigma_l = \gamma\mu_l$ and γ is heuristically set to 0.35.

$$\mathcal{N}(x;\mu_l,\sigma_l^2) = \frac{1}{\sqrt{2\pi}\sigma_l}\exp\left(-\frac{(x-\mu_l)^2}{2\sigma_l^2}\right). \quad (1)$$

Figure 5 provides an intuitive example of how the *a priori* duration model works. Observe that it provides the prior likelihood of an onset to occur according to the duration in the score.

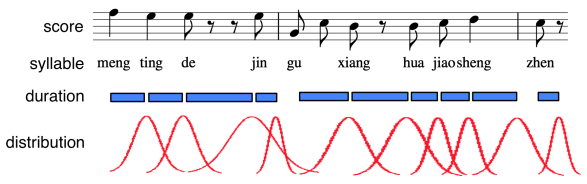


Figure 5. *A priori* relative duration distributions (bottom) of the syllables of a singing phrase.

The relative duration of each note is measured considering a quarter note length as a unit, so an eighth note has a duration of 0.5. We only keep the relative duration and discard the tempo information of the score. By normalizing the summation of the notes' relative durations to the incoming audio recording's duration, we obtain the absolute notes' durations. Then, a sequence of syllable absolute durations $M = \mu_1\mu_2\cdots\mu_L$ is deduced by summing the notes' absolute durations corresponded with each syllable, where L is the total syllable number of the score. The a priori duration model distributions will be incorporated into the Viterbi algorithm as state transition probabilities to inform the algorithm where syllable boundary is likely to occur.

4.3 Decoding of the syllable boundaries

To decode the syllable boundaries, we construct a hidden markov model characterized by the following:

1. The hidden state space is a set of N candidate onset positions $S_1, S_2, \cdots, S_N$ discretized by the hop size, where S_N is the offset position of the last syllable.
2. The state transition probability at the time instant l associated with state changes is defined by *a priori* duration distribution $\mathcal{N}(d_{ij}; \mu_l, \sigma_l^2)$, where d_{ij} is the time distance between states S_i and S_j ($j > i$). The length of the decoded state sequence is equal to the total syllable number L written in the score.
3. The observation probability for the state S_j is represented by its corresponding value in the syllable ODF p, which is denoted as p_j.

The goal is to find the best onset position state sequence $Q = q_1 q_2 \cdots q_L$ for a given *a priori* duration sequence M, where q_i denotes the onset of the $i+1$th decoding syllable. q_0 and q_L are fixed as S_1 and S_N as we expect that the onset of the first syllable to be located at the beginning of the incoming audio and the offset of the last syllable is located at the ending of the audio. One can fulfill this assumption by truncating the silences at the beginning and at the end of the incoming audio. According to the logarithmic form of Viterbi algorithm [13], we define:

$$\delta_l(i) = \max_{q_1, q_2, \cdots, q_l} \log P[q_1 q_2 \cdots q_l,\ \mu_1 \mu_2 \cdots \mu_l]$$

with the initial step as follows:

$$\delta_1(i) = \log(\mathcal{N}(d_{1i}; \mu_1, \sigma_1^2)) + \log(p_i)$$
$$\psi_1(i) = S_1$$

with the following recursive step:

$$\delta_l(j) = \max_{1 \leqslant i < j} [\delta_{l-1}(i) + \log(\mathcal{N}(d_{ij}; \mu_l, \sigma_l^2))] + \log(p_j)$$
$$\psi_l(j) = \arg\max_{1 \leqslant i < j} [\delta_{l-1}(i) + \log(\mathcal{N}(d_{ij}; \mu_l, \sigma_l^2))]$$

and the following termination step:

$$\log P^* = \max_{1 \leqslant i < N} [\delta_{L-1}(i) + \log(\mathcal{N}(d_{iN}; \mu_L, \sigma_L^2))]$$
$$q_L^* = \arg\max_{1 \leqslant i < N} [\delta_{L-1}(i) + \log(\mathcal{N}(d_{iN}; \mu_L, \sigma_L^2))]$$

Finally, the best offset position state sequence Q is obtained by the backtracking step. An example of the best boundary position decoding path is showed in figure 6.

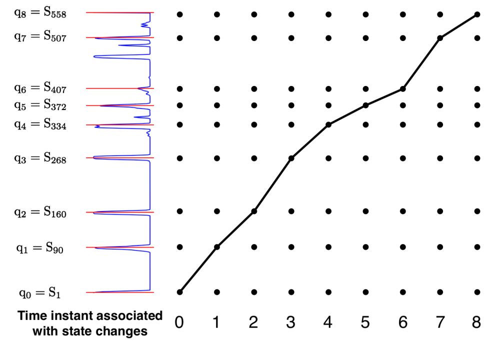


Figure 6. Illustration of an example of the best boundary position decoding path (black) with $N = 558$ and $L = 8$. The intermediate states are omitted in the vertical direction for a clear visualization. Blue curve: syllable ODF; red horizontal lines: decoded boundary positions.

5. EXPERIMENTS AND RESULTS

5.1 Performance metrics

The syllable segmentation task consists on determining the time positions of syllable boundaries. The proposed evaluation consists in comparing the detected syllable onsets/offsets to their reference ones. We report F-measure results in this paper. The definition of a correct segmented syllable is borrowed from the note transcription literature [9]. For syllable onsets, we choose an evaluation tolerance of $\pm\tau$ ms. For offsets, we choose an evaluation tolerance of either *(a)* $\pm 20\%$ of the syllable's duration annotation, or *(b)* $\pm\tau$ ms – whichever is larger. If both the onset and the offset of a syllable lie within the tolerance of their annotated counterparts and the syllable is correctly labeled, we consider that it's correctly segmented. We report the results for a tolerance of $\tau = 0.05$ (seconds).

5.2 Results and discussion

Two state-of-the-art methods are set as baselines: Obin *et al.* [10] as a traditional approach, and Schlüter *et al.* [16] as a deep learning method – both already introduced. We explore Pons *et al.*'s CNNs design strategy [11, 12] as a way to improve our results. Code is available online [5]. All syllable ODFs are decoded by using the same approach (described in section 4.3). F-measure results are reported in Table 5.2. The evaluation can not be performed by using peak-picking because the syllabic label can not be attached. However, in section 2 we already discussed the potential problems of peak-picking which are explicitly addressed with the proposed score-informed method.

Results show that proposed architectures improve the state-of-the-art in all cases. These results support the idea that using different filter shapes in the first CNN layer is beneficial. Moreover, observe that #params is reduced at least to the half – interpret #params (number of parameters of the CNN) as a measure of the representational capacity of the network. This denotes how efficient can be CNNs if these are designed to capture the relevant features for the task at hand. Also note that if the #params of a network is reduced, the over-fitting risk is reduced as well.

We also observe that late-fusing the predictions helps to improve the results. The interesting idea behind this approach is that individual networks still need to be able to solve the task by their means – and we propose fusing

[5] https://github.com/ronggong/jingjuSyllabicSegmentaion/tree/v0.1.0

Syllable ODFs	**#params**	**F-measure (%)**
Late-fusion *(32)*	-	**86.37**
Temporal *(32)*	210,403	83.28
Timbral *(32)*	185,420	84.83
Late-fusion *(20)*	-	84.05
Temporal *(20)*	132,127	83.86
Timbral *(20)*	119,432	82.89
Schlüter *et al.* [16]	535,290	81.93
Obin *et al.* [10]	-	40.61

Table 1. Syllable segmentation results using different methods for estimating the syllable ODF. Numbers in parenthesis indicate the #filters in the second CNN layer.

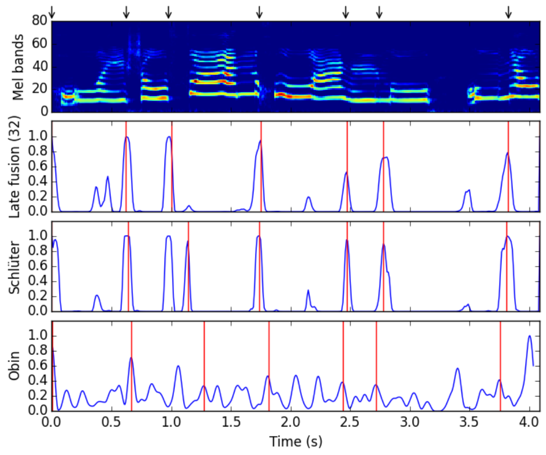


Figure 7. Three syllable ODFs. Red vertical line: decoded syllable onset time positions. Black arrows: ground truth.

models that are tailored towards learning complementary time-frequency scales: temporal and timbral architectures.

We also see a significant performance gap between Obin *et al.* and CNN-based methods. For example in Figure 7 numerous noisy peaks can be observed for Obin *et al.*'s method, what is significantly decreasing its performance. This result denotes the difficulty of designing rules or handcrafting features for estimating the syllable ODF.

5.3 Error analysis

We conduct an error analysis to study the causes of the segmentation errors produced by our best performing method, what points out future research directions for further improvement. We study falsely detected onsets in the test dataset by comparatively observing the detected syllable onset positions, the ground truth positions, the spectrogram and the corresponding scores. Errors out of 0.05s tolerance are analyzed and 70 syllable onsets out of 512 are identified as false detections. Errors are then classified according to their causes. Two main error classes are found: score-performance mismatching and ambiguous syllable transitions.

The majority of the errors (71.42%) are caused by performance deviations from the score: syllable insertion, syllable deletion or performing different syllable durations than the score. The proposed syllable boundary decoding algorithm is not able to preserve the correctness when the difference between the score and its performance becomes too large. Furthermore, since the length of the decoded state sequence of the algorithm must be equal to the number of syllables in the score, singing syllable insertions and deletions not expressed in the score can not be handled properly. One possible solution is to incorporate more domain knowledge of jingju singing into the decoding process – such as considering the probability of a padding-character to be inserted/deleted, or the expectation of prolonging the last syllable in a phrase.

The second source of errors include ambiguous syllable transitions (15.61%) – such as transitions from vowel to vowel, from vowel or to semi-vowel, etc. These errors are very difficult to be corrected because no prominent spectral changes can be discerned within such transitions [15]. In future investigations we shall detect "onset regions" rather than "onset time positions" since this kind of syllable transitions usually manifest themselves as gradual spectral changes.

6. CONCLUSIONS

This paper introduces a new score-informed method for the segmentation of jingju singing into syllables. Two main contributions are presented in this paper: *(i)* improvements for estimating the syllable ODF with CNNs, and *(ii)* a method for incorporating score information into Viterbi's algorithm for estimating syllable boundaries.

The improvements to the CNN architecture consisted of using different filter shapes in the first layer and late-fusing the predictions of two models – designed to learn complementary representations (temporal and timbral). By doing so, we increased the expressiveness of the first layer and enabled the networks to efficiently capture different time-frequency scales useful for detecting syllable onsets. Proposed models, with many filter shapes in the first layer, has proven to be more effective than the state-of-the-art model based on a single filter shape in the first layer.

Moreover, we proposed an *a priori* duration model that describes the probability of a syllable boundary given the score. The likelihood of a syllable boundary is shaped with the *a priori* duration model and incorporated into Viterbi's algorithm as state transition probabilities – this being the core of the proposed score-informed Viterbi algorithm.

We validated the proposed method on a jingju a cappella singing dataset, which achieved better performance than the state-of-the-art. Although the proposed ameliorations helped to improve our results, the proposed method has not yet solved the task. To this end, we plan to investigate incorporating more domain knowledge into the decoding process, and to further improve the ODF with RNNs – that has proven to be very useful for similar tasks [1, 4].

7. ACKNOWLEDGMENTS

We are grateful for the GPUs donated by NVidia. This work is partially supported by the Maria de Maeztu Programme (MDM-2015-0502) and the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583).

8. REFERENCES

[1] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *ISMIR*, New York City, USA, 2016.

[2] Paul Boersma. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345, 2001.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[4] Florian Eyben, Sebastian Böck, Björn W Schuller, and Alex Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *ISMIR*, Utrecht, Netherlands, 2010.

[5] Brett Kessler and Rebecca Treiman. Syllable structure and the distribution of phonemes in english syllables. *Journal of Memory and language*, 37(3):295–311, 1997.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[7] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *ICASSP*, Phoenix, USA, 1999.

[8] Anna M. Kruspe. Keyword spotting in a-capella singing. In *ISMIR*, Taipei, Taiwan, 2014.

[9] Emilio Molina, Ana M. Barbancho, Lorenzo J. Tardn, and Isabel Barbancho. Evaluation Framework for Automatic Singing Transcription. In *ISMIR*, Taipei, Taiwan, 2014.

[10] N. Obin, F. Lamare, and A. Roebel. Syll-O-Matic: An adaptive time-frequency representation for the automatic segmentation of speech into syllables. In *ICASSP*, Vancouver, Canada, 2013.

[11] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *ICASSP*, New orleans, USA, 2017.

[12] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *arxiv:1703.06697*, 2017.

[13] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[14] Rafael Caro Repetto and Xavier Serra. Creating a Corpus of Jingju (Beijing Opera) Music and Possibilities for Melodic Analysis. In *ISMIR*, Taipei, Taiwan, 2014.

[15] Odette Scharenborg, Vincent Wan, and Mirjam Ernestus. Unsupervised speech segmentation: An analysis of the hypothesized phone boundaries. *The Journal of the Acoustical Society of America*, 127(2):1084–1095, 2010.

[16] J. Schlüter and S. Böck. Improved musical onset detection with convolutional neural networks. In *ICASSP*, Florence, Italy, 2014.

[17] Chee-Chuan Toh, Bingjun Zhang, and Ye Wang. Multiple-feature fusion based onset detection for solo singing voice. In *ISMIR*, Philadelphia, USA, 2008.

[18] Elizabeth Wichmann. *Listening to Theatre: The Aural Dimension of Beijing Opera*. University of Hawaii Press, 1991.

TOWARDS AUTOMATIC MISPRONUNCIATION DETECTION IN SINGING

Chitralekha Gupta[1,2] **David Grunberg**[1] **Preeti Rao**[3] **Ye Wang**[1]

[1] School of Computing, National University of Singapore, Singapore
[2] NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore
[3] Department of Electrical Engineering, Indian Institute of Technology Bombay, India
chitralekha@u.nus.edu, wangye@comp.nus.edu.sg

ABSTRACT

A tool for automatic pronunciation evaluation of singing is desirable for those learning a second language. However, efforts to obtain pronunciation rules for such a tool have been hindered by a lack of data; while many spoken-word datasets exist that can be used in developing the tool, there are relatively few sung-lyrics datasets for such a purpose. In this paper, we demonstrate a proof-of-principle for automatic pronunciation evaluation in singing using a knowledge-based approach with limited data in an automatic speech recognition (ASR) framework. To demonstrate our approach, we derive mispronunciation rules specific to South-East Asian English accents in singing based on a comparative study of the pronunciation error patterns in singing versus speech. Using training data restricted to American English speech, we evaluate different methods involving the deduced L1-specific (native language) rules for singing. In the absence of L1 phone models, we incorporate the derived pronunciation variations in the ASR framework via a novel approach that combines acoustic models for sub-phonetic segments to represent the missing L1 phones. The word-level assessment achieved by the system on singing and speech is similar, indicating that it is a promising scheme for realizing a full-fledged pronunciation evaluation system for singing in future.

1. INTRODUCTION

Educators recommend singing as a fun and effective language learning aid [6]. In fact, it has been observed that the use of songs and karaoke is helpful in teaching and improving pronunciation in adult second language (L2) classes [1, 17] . Scientific studies have shown that there is a connection between the ability of phonemic production of a foreign language and singing ability [16], and singing ability often leads to better imitation of phrases in an unknown language [15]. More recently, evidence from experimental psychology suggests that learning a new language through songs helps improve vocabulary gain, memory recall, and pronunciation [11]. Additionally, singing releases the need to focus on prosody, as melody of the song overrides the prosodic contrasts while singing [13]. So, given a familiar melody, all the attention can be on articulating the lyrics correctly.

Given the potential of singing in pronunciation training, it is of interest to research automatic pronunciation evaluation for sung lyrics similar to the large body of work in computer-aided pronunciation training (CAPT) for speech [18, 29]. There is little work on mispronunciation detection for sung lyrics. Jha et al. attempted to build a system for evaluating vowels in singing with Gaussian Mixture Model (GMM) and linear regression using Mel Frequency Cepstral Coefficients (MFCC) and pitch as features [12]. However, they did not account for possible pronunciation error patterns in singing, and further, their work did not extend to consonants. There have been a few other studies that have subjectively compared the pronunciation in singing versus that in speech. Yoshida et al. [26] conducted a subjective mispronunciation analysis in singing and speech in English for Japanese natives and found that the subjects familiar with singing tend to make less mistakes in pronunciation while singing than speaking. Another study found that the most frequent pronunciation errors by Indonesian singers in singing English songs occur in the consonants [21]. None of these studies however attempted to build an automatic evaluator of pronunciation in singing.

Though studies have been conducted to compare singing and speech utterances [5, 19], the automated assessment of singing pronunciation is hampered by the lack of training datasets of phone-level annotated singing. Duan et al. created a dataset to analyse the similarities and differences between spoken and sung phonemes [8]. This dataset consists of sung and spoken utterances from 12 unique subjects, out of which 8 were noted as non-native speakers. But their work did not study the pronunciation error patterns in singing or speech. A part of this dataset was phonetically transcribed in 39 CMU phones [25], which is inadequate for annotating non-native pronunciations. We use a subset of audio clips from this dataset for our work (as explained in Section 2.2). But we did not use their phonetic annotations due to these limitations.

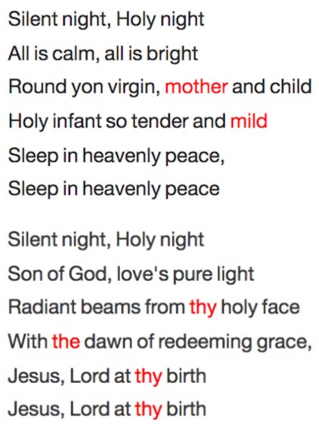
In this work, we demonstrate a knowledge-based approach with limited data to automatically evaluate pronunciation in singing in an automatic speech recognition (ASR) framework. We will adopt a basic method of phonetic evaluation that is used for speech, i.e. deriving pronunciation variants based on L1-L2 pair error patterns, and incorporating this knowledge in the ASR framework for evaluation [3, 23]. In our study, we analyze the error patterns in singing versus those in speech in the accents of South-East Asia - Malaysian, Indonesian, and Singaporean. South-East Asia is one of the most populous regions of the world, where the importance of speaking standard English has been recognized [14], and hence such a pronunciation evaluation system is desired. Given that the data available to train acoustic models is restricted to a native American English speech database [10], we present a novel approach of combining sub-phonetic segments to represent missing L1-phones. Also, we demonstrate how the systematic incorporation of the knowledge of the error patterns helps us obtain a reliable pronunciation evaluation system for non-native singing.

2. PRONUNCIATION ERROR PATTERNS

2.1 Previous Studies

In the process of learning a second language L2, a common type of mispronunciation is replacing phones of L2 that do not exist in the native language (L1) with the closest sounding phoneme of L1 [4]. In Malaysian, Singaporean, and Indonesian English, the dental fricatives /th/ and /dh/ are often substituted by alveolar stops /t/ and /d/ respectively (eg. "three"→"tree", "then"→"den"). These accents are influenced by Malay, Mandarin, and Indonesian languages, in which the dental fricatives /th/ and /dh/ are absent [2, 7, 9]. Also, a pattern particularly seen in Indonesian English accent is that the alveolar stop consonants /t/ and /d/ tend to be substituted by their apico-dental unaspirated stop variant. The reason for this confusion is that in the Indonesian language, the phones /d/ and /t/ can be both alveolar or dental [2, 22]. Another pattern in Singaporean and Malaysian accents is that they tend to omit word-end consonants, or replace them with glottal stops. Note the lack of word-end consonants in the Malay counterparts of words like "product" is "produk". Also in Mandarin, most words do not end with a consonant, except /ng/ and /n/. Vowel difficulties are seen in all these accents, such as long-short vowel confusions like "bead"→"bid", because the long /iy/ is absent in the Indonesian language. Another clear pattern reported is the voiced post-alveolar approximant /r/ in English being pronounced as an apical post-dental trill /r/ in Indonesian, that sounds like a rolling "r".

Here, we investigate the general rules of mispronunciation in singing, which will be then used for automatic pronunciation evaluation in singing. We will, henceforth, refer to the L1 roots of Malaysian, Singaporean, and Indonesian English as M, S, and I, respectively.


Silent night, Holy night
All is calm, all is bright
Round yon virgin, mother and child
Holy infant so tender and mild
Sleep in heavenly peace,
Sleep in heavenly peace

Silent night, Holy night
Son of God, love's pure light
Radiant beams from thy holy face
With the dawn of redeeming grace,
Jesus, Lord at thy birth
Jesus, Lord at thy birth

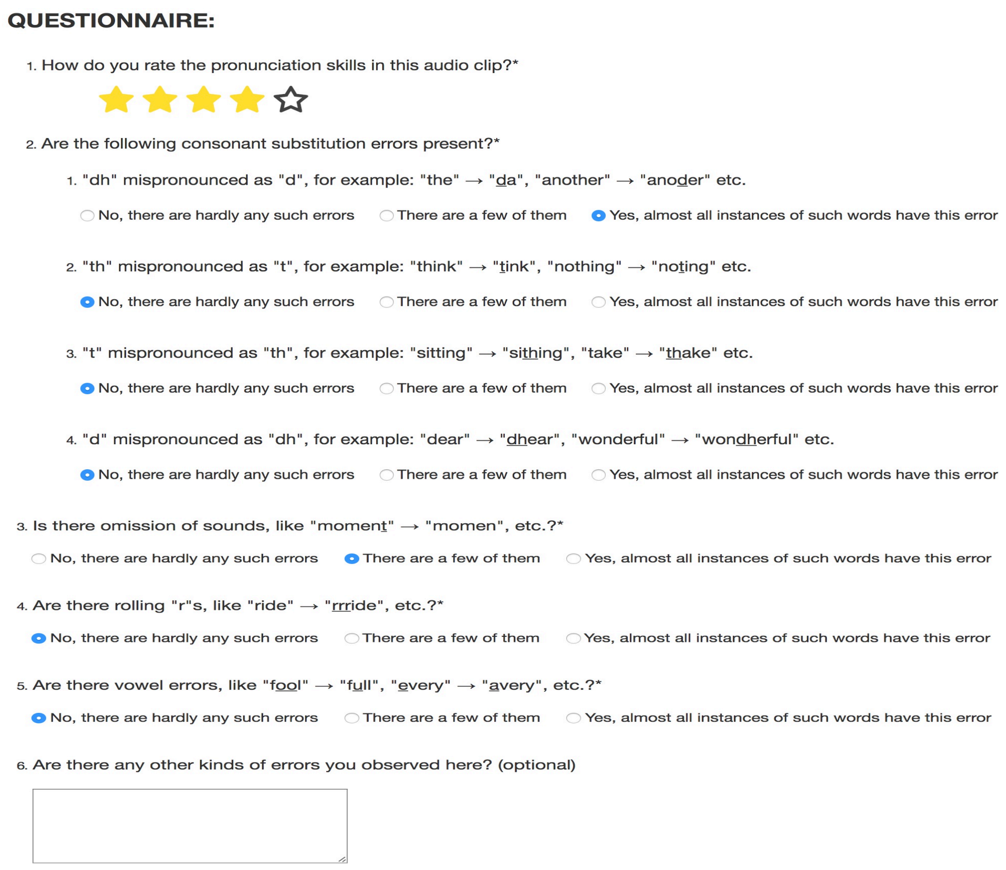
Figure 1: Example of word-level subjective evaluation on the website (incorrect words marked in red).

2.2 Dataset

The dataset (a subset of a published dataset [8]) consists of a total of 52 audio files: 26 sung and 26 spoken, from 15 popular English songs (like Do Re Mi, Silent Night, etc.). Each song has 28 lines (phrases) on an average. These songs were sung and spoken by 8 unique subjects (4 male, 4 female) - 3 Indonesian, 3 Singaporean, and 2 Malaysian. The subjects were students at National University of Singapore, with experience in singing. The subjects were asked to familiarize themselves with lyrics of the songs before the recording session and could use a printed version of the lyrics for their reference during recording. No background accompaniments were used while recording except for metronome beats which were sent to headphones.

We developed a website to collect subjective ratings for this dataset. The website consisted of the audio tracks, their corresponding lyrics, and a questionnaire. Each word in the lyrics could be clicked by the rater to mark it as incorrectly pronounced (red), as shown in the screenshot of the webpage in Figure 1. For each sung and spoken audio clip, the raters were asked to first listen to the track and mark the words in the lyrics that were incorrectly pronounced, and then fill up the questionnaire based on their word-error judgment, as shown in Figure 2. We asked 3 human judges (two North American native English speakers, and one non-native speaker proficient in English), to do this task. Here, native English pronunciation (North American) is considered as the benchmark for evaluating pronunciation.

In the questionnaire, the judges evaluated the overall pronunciation quality on a 5 point scale. On a 3 point scale, they evaluated the presence of each consonant substitution (C1-C4), vowel replacement (V), word-end consonant deletion (CD), and rolling "r" (R), each corresponding to the rules listed in Table 1, where rating 1 means there


QUESTIONNAIRE:

1. How do you rate the pronunciation skills in this audio clip?*

★★★★☆

2. Are the following consonant substitution errors present?*

   1. "dh" mispronounced as "d", for example: "the" → "da", "another" → "anoder" etc.

      No, there are hardly any such errors There are a few of them [✓ Yes, almost all instances of such words have this error]

   2. "th" mispronounced as "t", for example: "think" → "tink", "nothing" → "noting" etc.

      [✓ No, there are hardly any such errors] There are a few of them Yes, almost all instances of such words have this error

   3. "t" mispronounced as "th", for example: "sitting" → "sithing", "take" → "thake" etc.

      [✓ No, there are hardly any such errors] There are a few of them Yes, almost all instances of such words have this error

   4. "d" mispronounced as "dh", for example: "dear" → "dhear", "wonderful" → "wondherful" etc.

      [✓ No, there are hardly any such errors] There are a few of them Yes, almost all instances of such words have this error

3. Is there omission of sounds, like "moment" → "momen", etc.?*

   No, there are hardly any such errors [✓ There are a few of them] Yes, almost all instances of such words have this error

4. Are there rolling "r"s, like "ride" → "rrride", etc.?*

   [✓ No, there are hardly any such errors] There are a few of them Yes, almost all instances of such words have this error

5. Are there vowel errors, like "fool" → "full", "every" → "avery", etc.?*

   [✓ No, there are hardly any such errors] There are a few of them Yes, almost all instances of such words have this error

6. Are there any other kinds of errors you observed here? (optional)

Figure 2: Questionnaire for every audio clip on the subjective evaluation webpage.

are hardly any errors of that category, while rating 3 means almost all of the occurrences have error. We shall refer to these ratings as Rating Set 1. These questions cover all the phone error categories in speech in the accents concerned according to the literature, as described in section 2.1. Additionally, the questionnaire included a comment text-box in which the rater could mention about any other kinds of errors that they observed, which were not covered by the other questions. In this way, we tried to ensure that the subjective evaluation was not biased by the mentioned error categories in the questionnaire.

The average inter-judge correlation (Pearson's) of the overall rating question was 0.68 for the sung clips and 0.62 for the spoken clips, and that for the questions on error-categories was 0.89 for the sung clips and 0.74 for the spoken clips. Thus the inter-judge agreement was high. Also, in the comment text-box, the judges provided only rare minor comments, such as mispronouncing "want to" as "wanna", which could not be categorized as systematic errors due to L1 influence, and hence are not included in the current study.

We chose the word-level pronunciation assessment ("correct"/ "wrong") of one of the North American native English speaking judges as the ground truth. We shall refer to these ratings as Rating Set 2.

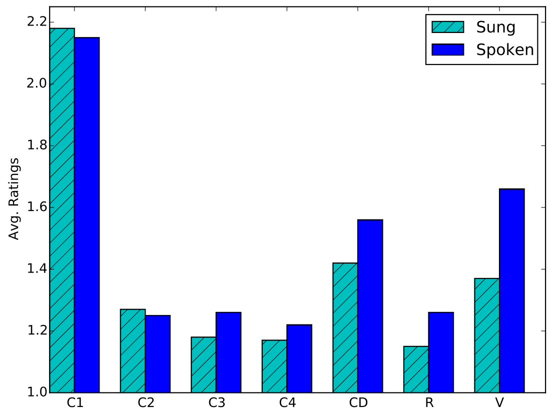


Figure 3: Average subjective rating for seriousness of each error category for singing and speech. Error category labels are as per Table 1.

2.3 Analysis of Error Patterns: Singing vs. Speech

From the rating set 2, we obtained a list of consonant and vowel error patterns in speech and singing, and examples of such words, as shown in Table 2. These error categories can be directly mapped to the questions in the questionnaire, and are consistent with the literature on error patterns in South-East Asian accents.

Our aim here is to derive a list of rules relevant to mis-

Label	Error Rule	p-value
C1	/dh/ → /d/ (WB,WM)	0.414
C2	/th/ → /t/ (WB,WM)	0.382
C3	/t/ → /th/ (WB,WM,WE)	0.079
C4	/d/ → /dh/ (WB,WM,WE)	0.243
CD	Consonant deletion (WE)	**0.032**
R	Rolling "r"	0.112
V	Vowel substitutions	**$5*10^{-5}$**

Table 1: Statistical significance of the difference of each error category between singing and speech (WB: Word Beginning, WM: Word Middle, WE: Word Ending).

pronunciation in singing. From rating set 1, we compute the average subjective rating for each of the mispronunciation rules for singing and speech, as shown in Figure 3. To identify the rules that are relevant for singing, we compute the difference of the average ratings between singing and speech for every rule for each of the 26 pairs of audio files, and compute the $p - value$ of this difference. For a particular rule, if the overall average rating of singing is less than that of speech, and the difference of average ratings between singing and speech is significant ($p - value < 0.05$), then that particular kind of mispronunciation is not frequent in singing, and thus the rule is not relevant for singing. For example, we found that most of the detectable mispronunciations in singing were seen in consonants, which agrees with the literature previously discussed [2, 7, 9, 21, 22]. The mean rating for the question "Are there vowel errors?" was much lower for singing than for speech, meaning that there are fewer vowel errors perceived in singing than in speech (as shown in Figure 3). The difference of the 26 ratings for this question between singing and speech is statistically significant ($p - value = 5 \times 10^{-5}$) (Table 1), and hence confirms this trend. In singing, the vowel duration and pitch in singing are usually dictated by the corresponding melodic note attributes, which makes it different from spoken vowels. For example, the word "sleep" is stretched in duration in the song "Silent Night", thus improving the pronunciation of the vowel. However, in speech the word might tend to be pronounced as "slip". The explanation could lie in the way singers imitate vowels based on timbre (both quality and duration) rather than by the categorical mode of speech perception which is applied only to the consonants. In the same way, we also found that the "word-end consonant deletion" category of errors is significantly less frequent in singing than in speech ($p - value = 0.032$) (Table 1). This implies that either the word-end stop consonants like /t/ and /d/ are hardly ever omitted in singing or are imperceptible to humans. This leads us to the conclusion that only a subset of the error patterns that occur in speech are seen to be occurring in singing. This is a key insight that suggests a possible learning strategy: learning this "subset" of phoneme pronunciation through singing, and the rest through speech.

Another interesting inference from Figure 3 is that on an average, singing has a lower extent of perceived pro-

Consonants			
Error	**WB**	**WM**	**WE**
/dh/ → /d/	the, they, thy, then	mother, another,	
/th/ → /t/	thought, thread,	nothing	
/t/ → /th/	to, tea, take	spirits, into, sitting	note, it, got
/d/ → /dh/	drink, dear	outdoors, wonderful	cloud, world
Consonant deletion (only WE)			night, moment, child
rolling "r"	run, ray,round	bread, drop, bright	brighter, after

Vowels		
Error	**Actual word**	**What is spoken**
ow-->ao	golden	gawlden
uw-->uh	fool	full
iy-->ih,ix	seem, sees, sleeping,	sim, sis, slipping
eh-->ae	every	avry

Table 2: Error categories in singing and speech, and examples of words where they occur.

L1	Label	Rule	Dictionary A		Dictionary B	
			Can.	Mis.	Can.	Mis.
M, S, I	C1	WB,WM /dh/ → /d/	dh → vcl d		dh → vcl d vcl dh →vcl d	
M, S, I	C2	WB,WM / th/ → /t/	th → cl t		th → cl t cl th → cl t	
I	C3	WB,WM,WE /t/ → /th/	cl t → th		cl t → cl th	
I	C4	WB,WM,WE /d/ → /dh/	vcl d → dh		vcl d→ vcl dh	

Table 3: Mispronunciation rules for singing, and corresponding transcriptions for Dictionaries A and B. Can.: Canonical, Mis.: Mispronunciation (cl: unvoiced closure, vcl: voiced closure, dh: dental voiced fricative, d: alveolar voiced stop, th: dental unvoiced fricative, t: alveolar unvoiced stop).

nunciation errors compared to speech, which is also indicated by the average of the overall rating, which is higher for singing (singing = 3.87, speech = 3.80). This suggests that if the non-native subject is familiar with a song and its lyrics, he/she makes fewer pronunciation mistakes in singing compared to speech. Also, a non-native speaking accent is typically characterised by L1-influenced prosody as well such as stress and intonation aspects, which can influence subjective ratings. Singing on the other hand uses only the musical score and is therefore devoid of L1 prosody cues.

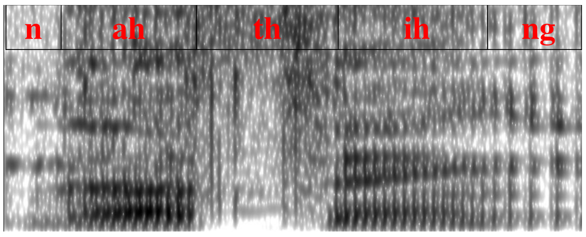
Table 3 lists the L1-specific mispronunciation rules for singing that we derived, in which the word-end consonant deletion and vowel substitution rules have been omitted for reasons mentioned above. In the Indonesian accent, the phone "r" was often replaced with a rolling "r" (trill) (Figure 3), which occurs frequently in singing as well (Table 1). But this phone is absent in American English, so we do not have a phonetic model to evaluate it. So, we have excluded this error pattern in this study.

With our dataset of sung utterances from 8 subjects, we could see clear and consistent mispronunciation patterns across the speakers in our subjective assessment study, and these patterns agree with the phonetic studies of L1 influence on English from these accents in the literature. Therefore, even if the dataset is small, it captures all the expected diversity.

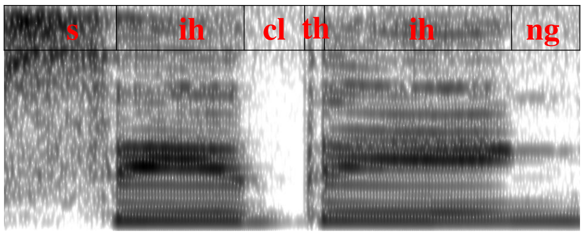
3. AUTOMATIC DETECTION OF MISPRONUNCIATION IN SINGING

Our goal is to use an automatic speech recognition (ASR) system to detect mispronunciations in singing. In previous studies, vowel error patterns in Dutch pronunciation of L2 learners were used by Doremalen et al. to improve automatic pronunciation error detection systems [23]. In another work, Black et al. derived common error patterns in children's speech and used an adapted lexicon in an extended decoding tree to obtain word-level pronunciation assessment [3]. A standard way to detect mispronunciation is to let the ASR recognize the most likely sequence of phonemes for a word from a given set of acceptable (canonical) and unacceptable (mispronounced) pronunciation variants of that word. A pronunciation is detected as unacceptable if the chosen sequence of phonemes belongs to the list of unacceptable pronunciation variants of the word (called the "lexicon" or the "LEX" method in [3]). While the present work is similar in principle to the above, we face the additional challenge of lack of training data for the L1 phones not present in L2. Yu et al. [27] have used a data-driven approach to convert the foreign-language lexicon (L2) to native-language lexicon (i.e. using L1 phones only), where they had large L1 speech training data, in contrast to our case of availability of L2 training speech only. In the current work, we use a novel knowledge-based approach to overcome the constraints of lack of L1 training data for both speech and singing. We compare the case of restricting ourselves to L2 phones with a method that uses L1 phones derived from a combination of sub-phonetic segments of L2 speech to approximate unavailable L1 phones.

We compare a Dictionary A that contains only American English (TIMIT [10]) phones (L2), with a Dictionary B that contains TIMIT phones+modified (L1-adapted) phones. To design Dictionary B, we compared the phones of the South-East Asian accents with that of the American English TIMIT speech dataset [10]. We found that the dental fricatives /th/ and /dh/ are often mispronounced as alveolar stops /t/ and /d/ respectively (rules C1, C2). Both of the substituted phones /t/ and /d/ are present in American English, and hence their phone models are available in TIMIT. But when L1 is Indonesian, the alveolar stop consonants /t/ and /d/ tend to be substituted by their apico-dental unaspirated stop variant (rules C3, C4), as explained in Section 2.1. But dental stop phones are not annotated in American English datasets like TIMIT [10]. In order to solve this problem of lack of dental stop phone models in L2, we combined sub-phonetic TIMIT models. We observed that the dental stop phones consist of a closure period followed by a burst period with dental place of articulation. So we combined the TIMIT models for unvoiced closure model /cl/ with the unvoiced dental fricative model /th/ to approximate unvoiced dental stop /t/, as shown in Figure 4, and voiced closure model /vcl/ with the voiced dental fricative model /dh/ to obtain voiced dental stop /d/. It is important to note that in these accents, the dental fricatives /th/ and /dh/ are also often substituted by dental stops /cl th/ and /vcl dh/. But this particular substitution pattern is common in American English [28], and hence not considered to be mispronunciation. Hence, we add these variants to the list of acceptable variants (canonical).



(a)



(b)

Figure 4: (a) American speaker (TIMIT) articulating word-middle unvoiced dental fricative /th/ in "nothing" (note: there is no closure) (b) Indonesian speaker substituting unvoiced alveolar stop with unvoiced dental stop ("sitting" as "sithing") modeled as /cl th/.

In summary, the mispronunciation rules in Dictionary B are: dental fricative and stop /dh/ being mispronounced as alveolar stop /d/ (L1: M, S, I); dental fricative and stop /th/ being mispronounced as alveolar stop /t/ (L1: M, S, I); alveolar stop /t/ being mispronounced as dental stop /th/ (L1: I); and alveolar stop /d/ being mispronounced as dental stop /dh/ (L1: I). These mispronunciation rules are listed in Table 3.

3.1 Methodology

We use the toolkit KALDI [20] for training 48 context independent GMM-HMM and DNN-HMM phonetic models using the TIMIT train set [10] with the parameters set by Vesely et al. [24]. The HMM topology is 3 active states, the MFCC features are frame-spliced by 11 frames, dimension-reduced by Linear Discriminant Analysis (LDA) to 40 dimensions. Maximum Likelihood Linear Transformation (MLLT), feature-space Maximum Likelihood Linear Regression (fMLLR), and Cepstral Mean and Variance Normalization (CMVN) are applied for speaker adaptive training. The DNN has 6 hidden layers, 2048 hidden units per layer. The Restricted Boltzmann Machine (RBM) pre-training algorithm is contrastive divergence and the frame cross-entropy training is done by mini-batch stochastic gradient descent. Phone recognition performance of the acoustic models trained and tested on TIMIT was consistent with the literature [24].

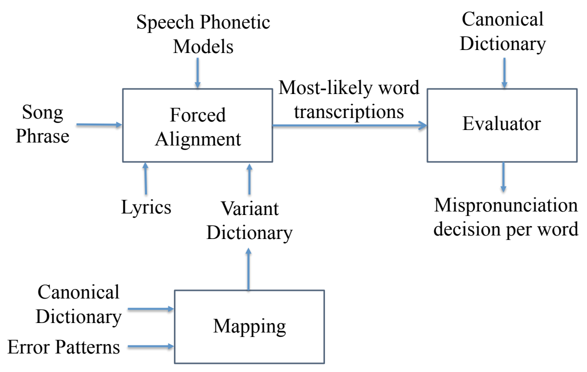


Figure 5: Overview of automatic mispronunciation detection in singing.

L1 (#EP-W)	AM	Speech						Singing					
		Dictionary A			Dictionary B			Dictionary A			Dictionary B		
		P	R	F	P	R	F	P	R	F	P	R	F
M,S (245)	DNN-HMM	0.51	0.58	0.54	0.60	0.68	**0.63**	0.62	0.65	0.63	0.69	0.66	**0.67**
	GMM-HMM	0.41	0.54	0.47	0.49	0.63	0.55	0.54	0.52	0.53	0.55	0.51	0.53
#GT-E		78						86					
I (834)	DNN-HMM	0.29	0.55	0.38	0.56	0.46	**0.50**	0.23	0.59	0.33	0.42	0.54	**0.47**
	GMM-HMM	0.27	0.50	0.35	0.46	0.40	0.43	0.21	0.58	0.31	0.33	0.34	0.34
#GT-E		219						176					

Table 4: Performance of automatic mispronunciation detection for singing and speech. P: Precision = TP/(TP+FP); R: Recall = TP/(TP+FN); F: F-score = 2.P.R/(P+R); AM: Acoustic Models; #GT-E: no. of error-prone words mispronounced; #EP-W: no. of error-prone words. L1 languages - M: Malaysian, S: Singaporean, and I: Indonesian.

We use these speech trained acoustic-phonetic models, along with the L1-specific variant dictionary (A or B) to force-align the lyrics to the sung and spoken audio files, to obtain word-level automatic pronunciation evaluation by the "LEX" method, as described before. An overview of this system is shown in Figure 5. We first segment the audio files at phrase level by aligning its pitch track with a template containing a reference pitch track and marked phrase-level boundaries, using dynamic time warping. The 26 songs are segmented into 740 phrases, containing a total of 5107 words. For singing, out of these 5107 words, 1079 words are the error-prone words, i.e. they fall under the mispronunciation rules. Only 14 out of the rest 4028 non-error-prone words (0.3%) are subjectively evaluated as mispronounced in singing, which confirms that the mispronunciation rules for singing are correctly identified. To compare speech and singing, we apply the same rules for the speech phrases because we expect that the words that are mispronounced in singing are likely to be mispronounced in speech as well.

Table 4 shows the validation results for the L1-specific error-prone words from the two acoustic model configurations in singing and speech, using the dictionaries A and B, where the ground truth is the word-level subjective evaluation as obtained in rating set 2. To evaluate the performance of the system, we compute the metrics precision, recall, and F-score [3], where TP (True Positive) is the number of mispronounced words detected as mispronounced, FP (False Positive) is the number of correctly pronounced words detected as mispronounced, and FN (False Negative) is the number of mispronounced words detected as correctly pronounced (Table 4).

4. RESULTS AND DISCUSSION

We note that the method of combining sub-phonetic American English models for approximating the missing phone models of L1 is effective as the F-scores indicate that the system using dictionary B outperforms the one using A in all the cases. DNN-HMM outperforms GMM-HMM consistently for the task for pronunciation evaluation in singing, as it has been widely observed in speech recognition. Also, the F-score values of singing and speech are similar, which shows that our knowledge-based approach for singing pronunciation evaluation is promising.

A source of false positives is the rule /t/→/th/ which causes error when /t/ is preceded by a fricative (eg. /s/), for example "just" [jh, ah, s, cl, t]. Since both /s/ and /th/ are fricatives, the system gets confused and aligns /th/ at the location of /s/. A way to handle such errors is to obtain features specific to classifying the target and the competing phonemes, which will be explored in the future.

5. CONCLUSION

In this paper, we have analysed pronunciation error patterns in singing vs. those in speech, derived rules for pronunciation error patterns specific to singing, and demonstrated a knowledge-based approach with limited data towards automatic word-level assessment of pronunciation in singing in an ASR framework. From subjective evaluation of word pronunciation, we learn that nearly all identified mispronunciations have an L1-based justification, and singing has only a subset of the errors found in speech. We provide the rules that predict singing mispronunciations for a given L1. In order to solve the problem of unavailable L1 phones due to the lack of training speech data from L1 speakers, we propose a method that uses a combination of sub-phonetic segments drawn from the available native L2 speech to approximate the unavailable phone models. This method is shown to perform better than the one that restricts to only L2 phones. And finally, the performance of this system on singing and speech is comparable, indicating that this approach is a promising method for developing a full-fledged pronunciation evaluation system. In future, we would explore a combination of data-driven methods such as in [27] and our knowledge-based methods to improve the mispronunciation detection accuracy.

6. REFERENCES

[1] Developing pronunciation through songs. `https://www.teachingenglish.org.uk/article/developing-pronunciation-through-songs`. Accessed: 2017-04-27.

[2] B. Andi-Pallawa and A.F. Abdi Alam. A comparative analysis between English and Indonesian phonological systems. *International Journal of English Language Education*, 1(3):103–129, 2013.

[3] M. Black, J. Tepperman, and S. Narayanan. Automatic prediction of children's reading ability for high-level literacy assessment. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):1015–1028, 2011.

[4] P. Bonaventura, D. Herron, and W. Menzel. Phonetic rules for diagnosis of pronunciation errors. In *KONVENS: Sprachkommunikation*, pages 225–230, 2000.

[5] W. Chou and G. Liang. Robust singing detection in speech/music discriminator design. In *IEEE International Conference on Acoustics, Speech, and Signal Processing Proceedings 2001*, pages 865–868, 2001.

[6] F. Dege and G. Schwarzer. The effect of a music program on phonological awareness in preschoolers. *Frontiers in Psychology*, 2(124):7–13, 2011.

[7] D. Deterding. *Singapore English*. Edinburgh University Press, 2007.

[8] Z. Duan, H. Fang, L. Bo, K. C. Sim, and Y. Wang. The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA) 2013, IEEE*, pages 1–9, 2013.

[9] S.Y. Enxhi, T.B. Hoon, and Y.M. Fung. Speech disfluencies and mispronunciations in English oral communication among Malaysian undergraduates. *International Journal of Applied Linguistics and English Literature*, 1(7):19–32, 2012.

[10] J. Garofolo. TIMIT acoustic-phonetic continuous speech corpus. In *Philadelphia: Linguistic Data Consortium*, 1993.

[11] A. Good, F. Russo, and J. Sullivan. The efficacy of singing in foreign-language learning. *Psychology of Music*, 43(5):627–640, 2015.

[12] P. Jha and P. Rao. Assessing vowel quality for singing evaluation. In *National Conference on Communications (NCC) 2012, IEEE*, pages 1–5, 2012.

[13] I. Lehiste. Prosody in speech and singing. In *Speech Prosody 2004, International Conference*, 2004.

[14] L. Lim, A. Pakir, and L. Wee. *English in Singapore: Modernity and management*, volume 1. Hong Kong University Press, 2010.

[15] C. Markus and S.M. Reiterer. Song and speech: examining the link between singing talent and speech imitation ability. *Frontiers in psychology*, 4:874, 2013.

[16] R. Milovanov, P. Pietil, M. Tervaniemi, and P. Esquef. Foreign language pronunciation skills and musical aptitude:a study of Finnish adults with higher education. *Learning and Individual Differences*, 20(1):56–60, 2010.

[17] H. Nakata and L. Shockey. The effect of singing on improving syllabic pronunciation–vowel epenthesis in japanese. In *International Conference of Phonetic Sciences*, 2011.

[18] L. Neumeyer, H. Franco, V Digalakis, and M. Weintraub. Automatic scoring of pronunciation quality. *Speech Communication*, 30(2):83–93, 2000.

[19] Y. Ohishi, M. Goto, K. Itou, and K. Takeda. Discrimination between singing and speaking voices. In *Proceedings of the Annual Conference of the International Speech Communication Association (Interspeech 2005)*, pages 1141–1144, 2005.

[20] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. The Kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[21] I. Riyani and J. Prayogo. An analysis of pronunciation errors made by Indonesian singers in Malang in singing English songs. *SKRIPSI Jurusan Sastra Inggris-Fakultas Sastra UM*, 2013.

[22] E. Setyati, S. Sumpeno, M. Purnomo, K. Mikami, M. Kakimoto, and K. Kondo. Phoneme-Viseme mapping for Indonesian language based on blend shape animation. *IAENG International Journal of Computer Science*, 42(3), 2015.

[23] J. van Doremalen, C. Cucchiarini, and H. Strik. Automatic pronunciation error detection in non-native speech: The case of vowel errors in Dutch. *The Journal of the Acoustical Society of America*, 134(2):1336–1347, 2013.

[24] K. Vesely, A. Ghoshal, L. Burget, and D. Povey. Sequence-discriminative training of deep neural networks. In *Interspeech*, pages 2345–2349, 2013.

[25] R. Weide. The Carnegie Mellon pronouncing dictionary [cmudict. 0.6], 2005.

[26] K. Yoshida, N. Takashi, and I. Akinori. Analysis of English pronunciation of singing voices sung by Japanese speakers. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IEEE)*, 2014.

[27] D. Yu, L. Deng, P. Liu, J. Wu, Y. Gong, and A. Acero. Cross-lingual speech recognition under runtime resource constraints. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4193–4196. IEEE, 2009.

[28] S. Zhao. Stop-like modification of the dental fricative /dh/: An acoustic analysis. *The Journal of the Acoustical Society of America*, 128(4):2009–2020, 2010.

[29] Y. Zheng, R. Sproat, L. Gu, I. Shafran, H. Zhou, Y. Su, D. Jurafsky, R. Starr, and S. Yoon. Accent detection and speech recognition for shanghai-accented mandarin. In *Interspeech*, pages 217–220. Citeseer, 2005.

UNDERSTANDING THE EXPRESSIVE FUNCTIONS OF JINGJU METRICAL PATTERNS THROUGH LYRICS TEXT MINING

Shuo Zhang
Music Technology Group
Universitat Pompeu Fabra
ssz6@georgetown.edu

Rafael Caro Repetto
Music Technology Group
Universitat Pompeu Fabra
rafael.caro@upf.edu

Xavier Serra
Music Technology Group
Universitat Pompeu Fabra
xavier.serra@upf.edu

ABSTRACT

The emotional content of jingju (aka Beijing or Peking opera) arias is conveyed through pre-defined metrical patterns known as *banshi*, each of them associated with a specific expressive function. In this paper, we first report the work on a comprehensive corpus of jingju lyrics that we built, suitable for text mining and text analysis in a data-driven framework. Utilizing this corpus, we propose a novel approach to study the expressive functions of *banshi* by applying text analysis techniques on lyrics. First we apply topic modeling techniques to jingju lyrics text documents grouped at different levels according to the *banshi* they are associated with. We then experiment with several different document vector representations of lyrics in a series of document classification experiments. The topic modeling results showed that sentiment polarity (positive or negative) is better distinguished between different *shengqiang-banshi* (a more fine grained partition of *banshi*) than *banshi* alone, and we are able to achieve high accuracy scores in classifying lyrics documents into different *banshi* categories. We discuss the technical and musicological implications and possible future improvements.

1. INTRODUCTION

Traditionally, the emotional content of jingju (aka Beijing or Peking Opera) music is conveyed through pre-defined melodic and metrical patterns known as *shengqiang* and *banshi*. With the general absence of professional composers, the melodic material of jingju was taken from local tunes, and lyrics were arranged by performers according to their poetic structure. In order to convey different emotional contents, the original melodic outlines were transformed rhythmically, according to a pre-defined set of labelled metrical patterns. Each of the metrical patterns, known as *banshi*, is associated with an expressive function. Each of the melodic materials to which this metrical patterns were applied is known as *shengqiang*, and is also associated with emotional content at a larger scale.

There exists many general descriptions and rules for the expressive functions associated with each *banshi* in musicological literature [11, 15] and jingju textbooks [2, 3, 14]. However, the actual realization of these associativities across existing jingju repertoires has not been characterized in a clear manner. Such a task is well suited for a data-driven computational analysis.

In this work, we first report the work on constructing the Jingju Lyrics Collection data collection, a comprehensive corpus of jingju lyrics that we built through web scraping `xikao.com`, suitable for text mining and text analysis in a data-driven framework. We describe substructures of this data collection as well as relevant corpus statistics based on musicological entities and considerations. Utilizing this corpus, we propose a novel approach to study the expressive functions of *banshi* by applying text analytics techniques on lyrics.

The rest of the paper is organized as follows. Section 2 provides necessary musicological concepts and background that lead to the research questions we are concerned with, namely, understanding the emotional content of *banshi* metrical patterns through lyrics text analytics. Section 3 reports the construction of the JLC lyrics data collection and describes its substructures as well as relevant corpus statistics. Following the introduction of the JLC data collection, we then report text analytics experiments aimed at revealing different semantic content in different *banshi*, including topic modeling (Section 4) and document classification (Section 5). Finally we discuss the results and future directions.

2. BACKGROUND

As stated above, *shengqiang* (SQ) and *banshi* (BS) are the melodic and rhythmic devices used in arranging the music in jingju. They are selected in order to deliver the emotional content of lyrics, the psychological profile of the characters or the general atmosphere of the play. The two main *shengqiang* of jingju are *xipi* and *erhuang*. There are around twelve types of most common *banshi*, each interrelated with others. For example, *yuanban*, literally meaning 'original meter', is considered a default medium tempo meter, and the rest of *banshi* can be considered transformations of this one: *manban*, the result of slowing down *yuanban* in tempo and stretching it in meter; *kuaiban*, the result of speeding *yuanban* up in tempo and compressing it

Banshi	**Code**	**Postulated expressive function**	**Musical feature**
yuanban	YB	straightforward, unemotional, narration, facts and explanation	medium
manban	MB	peaceful, introspective	slow
kuaiban	KB	animated, excitement, anticipation	fast
yaoban	YAB	exterior calm and interior tension	free meter
Sheng -qiang	**Code**	**Postulated expressive function**	**Musical feature**
xipi	XP	sprightly, bright and clear, energetic, forceful, and purposeful	melodic skeleton
erhuang	EH	dark, deep and profound, heavy and meticulous	melodic skeleton

Table 1. List of common *banshi* (rhythmic) and *shengqiang* (melodic) types and their acronyms (Code) in this paper. The entity column contains 4 *banshi* types in the first 4 rows, and 2 *shengqiang* types in the last 2 rows.

in meter, etc. The combination of a *shengqiang* with a particular *banshi* results in a unique musical form (henthforce referred to as SQBS), which is referred to by combining both elements, such as *erhuang_yuanban*, *xipi_manban*, etc. In general, *shengqiang* are associated with general emotional frameworks, and *banshi* with specific expressive functions [11]. Table 1 lists the most common types of *banshi* and *shengqiang*, their musical and postulated expressive functions.

The goal of this paper is twofold: first, to introduce a jingju lyrics corpus that we constructed especially for computational text analysis in this domain; second, to understand the expressive functions associated with each *banshi* through large-scale text analytics. In the current context, we take the implicit assumption that the emotional content (i.e., the target of the expressive functions for a *banshi*) of the jingju can be represented by inspecting the semantic content expressed in lyrics. We define the following research questions: (1) What are the document-topic-word distributions that characterize the lyrics texts found in each type of *banshi*? (2) How distinct are these distributions among different *banshi*? (3) Are we able to distinguish between one *banshi* and another from lyrics? (classification) (4) How does the interplay between *shengqiang* and *banshi* affect this characterization?

In recent years, there has been a number of studies employing data-driven and computational approaches to various facets of jingju music [6–8,12,13]. However, all of the previous works rely on audio recording or score as their primary data source. To the best of our knowledge, the current work is novel in its use of large-scale lyrics text corpus for jingju and the application of state-of-the-art NLP and text mining algorithms to uncover the associations between jingju music and expressive functions.

3. BUILDING THE JINGJU LYRICS COLLECTION

3.1 Web Scraping Xikao Database

There are a limited number of traditional style plays in the jingju repertoire (as they are not being expanded much in modern times). In order to build a comprehensive corpus of collection of jingju lyrics, we have chosen to extract data from the well-maintained open source jingju libretto database website `xikao.com`. As this website provides jingju librettos in HTML and PDF formats not ready for corpus analysis purposes, we have crawled the website to extract all lyrics in plain text format through web scraping. All texts from this website is of Creative Commons License and is free to use for non-commercial purposes. We denote our overall collection of the lyrics data (including subsequence creation of substructures within the collection) as JLC (Jingju Lyrics Collection). We use this generic name to accommodate future possibilities of expanding the collection from other sources.

`xikao.com` is a community collaboration platform aimed at building the most comprehensive collection of jingju plays for jingju professionals and aficionados by collaboratively digitizing published jingju librettos available in prints. It is being actively maintained since its inception in 2000, and there has been a steady growth in the number of digitized librettos. At the time of writing, there are a total of 2163 published librettos in print being considered for digitization, whereas there are 850 works already digitized and proof-read/edited, and there are currently 360 plays at the various stages of being digitized by dozens of anonymous users/editors/annotators. Due to the dynamic growth of its content, we can also periodically re-apply our web scraping pipeline in order to expand our data collection to reflect the most comprehensive coverage to date.

Librettos in Xikao is organized by play as a basic unit. Metadata, *banshi* (metrical pattern), *shengqiang* (melodic skeleton), role type, as well as other information such as the instrumental interlude and oral delivery mode (spoken dialogue, singing) are also annotated in the digitized documents. Meanwhile, as noted above, the overall goal of Xikao is not oriented towards computational analysis, therefore we need to apply several transformations in order to create the most useful data sets for our study (detailed in Section 3.2 and 3.3). Several examples of these shortcomings are illustrated here. First, the organization by play may not be the most useful for analysis aimed at understanding *shengqiang* or *banshi* or other musicologically meaningful categories. Second, the meta data information are also spread within the documents, making it hard to retrieve in an straightforward way. Overall, the Xikao website in its original form (before or after we have scraped its contents and stored in plain text files) is considered unstructured data that needs to be re-structured and augmented in order to use for large scale data-drive text analysis.

3.2 Text Processing

In a post-processing stage to the web scraping, we extract all lyrics that are sung to a particular *banshi* type clearly indicated (there are a good portion of a play that are spoken dialogue). As part of the standard NLP pipeline for Chinese [1], we perform word segmentation [2] on all text documents using the state-of-the-art, Conditional Random Field-based Stanford Word Segmenter [3] [10]. The result of the segmentation is verified by hand by a native speaker of Chinese and deemed reasonable. [4] After the segmentation is obtained, we use Unicode based tokenization (splitting on whitespaces) in the study, where each token is defined as one or more Unicode character [5]. In all subsequent processing steps we remove 125 frequent single-character words using a standard stop word list of Chinese. All punctuations are removed as a normalization step for NLP pipeline. The resulting corpus contains lyrics text files for 818 plays, a quite large size to study considering the small number of jingju plays that are still being performed today.

3.3 Data Sets Permutation and Creation

Following preprocessing, we extract subsets of the data collection and restructure them in order to create musicologically meaningful datasets for computational text analysis of jingju lyrics. We consider the creation of several data sets within this framework.

SQBS Dataset: This general data set consists of lyrics from all lines in all plays that correspond to a SQBS, in a tabular format, where the first column indicates SQBS category, second contains the lyrics line. Here, it's worth pointing out a 'lyrics line' is referring to the longest unit an actor is singing continuously in the same SQBS without switching to or interrupted by any other SQBS. The total number of SQBS categories in this data set is 151. In this case, it is possible to observe the distribution of the frequencies of each SQBS category. In Figure 1 and Figure 2, we show the top 10 SQBS categories in the SQBS dataset by number of lines and by number of words/characters (where a 'line' is defined as above). With few exceptions, we can see that the top 10 most frequent categories are mostly consistent when considered by number of lines vs. by number of words/characters. Meanwhile, we note that *xipi_yaoban* (XPYAB) is the most frequent musical form in jingju by all measures, which is often used in singing in the middle of spoken dialogues.

[1] Here we apply a shallow pipeline of word segmentation, tokenization, and stop-word removal.

[2] Since the Chinese language is written without spaces between characters and words, the word segmentation is a necessary and challenging task for any NLP or text mining analysis of Chinese text.

[3] Obtained at http://nlp.stanford.edu/software/segmenter.shtml.

[4] The linguistic style of the jingju lyrics is a mixed style that is typically similar to modern Chinese yet with occasional semi-classical style language. Therefore, unless there is a segmenter trained specifically on this language, using any pre-trained segmenter would not have yielded a perfect segmentation. To give an estimate of error, the original Stanford Segmenter paper [10] gives a overall F-score around 0.95, with a recall of known vocabulary greater than 0.95 and a recall of unknown vocabulary (OOV) in the range of 0.7s.

[5] In Chinese, a 'word' can be any number of characters, most commonly 1,2,3, or 4.

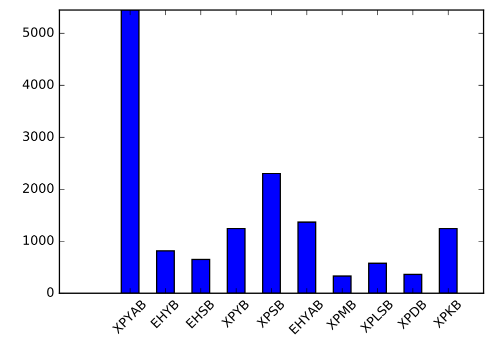


Figure 1. Distribution of top 10 SQBS categories by number of lines

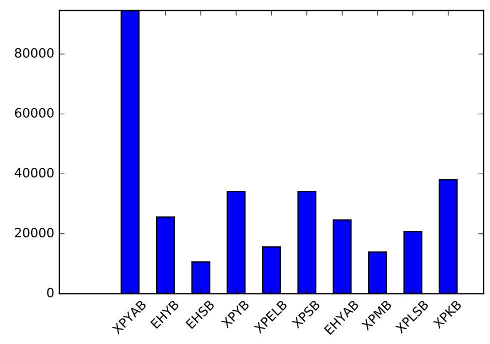


Figure 2. Distribution of SQBS top 10 categories by number of words/characters

SQBS7 Dataset: Among the 151 SQBS categories, we have selected a core set of 3 *banshi* types coupled with the two *shengqiang*, based on their musicological importance and the frequency of their occurrence in our corpus. Concretely, we consider the most basic *banshi* types for the two main *shengqiang*, that is *yuanban*, *manban*, and *kuaiban* for *xipi* and *erhuang* [6]. To also include a non metered *banshi*, we have also considered the one with a more frequent occurrence in our corpus, that is, *yaoban*, giving rise to 7 core SQBS categories that are most representative in analysis. We denote this data set as SQBS7, which is a subset of SQBS data set. All following data sets to be used in this study are transformed from the SQBS7 dataset.

PL* Datasets: The PL* data sets are grouped by play and one or two other musicological entities (BS or SQBS). First, we create the PLay-ShengQiang-BanShi (PLSQBS) data set, where a document is defined to be all the texts associated with a particular *shengqiang_banshi* within the same play. For example, all the *erhuang_ manban* texts from one play form one PLSQBS_document, whereas all the *erhuang_ yuanban* texts from the same play form another PLSQBS_document. This is aimed at looking at a particular combination of *shengqiang_banshi* type. Second, we collapse all *shengqiang* categories and create the

[6] It has to be noticed that in traditional plays *erhuang* was never set to *kuaiban*.

PLay-BanShi (PLBS) documents. Each document in this data set is defined to be all the texts associated with a particular *banshi* within the same play. Therefore, regardless of *shengqiang*, all the *manban* texts from one play form one PLBS document.

One potential problem with the PL* data sets is that the partition of documents may result in very short documents, creating a data sparseness problem in document modeling algorithms. Figure 4 shows the distribution of document length in the PLSQBS data set. We observe that there is a peak at less than 200 words per document, whereas there are also a few documents with more than 2000 or 4000 words. This may be taken into consideration when performing text mining experiments on these data sets.

AG* Datasets: We aggregate all PLSQBS documents to form 7 AGSQBS ('AG' for aggregate) documents (as there are 7 types of *shengqiang_banshi* considered in the current study), in order to study the characteristics in all texts associated with a particular *shengqiangbanshi*. By analogy, we aggregate all PLBS documents to form the 4 AGBS documents (i.e., all texts for a particular *banshi*).

We provide an overview of the relationships between data sets in Figure 3. Table 2 gives a more detailed description of the PL* and AG* data sets used in the subsequent experiments. The entire data collection is openly available through Github [7], and the datasets used in the experiments of this paper are available through CompMusic project website [8].

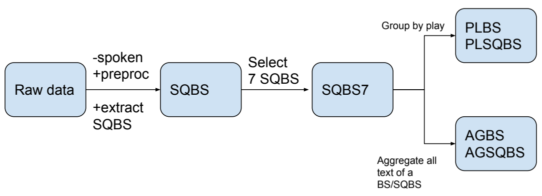


Figure 3. Block diagram overview of data sets created

Name	Description of document	#documents
PLSQBS	all the texts associated with a particular *shengqiang-banshi* within a particular play	1429
PLBS	all the texts associated with a particular *banshi* within the a particular play	1247
AGSQBS	all the texts associated with a particular *shengqiang-banshi*	7
AGBS	all the texts associated with a particular *banshi*	4

Table 2. Overview of data sets used in the experiments of this paper

[7] https://github.com/MTG/Jingju-Lyrics-Collection
[8] http://compmusic.upf.edu/jingju-lyrics-datasets

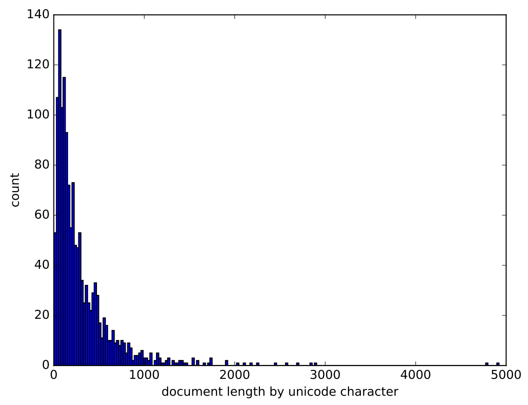


Figure 4. Distribution of document length in the PLSQBS dataset

4. EXPLORING TOPIC STRUCTURES OF SUBSECTIONS OF JLC DATA COLLECTION

In this section, we use probabilistic topic models to explore the topic structures of lyrics in different *banshi* in an unsupervised setting [9].

4.1 Topic Modeling

Topic model is a class of unsupervised statistical models for uncovering the underlying semantic structure of a document collection. The idea is to model each document as arising from multiple background (latent) topics, where a topic is defined to be a distribution over a fixed vocabulary of terms. Specifically, we assume that K topics are associated with a collection, and that each document exhibits these topics with different proportions. In a topic model, we typically obtain a document-topic distribution (the probabilistic distribution of all topics in a particular document), and a topic-word distribution (the distribution of the terms that are associated with one topic). In the current context, we are interested in the topics that characterize each *banshi*(BS) or *shengqiang_banshi*(SQBS) in the AGBS and AGSQBS data sets.

Here we consider a state-of-the-art topic modeling techniques known as Latent Dirichlet Allocation(LDA) [1]. In LDA, each topic z is associated with a multinomial distribution over the vocabulary Φ_z, which is drawn from a Dirichlet prior $Dir(\beta)$. A given document D_i is then generated by the following process:(1) Choose Θ_i ˜$Dir(\alpha)$, a topic distribution for D_i; (2) For each word $w_j \in D_i$: (a) Select a topic z_j ˜Θ_i (b) Select the word w_j ˜Φ_{zj}. We use collapsed Gibbs sampling implementation in Mallet [10] to infer the values of the latent variables Φ and Θ.

We compute the perplexity measure for a held-out data set defined in the LDA model to determine the optimal number of background latent topics in the current experiments. The perplexity, used by convention in language

[9] The code for all experiments in this paper is available at https://github.com/MTG/Jingju-Lyrics-Text-Analysis.
[10] Downloaded from http://mallet.cs.umass.edu/

YB	0:Hanxin (military General), youth, wealth, ruthless
KB	2:family, mother, husband-wife, brother; 16: war, military, mails, destruction
MB	4:princess, death, crime, Chang'an, brave; 19: wish, sir, madam, pain, defeat
YAB	7: sudden news, human head, revenge, affection

Table 3. Top topics for each *banshi* and their top words. Sorted by topic index number (0 is topic 0 assigned by the LDA model, etc.)

modeling, is monotonically decreasing in the likelihood of the test data, and is algebraically equivalent to the inverse of the geometric mean per-word likelihood [1]. A lower perplexity score indicates better generalization performance. More formally, for a test set of M documents, the perplexity is:

$$perplexity = exp\left\{ -\frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d} \right\} \quad (1)$$

where w_d is a word in the document, and N_d is the length of the document (total number of words). In our experiments, we compute perplexity for each topic model given number of topics from 8 to 50. The result shows that using 20 topics results in the lowest perplexity and is the optimal choice.

4.2 Topic Modeling Results

First, we present the topic modeling results for the AGBS and AGSQBS data sets. In this case, the output from the MALLET LDA model contains a document-topic distribution and a topic-word distribution. The former shows the distribution of topics (number of topic K=20, as determined in Section 4.1) in each BS/SQBS (i.e., each document in the data set), and the latter shows the top 20 words associated with each topic. Since we are interested in characterizing the main topics found in each BS/SQBS, we show results for both of these components.

To better understand the topics that characterize each category, we extracted the most salient topics from the topic distribution of each BS or SQBS. In doing so, we removed common topics with high occurrences in all categories, and only select those with a high occurrences in each category. We present these topics and summarize their top words in English in Table 3 and Table 4. Many of these topics have to do with specific stories in Chinese history that are well known in jingju repertoires. Comparing these results to the general descriptions found in Table 1, we see a reasonable interpretation for each category - although we observe that the division between the positive and negative emotions of *xipi banshi* and *erhuang banshi* in Table 4 are much more salient than the topics that distinguish the four *banshi* types in Table 3.

EHYB	0: pity, old, heavy, prince, depart, pain, cold
EHMB	2: unfortunate, pity, worry; 14: war, military, courtesy, hero
EHYAB	4: tears, sir, madam, wish, leave, pain, hurt, stab; 18: life, run, death, brave, sword
XPYB	7: Kings from The Three Kingdoms, drink, happy
XPKB	10: traitor, laugh, believe
XPMB	12: youth, beautiful view, morning, world
XPYAB	11: general(military), prime minister, angry, military, step forward; 14: (see EHMB)

Table 4. Top topics for each *shengqiang-banshi* (SQBS) and their top words. Sorted by topic index number(0 is topic 0 assigned by the LDA model, etc.)

5. DOCUMENT CLASSIFICATION IN JLC DATA COLLECTION

In this section we propose a supervised document classification task with the aim of classifying lyrics documents in the PL* data sets into different *banshi* categories.

5.1 Document Vector Representation

In the vector-space model (VSM) of information retrieval, a text document is represented by a document-term vector where each attribute represents the frequency (count) with which a particular term $w_{k,i}$ (a word in the vocabulary) occurs in the document d_i (aka bag-of-words or BOW). A highly effective transformation of BOW word vector weights is tf-idf weighted vectorization. [11] However, a shortcoming of these types of traditional word vectors is that it is high-dimensional and very sparse.

Recent advances in NLP have concentrated on training word embeddings with neural networks that result in low-dimensional dense vector representations of words [5] by predicting target words from context (or vice versa). These high quality word embeddings also have the desired property of reflecting semantic similarity in vector space (semantic similarities can be captured by vector arithmetics). Expanding on this idea of word embeddings, [4] trained embeddings for sentences or longer units, which they denote "paragraph vectors". These paragraph vectors have the similar properties of reflecting semantic similarity above the word level, and is shown to be highly effective in a series of document and sentiment classification tasks.

5.2 Document Classification in JLC

To characterize the strength of the association between the lyrics text and its associated *banshi*, we define a document classification task: to what degree can we use textual features extracted from the lyrics to classify the documents in the PLBS and PLSQBS data sets into one of the four BS or seven SQBS classes?

[11] In the tf-idf (term frequency - inverse document frequency) weighting scheme [9], the term frequency count is compared to an inverse document frequency count, which measures the number of occurrences of a word in the entire corpus. Thus the tf-idf transforms the document into a weighted vector that assigns higher value to terms that have high occurrences in a small number of documents.

We use several varieties of vector representation for documents in our classification experiment, as described above: (1) BOW; (2) tf-idf BOW; (3) Paragraph Vectors (D2V); (4) document-topic distribution from the topic models (TM) we derived in Section 4. The document-topic distribution can be seen as a very low-dimension representation of a document, which has been shown to perform well in document classification tasks in place of BOW representation [1]. For Paragraph Vectors, we train document embeddings on our PL* data sets using the `Doc2Vec` (D2V) implementation available in the Python library `gensim`. The resulting embeddings has 100 dimensions for each document. We use Support Vector Machine (SVM) with RBF kernel for all classification experiments.

5.3 Document Classification Results

For the PLSQBS and PLBS data sets, we present the document classification accuracy [12] in Table 5. We observe that document classification accuracy scores are significantly above chance in both data sets (chance being $1/7 == 0.14$ in PLSQBS data set and $1/4 == 0.25$ in PLBS data set), with best accuracy scores of 0.41 and 0.53. This indicates that supervised learning is able to capture the important features that distinguish the different *banshi* or *shengqiang-banshi* classes, which in these particular JLC data sets, are somewhat unintuitive even for human judgments [13].

Contrasting the performance of different features, we note that tf-idf is more effective than BOW, as expected. The D2V Paragraph Vectors achieves comparable or lower results with the tf-idf (200 times higher in dimension). This is somewhat unexpected since these Paragraph Vectors are supposed to be a higher quality vector representation that captures both semantic similarity and word order that is absent from BOW representations such as tf-idf [4]. We attribute this under-performance of D2V to the smaller amount of training data available in the PL* data sets (comparing to much larger general-domain training corpora used in literature for D2V). In the mean time, we observe that the topic models features are ineffective at capturing the document-level distinctions.

6. CONCLUSION

In this work, we have introduced the Jingju Lyrics Collection, a comprehensive data collection of jingju lyrics enriched and re-structured with several extracted datasets based on musicological considerations. Utilizing this data, we performed topic modeling in order to explore the topic structures of the jingju lyrics as related to different *banshi* and SQBS types. The results show that while the topics are in general reasonable, the distinctions between

Dataset	Feature	Accuracy
PLBS	BOW (20000)	0.481
PLBS	tf-idf (20000)	0.527
PLBS	**D2V** (100)	**0.528**
PLBS	TM (20)	0.274
PLSQBS	BOW (20000)	0.356
PLSQBS	**tf-idf** (20000)	**0.408**
PLSQBS	D2V (100)	0.347
PLSQBS	TM (20)	0.112

Table 5. Document classification with SVM RBF kernel, with dimensionality of features shown in parenthesis

different *banshi* are less contrastive than between different *shengqiang-banshi* [14]. Document classification experiments are carried out to further understand the association within a supervised setting. The strong results in document classification support the associations between the expressive functions (as expressed in the lyrics) and the *banshi* or *shengqiang-banshi* (SQBS) categories.

We observe that unexpectedly, neither D2V Paragraph Vectors nor the topic models are more effective at document classification than the high-dimension tf-idf vectors. We postulate that these have several implications (even though our goal and contribution in this work do not lie in the use of these more advanced representations). First, it shows that latent topics may not be the most effective way to capture the different expressive functions in different *banshi* types (as opposed to, e.g., sentiment). It maybe of interest to perform feature and error analysis to understand what components of the document classification have made it more effective (e.g., sentiment polarity words, etc). Second, in addition to the training size problem discussed in Section 5.3, we attribute lower performance of D2V/TM to the potential errors in the NLP pipeline applied to the corpus, especially the Chinese segmentation (as already mentioned in Section 3.2). This includes two aspects: first, the automatic segmentation may introduce errors even for standard Chinese text; second, the language of jingju falls somewhere between modern and archaic Chinese, making it more challenging to segment automatically using a standard segmenter trained on modern language. Our on-going and future work, therefore, includes making corrections to the segmentation in the JLC while keeping expanding the collection.

7. ACKNOWLEDGEMENTS

This research is funded by the European Research Council under the European Union's Seventh Framework Program (FP7/2007- 2013), as part of the CompMusic project (ERC grant agreement 267583).

[12] The datasets are well balanced in their class sizes therefore accuracy is an appropriate measure.

[13] Here we are referring to a layperson who is a native speaker of Chinese but may not be a jingju expert. We are yet to evaluate this task on jingju experts.

[14] Our main goal in this paper is to investigate the expressive functions of *banshi*. Even though the result shows *shengqiang*'s importance in distinguishing positive from negative sentiments, we note that *shengqiang-banshi* is still a unique form combining both *shengqiang* and *banshi*.

8. REFERENCES

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2012.

[2] Cao B.. *Jingju changqiang banshi jiedu (Deciphering banshi in jingju singing)*. Renmin yinyue chubanshe, Beijing, 2010.

[3] Liu J.. *Jingju yinyue gailun(Introduction to jingju music)*. Renmin yinyue chubanshe, Beijing, 1998.

[4] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196, 2014.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119, 2013.

[6] Rafael Caro Repetto, Rong Gong, Nadine Kroher, and Xavier Serra. Comparison of the singing style of two jingju schools. In *16th International Society for Music Information Retrieval (ISMIR) Conference*, pages 507–513, Málaga, Spain, 26/10/2015 2015.

[7] Rafael Caro Repetto and Xavier Serra. Creating a corpus of jingju (beijing opera) music and possibilities for melodic analysis. In *15th International Society for Music Information Retrieval Conference*, pages 313–318, Taipei, Taiwan, 27/10/2014 2014.

[8] Rafael Caro Repetto and Xavier Serra. Melodic transformation processes in the arrangements of jingju banshi. In *Fourth International Conference On Analytical Approaches To World Music (AAWM 2016)*, New York, USA, 08/06/2016 2016.

[9] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[10] Huihsin Tseng. A conditional random field word segmenter. In *In Fourth SIGHAN Workshop on Chinese Language Processing*, 2005.

[11] Elizabeth Wichmann. *Listening to theatre: the aural dimension of Beijing opera*. University of Hawaii Press, 1991.

[12] Shuo Zhang, Rafael Caro Repetto, and Xavier Serra. Study of the similarity between linguistic tones and melodic pitch contours in beijing opera singing. In *15th International Society for Music Information Retrieval Conference*, pages 343–348, Taipei, Taiwan, 27/10/2014 2014.

[13] Shuo Zhang, Rafael Caro Repetto, and Xavier Serra. Predicting pairwise pitch contour relations based on linguistic tone information in beijing opera singing. In *16th International Society for Music Information Retrieval (ISMIR) Conference*, pages 107–113, Malaga, Spain, 26/10/2015 2015.

[14] Zhang Z. . *Jingju chuantongxi pihuang changqiang jiegou fenxi (Structural analysis of pihuang singing in jingju traditional plays)*. Renmin yinyue chubanshe, Beijing, 1981.

[15] Jiang J. . *Zhongguo xiqu yinyue (Music of Chinese traditional opera)*. Renmin yinyue chubanshe, Beijing, 2000.

Poster Session 2

A METRIC FOR MUSIC NOTATION TRANSCRIPTION ACCURACY

Andrea Cogliati
University of Rochester
Electrical and Computer Engineering
andrea.cogliati@rochester.edu

Zhiyao Duan
University of Rochester
Electrical and Computer Engineering
zhiyao.duan@rochester.edu

ABSTRACT

Automatic music transcription aims at transcribing musical performances into music notation. However, most existing transcription systems only focus on parametric transcription, i.e., they output a symbolic representation in absolute terms, showing frequency and absolute time (e.g., a piano-roll representation), but not in musical terms, with spelling distinctions (e.g., A♭ versus G♯) and quantized meter. Recent attempts at producing full music notation output have been hindered by the lack of an objective metric to measure the adherence of the results to the ground truth music score, and had to rely on time-consuming human evaluation by music theorists. In this paper, we propose an edit distance, similar to the Levenshtein Distance used for measuring the difference between two sequences, typically strings of characters. The metric treats a music score as a sequence of sets of musical objects, ordered by their onsets. The metric reports the differences between two music scores based on twelve aspects: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment. We also apply a linear regression model to the metric in order to predict human evaluations on a dataset of short music excerpts automatically transcribed into music notation.

1. INTRODUCTION

Automatic Music Transcription (AMT) is the process of inferring a symbolic representation of a musical performance. Despite four decades of active research, AMT is still an open problem, with humans being able to achieve better results than machines [2]. AMT systems can be broadly classified into two categories according to the chosen symbolic representation: parametric transcription and music notation transcription. Parametric transcription systems output a parametric representation of the musical performance, such as an unquantized MIDI pianoroll [14]. This representation is expressed in physical terms, such as seconds for note onset and duration, and hertz or MIDI numbers for pitch [7]. It can faithfully represent the musical performance, but normally it does not explicitly encode high-level musical structures, such as key, meter and voicing [21]. Music notation transcription systems, on the other hand, output a common music notation that human musicians read. This representation is expressed in musically meaningful terms, such as quantized meter for note onset and duration, and spelling distinctions (e.g., A♭ versus G♯) for pitch. Compared to parametric transcription, music notation transcription is generally more desirable for many applications connecting humans and machines, such as computational musicological analysis and music tutoring systems. The vast majority of existing AMT methods, however, are parametric transcription systems.

Researchers have put considerable effort toward building music notation transcription systems by identifying musical structures from unquantized parametric representations, especially MIDI files, from both MIR and cognitive perspectives [20]. Cambouropoulos [3] described the key components necessary to convert a MIDI performance into music notation: identification of elementary musical objects (i.e., chords, arpeggiated chords, and trills), beat identification and tracking, time quantization and pitch spelling. Takeda et al. [18] describe a Hidden Markov Model (HMM) for the automatic transcription of monophonic MIDI performances. Cemgil [4] presents a Bayesian framework for music transcription, identifying some issues related to automatic music typesetting (i.e., the automatic rendering of a musical score from a symbolic representation), in particular tempo quantization, and chord and melody identification. Karydis et al. [12] proposed a perceptually motivated model for voice separation capable of grouping polyphonic groups of notes, such as chords or other forms of accompaniment figures, into a perceptual stream. A more recent paper by Grohganz et al. [11] introduced the concepts of score-informed MIDI file (S-MIDI), in which musical tempo and beats are properly represented, and performed MIDI file (P-MIDI), which records a performance in absolute time. The paper also presented a procedure to approximate an S-MIDI file from a P-MIDI file – that is, to detect the beats and the meter implied in the P-MIDI file, starting from a tempogram then analyzing the beat inconsistency with a salience function based on autocorrelation.

Researchers have also attempted to infer musical structures directly from audio. Ochiai et al. [16] proposed a model for the joint estimation of note pitches, onsets, offsets and beats based on Non-negative Matrix Factorization

(NMF) constrained with a rhythmic structure modeled with a Gaussian mixture model. Collins et al. [8] proposed a model for multiple fundamental frequency estimation, beat tracking, quantization, and pattern discovery. The pitches are estimated with a neural network. An HMM is separately used for beat tracking. The results are then combined to quantize the notes. Note spelling is performed by estimating the key of the piece and assigning to MIDI notes the most probable pitch class given the key.

An immediate problem arising when building a music notation transcription system by incorporating the above-mentioned musical structure inference methods is to find an appropriate way to evaluate the transcription accuracy of the system. In our prior work [7], we asked music theorists to evaluate music notation transcriptions along three different musical aspects, i.e., the pitch notation, the rhythm notation, and the note positioning. However, subjective evaluation is time consuming and difficult to scale to provide enough feedback to further improve the transcription system. It would be very helpful to have an objective metric for music notation transcription, just like the standard metric F-measure for parametric transcription [1]. Considering the inherent complexity of music notation, such a metric would need to take into account all of the aspects of the high-level musical structures in the notation. To the best of our knowledge, there is no such metric, and the goal of this paper is to propose such a metric.

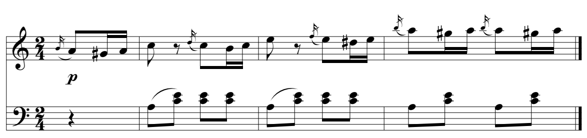
Specifically, in this paper we propose an edit distance, based on similar metrics used in bioinformatics and linguistics, to compare a music transcription with the ground-truth score. The design of the metric was guided by a data-driven approach, and by simplicity. The metric is calculated in two stages. In the first stage, the two scores are aligned based on the pitch content; in the second stage, the differences between the two scores are accumulated, taking into account twelve different aspects of music notation: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment. This will serve the same purpose as F-measure in evaluating parametric transcription. To validate the saliency and the usefulness of this metric we also apply a linear regression model to the errors measured by the metric to predict human evaluations of transcriptions.

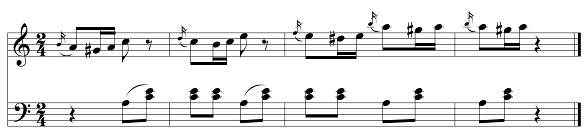
2. BACKGROUND

Approximate sequence comparison is a typical problem in bioinformatics [13], linguistics, information retrieval, and computational biology [15]. Its purpose is to find similarities and differences between two or more sequences of elements or characters. The sequences are assumed sufficiently similar but potentially corrupted by errors. Possible differences include the presence of different elements, missing elements or extra elements. Several metrics have been proposed to measure the distance between two sequences, including the family of edit metrics [15], and gap-penalizing alignment techniques [13].

A music score in traditional Western notation can be viewed as a sequence of musical characters, such as clefs, time and key signatures, notes and rests, possibly occurring concurrently, such as in simultaneous notes or chords. Transcription errors include alignment errors due to wrong meter estimation or quantization, extra or missing notes and rests, note and rest duration errors, wrong note spelling, wrong staff assignment, wrong note grouping and beaming, and wrong stem direction. All of these errors contribute to a various degree to the quality of the resulting transcription. However, the impact of each error and error category has not, to the best of our knowledge, been researched.

As an example, Fig. 1 shows two transcriptions of the same piece. Both transcriptions contain similar errors, i.e., wrong meter detection, but the transcription in Fig. 1c is arguably worse than that in Fig. 1b. A similar problem can be observed with the standard F-measure typically used to evaluate parametric transcriptions [1]; while the metric is objective and widely used, the impact of different errors on the perceptual quality of a transcription has not been researched. Intuitively, certain errors, such as extra notes outside of the harmony, should be perceptually more objectionable than others, such as octave errors. This is the reason for both proposing an objective metric and correlating the metric with human evaluations of transcriptions.


(a) Ground truth


(b) Transcription with a wrong pickup measure

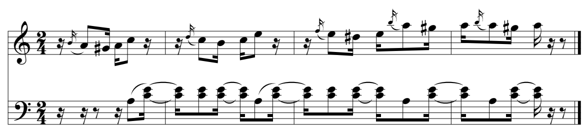

(c) Transcription off by a 16th note

Figure 1: Comparison of two transcriptions of the same piece containing similar errors but with different readability.

3. PROPOSED METHOD

The proposed metric is calculated in two stages: in the first stage, the transcription is aligned with the ground-truth music notation based on its pitch content only, i.e., all of the other objects, such as rests, barlines, and time and key signatures are ignored; in the second stage, all of the objects occurring at the aligned portions of the scores

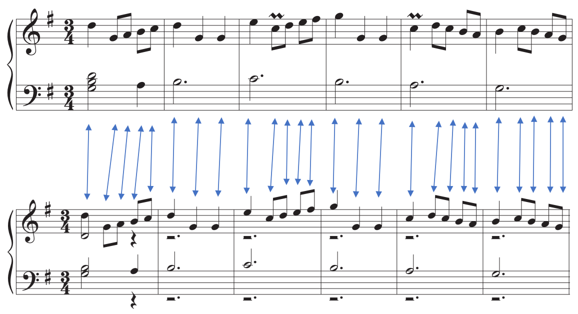


Figure 2: Alignment between the ground-truth (top) and a transcription (bottom) of Bach's Minuet in G. Arrows indicate aligned beats.

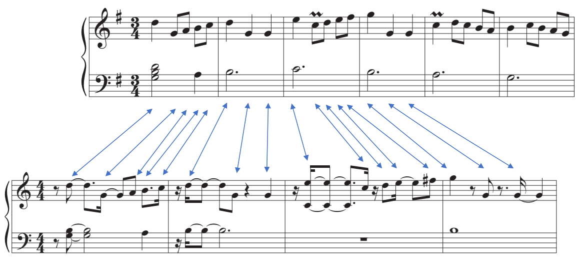


Figure 3: Alignment between the ground-truth (top) and another transcription (bottom) of Bach's Minuet in G. Arrows indicate aligned beats.

are grouped together and compared. The metric reports the differences in aligned portions in terms of twelve aspects: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment.

Some algorithms to efficiently calculate certain edit distances, e.g., the Wagner-Fischer algorithm to calculate the Levenshtein distance between two strings, are able to align two sequences and calculate the edit costs in a single stage. We initially tried to apply the same strategy to our problem, but we discovered that the algorithm was not sufficiently robust, especially with transcriptions highly corrupted by wrong meter estimation. Intuitively, notes are the most salient aspects of music, so it is arguable that the alignment of two transcriptions should be based primarily on that aspect, while the overall quality of the transcription should be judged on a variety of other aspects.

The ground truth and the transcription are both encoded in MusicXML, a standard format to share sheet music files between applications [10]. The two scores are aligned using Dynamic Time Warping [17]. The local distance is simply the number of mismatching pitches, regardless of duration, spelling and staff positioning.

To illustrate the purpose of the initial alignment, we show two examples in Fig. 2 and Fig. 3. The alignment stage outputs a list of pairs of aligned beats. Fig. 2 shows the alignment of a fairly good transcription of Bach's Minuet in G from the Notebook for Anna Magdalena Bach, with the ground truth, which corresponds to the following sequence, expressed in beats, numbered as quarter notes starting from 0 (GT is ground truth, T is transcription):

GT	0.0	1.0	1.5	2.0	2.5	3.0	4.0
T	0.0	1.0	1.5	2.0	2.5	3.0	4.0
4.0	5.0	6.0	7.0	7.5	8.0	8.5	9.0
5.0	5.0	6.0	7.0	7.5	8.0	8.5	9.0
10.0	10.0	11.0	12.0	13.0	13.5	14.0	14.5
10.0	11.0	11.0	12.0	13.0	13.5	14.0	14.5
15.0	16.0	16.5	17.0	17.5			
15.0	16.0	16.5	17.0	17.5			

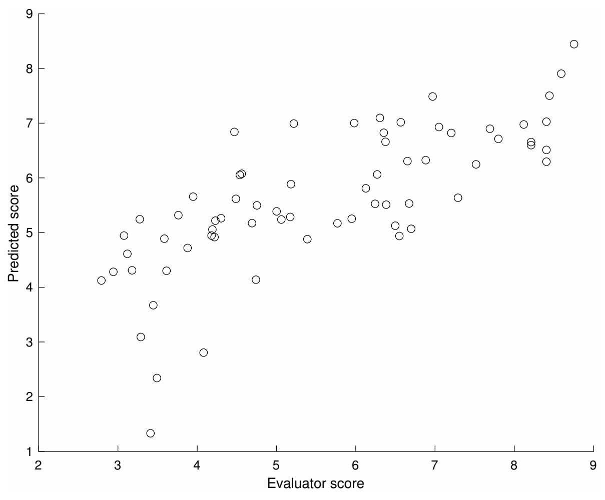
In this case, since the transcription is properly aligned with the ground truth, the sequence is just a list of all equal numbers, one for each onset of the notes in the score. However, beat 4.0 in the ground truth is matched with beats 4.0 and 5.0 in the transcription; the same happens for beats 10.0 and 11.0, so DTW cannot properly distinguish repeated pitches. Only one alignment is shown in the figure for clarity.

Fig. 3 shows an example of an alignment for a badly aligned transcription of the same piece. The corresponding sequence is the following:

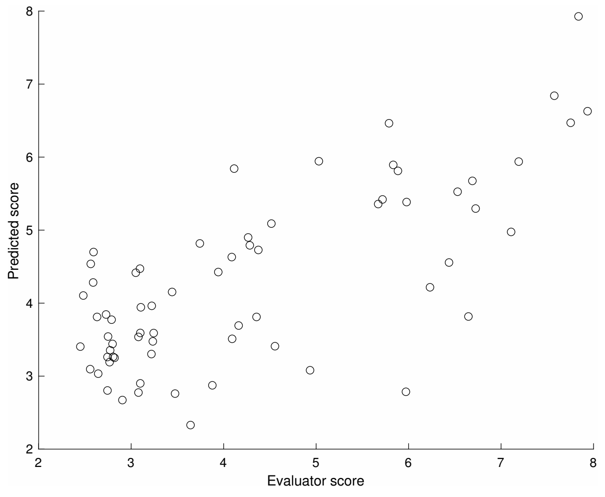
GT	0.0	0.0	0.0	1.0	1.0	1.5
T	0.0	0.5	1.0	1.75	2.0	2.5
2.0	2.5	3.0	3.0	3.0	4.0	4.0
3.0	3.75	4.25	4.5	5.0	5.5	7.0
5.0	6.0	6.0	6.0	7.0	7.5	8.0
7.0	8.25	8.5	9.0	9.75	10.25	10.75
8.0	8.5	9.0	10.0	10.0	10.0	11.0
11.0	11.5	12.0	13.5	14.75	15.0	15.0

In this case, multiple beats in the transcription correspond to the same beat in the ground truth, e.g., beat 1.0 in the ground truth corresponds to beats 1.75 and 2.0 in the transcription, because a single note in the ground truth has been transcribed as two tied notes. Only one alignment is shown in the figure for clarity.

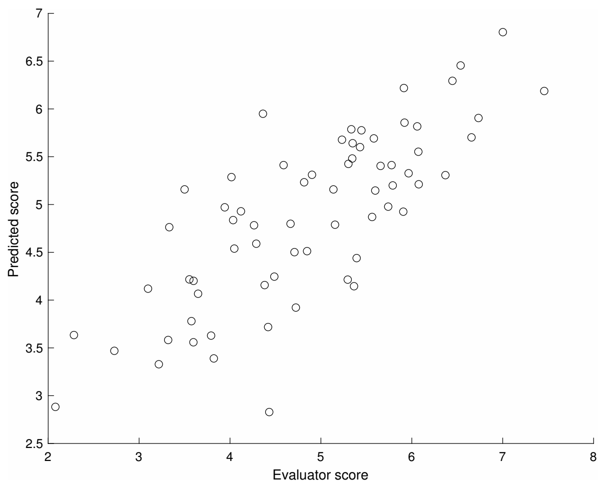
To calculate the distance between the two aligned scores, we proceed by first grouping all of the musical objects occurring inside aligned portions of the two scores into sets, thus losing the relative location of the objects within each set but preserving all of the other aspects, including staff assignment. Then the aligned sets are compared, and the differences between the two sets are reported separately. The following aspects only allow binary matching: barlines, clefs, key signatures, and time signatures. Rests are matched for duration and staff assignment, i.e., a rest with the correct duration but on the wrong staff will be considered a staff assignment error, a rest with the correct staff assignment but wrong duration will be considered a rest duration error. A missing or an extra rest will be considered a rest error. Notes are matched for spelling, duration, stem direction, staff assignment, and grouping into chords. For groupings, we only report the absolute value of the difference between the number of chords present in the two sets. The metric does not distinguish missing or



(a) Pitch Notation



(b) Rhythm Notation



(c) Note Positioning

Figure 4: Correlation between the predicted ratings and the average human evaluator ratings of all of the transcriptions in the dataset.

extra elements. These choices were dictated by simplicity of design and implementation.

All of the errors are cumulated for all of the matching sets. The errors for barlines, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment are then normalized by dividing the total number of errors for each aspect by the total number of musical objects taken into account in the score. This step is necessary to normalize the number of errors for pieces of different lengths. The errors for clefs, key signatures, and time signatures are not normalized, as they are typically global aspects of the scores, and not influenced by the length of the piece. This might be a limitation for pieces with frequent changes in key signature or time signature.

As an example, the set of objects at the first beat of the first measure of Fig. 2 include the initial barlines, clefs, time signature, key signature, and notes starting on the downbeat of the measure. Barlines, clefs, time signature, and key signature are all correctly matched. All of the notes are correct in pitch, spelling and duration, however there are two errors in stem direction, one error in grouping, and one error in staff assignment. All of the rests are considered rest errors at each respective onsets.

For the first beat of the first measure of Fig. 3, all of the elements of the transcription till the first transcribed notes (the three notes pointed by the first arrow) and the notes tied to them will be considered as part of the same set. The wrong key signature and time signature will be reported as errors. The two eight rests will be reported as rest errors. The three notes in the transcription are properly spelled, but their duration is wrong, so that will be counted as three note duration errors. The missing D from the chord will be reported as a note error. The extra tied notes will be reported as note errors as well.

In summary, the following twelve normalized error counts are calculated by the metric: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment. In order to translate these error counts into a musically relevant evaluation, we propose to use linear regression of the twelve error counts to fit human ratings of three musical aspects of automatic transcriptions, i.e., the pitch notation, the rhythm notation, and the note positioning. For each aspect, the linear regression learns twelve weights, one for each of the normalized error counts, to fit the human ratings. These weights can then be used to predict the human ratings of other music notation transcriptions.

4. EXPERIMENTAL RESULTS

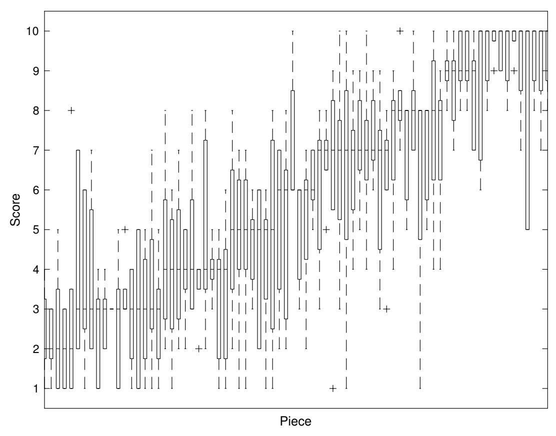
To evaluate the proposed approach, we calculate the normalized error count and run linear regression to fit human ratings of 19 short music excerpts collected in our prior work [7]. These music excerpts were from the Kostka-Payne music theory book, all of them piano pieces by well-known composers, and were performed on a MIDI keyboard by a semi-professional piano player. These excerpts were then transcribed into music notation using four different methods: a novel method proposed in the paper (which will be referred to as CDT), MuseScore, GarageBand and Finale. For each transcription, the human evaluators were asked to assign a numerical rating between 1 and 10 for three musical aspects, i.e., the pitch notation, the rhythm notation, and the note positioning.

The proposed method of calculating the error counts uses MusicXML [10], the de facto standard for sharing sheet music files between applications, as the format of music notation. Two of the methods evaluated in the paper (Finale and MuseScore) can output the scores into MusicXML. For GarageBand, CDT and the ground truth, however, MusicXML was not available or was difficult to output automatically. We had to manually convert the scores into MusicXML. The transcribed scores are named with the initial of the transcription method and a number indicating the excerpt. So, `M-8.mxl` represents the eight excerpt transcribed with MuseScore. The letter K, for Kostka-Payne, indicates the ground truth scores. This dataset and a Python implementation of the proposed approach are available at `http://www.ece.rochester.edu/~acogliat/repository.html`. The implementation uses the music21 toolkit [9] for parsing the MusicXML files and processing the imported scores. The implementation has been tested with music21 V3.1.0.

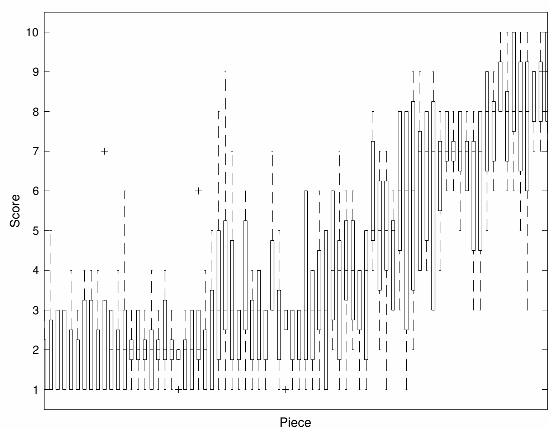
In order to validate the quality of the prediction we calculated the coefficient of determination R^2, which is the square of the Pearson correlation coefficient. The R^2 was 0.558 for the pitch notation correlation, 0.534 for the rhythm notation, and 0.601 for note positioning. These results are reflected in Fig. 4; the proposed metric fits the data adequately, in general, even though the correlation is not perfect. It can also be noted that the prediction of the score for note positioning is the best, while the prediction of the score for rhythm notation is the worst.

To understand the underlying causes of the covariance we firstly analyzed the ratings given by the human evaluators. As we can see from Fig. 5, the human evaluators were oftentimes in disagreement among themselves. It must also be noted that in our prior work [7], the human annotators were not given exact instructions on what features to consider for the evaluation, so a considerable amount of subjectivity and judgment calls were likely to be present in the ratings.

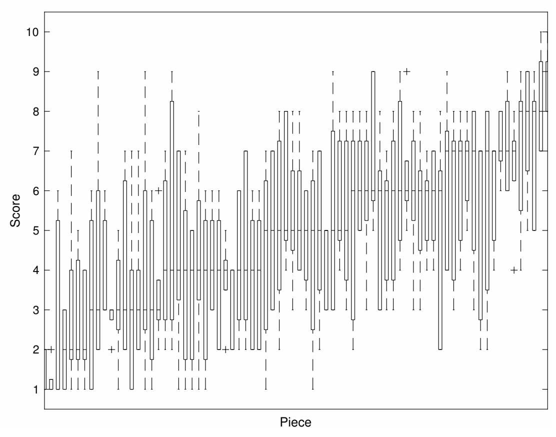
We also analyzed two transcriptions with the largest deviation from the predicted ratings, i.e., one transcription with a high predicted rating and a low human rating, and one transcription with a low predicted rating and a high human rating. The largest positive deviation occurred for the rhythm notation of transcription M-1, for which the proposed metric predicted a rating of 2.78, while the average human rating was 5.98. If we compare the transcription with the ground truth in Fig. 6 we can see that MuseScore misinterpreted the meter, causing the proposed metric to report a large number of note duration errors and barline errors, which resulted in a low rating. Human annotators, on the other side, likely penalized the meter error only once



(a) Pitch Notation



(b) Rhythm Notation

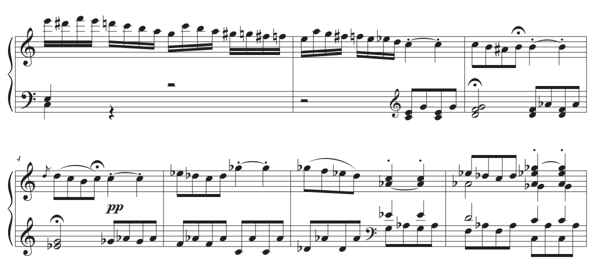


(c) Note Positioning

Figure 5: Distributions of the human ratings of the 76 transcriptions contained in the dataset. Each boxplot represents the ratings from 5 human evaluators.

globally, but still considered the transcription acceptable overall.

The largest negative deviation occurred for the pitch no-


(a) Ground Truth

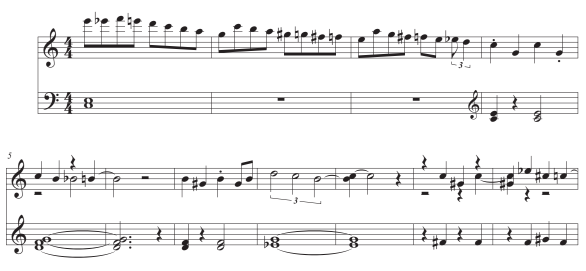
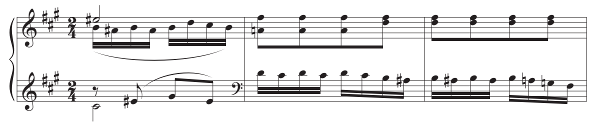

(b) M-1

Figure 6: Transcription of the first excerpt in the dataset by MuseScore, which shows the largest positive difference between the average human rating and the predicted rating, that is a high predicted rating and a low human rating. This evaluation difference occurs on the rhythm notation.


(a) Ground Truth

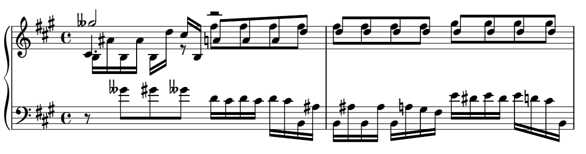

(b) C-13

Figure 7: Transcription of the thirteenth excerpt in the dataset by CDT, which shows the largest negative deviation between the average human rating and the predicted rating on rhythm notation, that is a low predicted rating and a high human rating. This evaluation difference occurs on the pitch notation.

tation of transcription C-13, for which the proposed metric predicted a rating of 6.83, while the annotators assigned an average score of of 4.48. If we compare the transcription with the ground truth in Fig. 7, we can notice that CDT makes a single mistake in notating the pitches, i.e., G𝄫 instead of E♯. It also makes a systematic error notating all Bs one octave lower. Finally, not grouping the eight notes in the treble staff makes the transcription hard to read. Possibly, the human annotators penalized the transcription because of its poor readability.

5. CONCLUSION AND FUTURE WORK

In this paper we proposed an objective metric to measure the differences between music notation transcriptions and the ground truth score. The metric is calculated by first aligning the pitch content of the transcription and the ground-truth music notation, and then counting the differences in twelve key musical aspects: barlines, clefs, key signatures, time signatures, notes, note spelling, note durations, stem directions, groupings, rests, rest duration, and staff assignment. We then used linear regression to predict human evaluator ratings along three aspects of music notation, namely, pitch notation, rhythm notation, and note positioning, from the error counts. Experiments show a clear correlation between the predicted ratings and the average human ratings, even though the correlation is not perfect.

One issue with the prediction is the high variance of the evaluator ratings, which likely originates from the inherent subjectivity of the tasks. Another issue of the proposed metric is that it does not incorporate music theory knowledge, such as the method proposed by Temperley to evaluate metrical models [19].

The current experiments were conducted on music notation transcriptions of human performances recorded on a MIDI keyboard; as a consequence, the transcriptions do not contain the errors commonly observed in audio-to-MIDI conversion processes, such as octave errors and extra or missing notes [5,6]. More research is necessary to evaluate the performance of the proposed method in the presence of such errors. In addition, the excerpts in the dataset were very short, compared to real piano pieces, so additional research is necessary to assess the robustness of the metric, and its computational complexity on longer pieces.

A Python implementation of the proposed approach, along with the dataset, is available at `http://www.ece.rochester.edu/~acogliat/repository.html`. This implementation can be used to calculate the twelve error counts as well as to predict human ratings on the three musical aspects of a music notation transcription.

6. REFERENCES

[1] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *Proc. of International Society for Music Information Retrieval (ISMIR)*, pages 315–320, 2009.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] Emilios Cambouropoulos. From MIDI to traditional musical notation. In *Proc. of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Mod-*

els for Composition, Performance and Analysis, volume 30, 2000.

[4] Ali Taylan Cemgil. *Bayesian music transcription*. PhD thesis, Radboud University Nijmegen, 2004.

[5] Andrea Cogliati and Zhiyao Duan. Piano Music Transcription Modeling Note Temporal Evolution. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 429–433, Brisbane, Australia, 4 2015. IEEE.

[6] Andrea Cogliati, Zhiyao Duan, and Brendt Wohlberg. Piano Transcription with Convolutional Sparse Lateral Inhibition. *IEEE Signal Processing Letters*, 24(4):392–396, 2017.

[7] Andrea Cogliati, David Temperley, and Zhiyao Duan. Transcribing human piano performances into music notation. In *Proc. of International Society for Music Information Retrieval (ISMIR)*, 2016.

[8] Tom Collins, Sebastian Böck, Florian Krebs, and Gerhard Widmer. Bridging the audio-symbolic gap: The discovery of repeated note content directly from polyphonic music audio. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, 2014.

[9] Michael Scott Cuthbert and Christopher Ariza. music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data. In *Proc. of International Society for Music Information Retrieval (ISMIR)*, 2010.

[10] Michael Good. MusicXML for notation and analysis. *The virtual score: representation, retrieval, restoration*, 12:113–124, 2001.

[11] Harald Grohganz, Michael Clausen, and Meinard Müller. Estimating Musical Time Information from Performed MIDI Files. In *Proc. of International Society for Music Information Retrieval (ISMIR)*, pages 35–40, 2014.

[12] Ioannis Karydis, Alexandros Nanopoulos, Apostolos Papadopoulos, Emilios Cambouropoulos, and Yannis Manolopoulos. Horizontal and vertical integration/segregation in auditory streaming: a voice separation algorithm for symbolic musical data. In *Proc. 4th Sound and Music Computing Conference (SMC2007)*, 2007.

[13] Jonathan M. Keith, editor. *Bioinformatics*, volume 1525 of *Methods in Molecular Biology*. Springer New York, New York, NY, 2017.

[14] Meinard Müller. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer, 2015.

[15] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 3 2001.

[16] Kazuki Ochiai, Hirokazu Kameoka, and Shigeki Sagayama. Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 133–136, 2012.

[17] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 2 1978.

[18] Haruto Takeda, Naoki Saito, Tomoshi Otsuki, Mitsuru Nakai, Hiroshi Shimodaira, and Shigeki Sagayama. Hidden Markov model for automatic transcription of MIDI signals. In *Multimedia Signal Processing, 2002 IEEE Workshop on*, pages 428–431, 2002.

[19] David Temperley. An Evaluation System for Metrical Models. *Computer Music Journal*, 2004.

[20] David Temperley. *Music and probability*. The MIT Press, 2007.

[21] David Temperley. A unified probabilistic model for polyphonic music analysis. *Journal of New Music Research*, 38(1):3–18, 2009.

A MULTIOBJECTIVE MUSIC RECOMMENDATION APPROACH FOR ASPECT-BASED DIVERSIFICATION

Ricardo S. Oliveira **Caio Nóbrega** **Leandro B. Marinho** **Nazareno Andrade**

UFCG - Federal University of Campina Grande, Brazil

{ricardooliveira, caionobrega}@copin.ufcg.edu.br, {lbmarinho, nazareno}@computacao.ufcg.edu.br

ABSTRACT

Many successful recommendation approaches are based on the optimization of some explicit utility function defined in terms of the misfit between the predicted and the actual items of the user. Although effective, this approach may lead to recommendations that are relevant but obvious and uninteresting. Many approaches investigate this problem by trying to avoid recommendation lists in which items are very similar to each other (aka diversification) with respect to some aspect of the item. However, users may have very different preferences concerning what aspects should be diversified and what should match their past/current preferences. In this paper we take this into consideration by proposing a solution based on multiobjective optimization for generating recommendation lists featuring the optimal balance between the aspects that should be held fixed (maximize similarity with users actual items) and the ones that should be diversified (minimize similarity with other items in the recommendation list). We evaluate our proposed approach on real data from Last.fm and demonstrate its effectiveness in contrast to state-of-the-art approaches.

1. INTRODUCTION

In scenarios of vast and dynamic availability of content, such as online music streaming services, users are quickly overloaded with a large and ever increasing space of choices. Recommender systems are successful tools for addressing this issue by modeling the preferences of users and anticipating their information needs. The most successful recommendation approaches are usually those based on the optimization of some explicit utility function defined in terms of the misfit between the predicted and the actual items consumed by the user. Although effective in many scenarios, the recommendation algorithms that optimize this kind of function are prone to deliver recommendations that are relevant but possibly uninteresting. For example, for a user who only listens to American punk rock bands from the 70's, a recommendation of more bands of this kind would probably be accurate, but possibly tedious given that this user may very likely be able to find these artists without aid.

In order to mitigate this problem, many approaches have appeared with the aim of increasing diversity in recommendations [6, 14, 17, 19]. This is usually achieved by mechanisms that avoid recommendation lists in which items are very similar to each other with respect to some aspect of the items (e.g. music genre). Such approaches can potentially increase users satisfaction by providing less obvious recommendations. However, users may have different preferences concerning what aspects should be diversified and what should match their past/current preferences. For example, a user may be very conservative concerning the music genres she likes to listen (e.g. Bossa Nova), but very open to discover how this genre is played across different countries (e.g. Bossa Nova played in Japan and India). This is exactly the recommendation scenario we investigate in this paper, i.e., we want to generate recommendation lists by explicitly holding one or more item aspects (e.g. Bossa Nova) constant, but increasing diversity in others (e.g. locality and time period). The aspects that are held fixed are the ones that correspond to the users past/current preferences while the others correspond to the way the users are open for diversification.

This problem has two different and possibly conflicting objectives that need to be optimized for each user's recommendation list: (i) find the items that maximize the similarity with the preferences of the user in terms of some set of selected aspects, and (ii) find the items that minimize the intra-list similarity (i.e. pairwise similarity of items in the recommendation list) regarding a different set of selected aspects. To generate recommendation lists that balance these two objectives we propose Multiobjective Aspect Diversification (MOAD), a recommendation approach that cast this problem as a multiobjective optimization problem. MOAD uses the Nondominated Sorting Genetic Algorithm - NSGA-II, which is an efficient solver for this kind of problem [5]. The main difference between our approach and other related work from the literature is that we allow the explicit choice of the aspects to diversify and hold constant.

Although music recommendation may refer to many distinct entities in the literature, such as song tracks, albums and artists, in this paper we focus on artists due to their relative abundance of aspects publicly available. In order to assess the effectiveness of our proposed approach, we collected artist listening historical data from Last.fm [1], a large online radio portal, and enriched this collection with artists metadata collected from Music Brainz [2] and DBPedia [3]. We conduct several experiments by considering several switches between the fixed and variable aspects and show that MOAD achieves the sought balance between both objectives. We also compare our approach with several diversification algorithms from the literature and show that our recommendations are more diverse and relevant in the chosen aspects in comparison to those.

2. PROBLEM FORMALIZATION

The problem that we address in this paper can be stated as follows: given a target user $u \in U$ (where U is the set of users), her item consumption history $I_u \subseteq I$ (where I is the set of items) [4], two disjoint sets X and Y of item aspects (possibly provided by the user), we want to find the top-n items that are more similar to I_u regarding X and more dissimilar from each other regarding Y.

Items may have different kinds of metadata associated to them, aka attributes, dimensions, contexts or side information. For example, if the recommendable item is a music artist, we can think of each possible music genre as a binary attribute. More formally, let G be the set of genres and $d : G \times I \to \{0, 1\}$ a function that indicates if genre $g \in G$ describes item $i \in I$. The set G is thus one aspect of the item and represents the set of attributes of the kind genre.

More generally, let $A = \{A_1, A_2 \ldots, A_t\}$ denote the set of t possible aspects and $f_j : A_j \times I \to \mathbb{R}$ be a feature extractor for any attribute $a \in A_j$ that describes item i. An item may now be represented as a feature vector regarding aspect j. For example, $\vec{i} = (f_1(a_1, i), f_1(a_2, i), \ldots, f_1(a_p, i))$ represents a feature vector of item i where $a_1, a_2, \ldots, a_p \in A_1$.

The input for the algorithm is a user $u \in U$, her consumption history I_u, the size n of the recommendation list and two disjoint sets of aspects: $X, Y \subseteq A$ where $X \cap Y = \emptyset$. We also consider two similarity functions: $g_a(R)$ that returns the intra-list similarity (cf. Section 4.4) of the items in the recommendation set R w.r.t aspect a; and $h_a(R, I_u)$ that returns the similarity between R and the user history I_u. Finally, let

$$\textit{diversity}(R, X) = 1 - \frac{1}{|X|} \sum_{a \in X} g_a(R) \tag{1}$$

denote the average intra-list distance for all aspects in X and

$$\textit{affinity}(R, Y) = \frac{1}{|Y|} \sum_{a \in Y} h_a(R, I_u) \tag{2}$$

[1] http://www.last.fm/

[2] https://musicbrainz.org/

[3] http://wiki.dbpedia.org/

[4] I_u can also be thought as a query in terms of MIR

the average similarity between R_u and I_u for all aspects in Y. Now, given the aforementioned inputs, we want to find a set $R \setminus I_u \subseteq I$ of n items (i.e. $|R| = n$) that maximize, at the same time, the objective functions defined in equations 1 and 2, i.e.:

$$\arg\max_R \, (\textit{diversity}(R, X), \textit{affinity}(R, Y)) \tag{3}$$

3. RELATED WORK

Several works have appeared in recent years proposing recommender systems concerned with other metrics beyond accuracy such as diversity, novelty or serendipity. Most of these works aim to maximize such an alternative metric without degrading accuracy. The seminal work of Ziegler et al. [19] laid the foundations for achieving diversity based on a re-ranking of collaborative filtering algorithms results. Several other works appeared following similar principles but based on different techniques such as graphs [8], machine learning [7, 18] and information retrieval [15].

Another strand of work considers this problem as a multiobjective optimization task. Realizing that accuracy, diversity and novelty might be conflicting objectives, Ribeiro et al. [11] proposed a hybrid recommendation system that combines algorithms through an evolutionary approach to maximize one objective, without sacrificing the others. Ouni et al. [9] proposed a genetic algorithm to recommend software libraries, finding a trade-off between three objectives. Wang et al. [16] developed a multiobjective solution to recommend accurate and unpopular items, called long tail recommendations. Zuo et al. [20] proposed personalized recommendations by balancing accuracy and diversification. And finally, Pampalk and Goto [10] proposed a graphic interface where users may adjust the recommendations received according to her desire by adjusting music aspects. Our work also use multiobjective evolutionary algorithms for promoting diversification, but differently from the aforementioned related works, we enable the explicit specification of the aspects that should be diversified.

4. MULTIOBJETIVE ASPECT DIVERSIFICATION

The main motivation for this research is to give users more control on their recommendations. We do this by letting the aspects that should be held constant and the ones that should be diversified user-definable. In this section we describe in detail the components of our approach.

4.1 Pareto Optimality

In multiobjective optimization the best solutions are the ones that cannot be improved in any of the objectives without degrading at least one of the other objectives. This property is known as *Pareto optimality*. In our case, a feasible solution $R \subseteq I$ (i.e. a recommendation list of size n) is said to dominate another solution $R' \subseteq I$ if:

1. $\textit{div}(R, X) \geq \textit{div}(R', X) \wedge \textit{aff}(R, Y) \geq \textit{aff}(R', Y)$

2. $\mathit{div}(R,X) > \mathit{div}(R',X) \lor \mathit{aff}(R,Y) > \mathit{aff}(R',Y)$

where *div* and *aff* are abbreviations for *diversity* and *affinity* respectively. A solution $R^* \subseteq I$ is called Pareto optimal if there is no other solution that dominates it. The set of Pareto optimal solution is also known as the Pareto front.

4.2 Evolutionary Algorithm Approach

Determining the Pareto front is known to be a computationally intensive task [2, 11]. In our case it basically requires an enumeration of all possible recommendation lists for the objectives evaluation (i.e. $\mathcal{O}(2^{|I|})$). Among the approaches used for addressing this kind of problem, evolutionary algorithms appear as the most efficient and used ones [4, 12, 16, 20]. Thus, we have decided to adopt a Multiobjective Evolutionary Algorithm (MOEAs) for addressing our research problem.

The idea is to generate a population of recommendation lists as individuals such that the dominating individuals are considered the fittest and are kept for the next generation. If the dominating individuals are insufficient to compose the new generation, some dominated individuals are chosen to compose the next generation. The crossing over - switching the items between neighbor individuals - and the mutation probability - used to replace some random items in individuals for items still not considered - allow new individuals to approach the Pareto front throughout several generations.

4.3 NSGA-II

Similarly to other related work, we have chosen the Non-dominated Sorting Genetic Algorithm II (NSGA-II) as the MOEA solution [9, 20]. Besides giving guarantees of convergence, it also offers a fast sorting function, called Fast Non-dominated Sorting, with $\mathcal{O}(MN^2)$ where M is the number of objectives and N is the population size. This sorting function separates individuals into levels of dominance. For individuals in the same level, NSGA-II estimates the density of solutions, privileging a set of solutions that are spread on the objective space, in a process called Crowding Distance Assignment.

Algorithm 1 presents NSGA-II pseudocode. The algorithm starts by creating an initial population of size N (line 2). The following steps are repeated for each generation. A new offspring is created based on the current population (line 4) and the individuals are ordered and selected to compose the population for the next generation (lines 5 to 11). This ordering considers first the selection of individuals whose objectives are not dominated by other individuals, made by *fast-nondominated-sort* (line 6), and second, the density of individuals provided by *crowding-distance-assignment* (line 8). If the non-dominated individuals are not enough to complete N, then individuals on the second level of dominance are chosen, and so on (lines 12 and 13). The population P_{t+1} is the output for the algorithm, and we select the individual with the greater sum of objective values as the final recommendation list.

Algorithm 1 NSGA-II

```
1: procedure NSGA-II($N, nGen, mProb, cProb$)
2:     $P_0 =$ create-initial-population$[N]$
3:     for t in 0 to nGen -1 do
4:         $Q_t =$ create-new-offspring$(P_t, mProb, cProb)$
5:         $R_t = P_t + Q_t$
6:         $F =$ fast-nondominated-sort$(R_t)$
7:         while $|P_{t+1}| + |F_i| \leqslant N$ do
8:             crowding-distance-assignment$(F_i)$
9:             $P_{t+1} = P_{t+1} \cup F_i$
10:            $i = i + 1$
11:        end while
12:        $Sort(F_i, \prec)$
13:        $P_{t+1} = P_{t+1} \cup F_i[N - |P_{t+1}|]$
14:    end for
15: end procedure
```

4.4 Item Representation and Similarity Metrics

In order to compute the objective functions defined in equations 1 and 2 we need to compute similarities between items concerning the sets of aspects used as input. Thus, we define feature extraction functions for each aspect such that similarity measures can be applied.

4.4.1 Aspects Definition

First we need to instantiate the set A of aspects that we consider in this paper:

- **Contemporaneity** (A_1)**:** refers to the year the artist was born (if the artist is solo) or the year the band was formed, in case the artist is a band.
- **Locality** (A_2)**:** refers often, but not always, to artist's birth/formation country.
- **Gender** (A_3)**:** refers to the artist gender (when applicable) together with its type (i.e. solo, band, orchestra, etc.). This aspect is a combination of two aspects where if the artist type is *person* (i.e. a solo artist) its gender is *male* or *female*. Otherwise, it has no gender but has a type that can be one of the following: *group*, *orchestra*, *choir*, *character* or *other*.
- **Music Genre** (A_4)**:** refers to the artist music genres.

We have chosen this aspects for two main reasons: (i) they are used recurrently in related works (not necessarily together) as side information for improving the preference modeling of users; and (ii) they are publicly available in MusicBraiz and DBpedia.

4.4.2 Similarity Metrics

Regarding A_1, each item is represented as one-dimensional vectors in which their single component is the year normalized to the range $[0, 1]$. More formally, for a given contemporaneity (i.e. year) $a \in A_1$ associated to artist i

$$f_1(a,i) = \frac{a - \min(A_1)}{\max(A_1) - \min(A_1)}$$

where $\min(A_1)$ and $\max(A_1)$ returns the minimum and maximum contemporaneity values of A_1 respectively. Now, the similarity of two items i and j regarding their respective contemporaneities $a_i, a_j \in A_1$ is simply computed as the inverse of their distance, i.e.,

$$sim_{A_1}(i,j) = 1 - (\vec{i} - \vec{j}) \tag{4}$$

where $\vec{i} = (f_1(a_i, i))$ and $\vec{j}$ is defined analogously. The intuition here is that artists from the same time epoch tend to produce similar music.

Concerning A_2, the feature extraction f_2 is basically an identity function, i.e., a function that returns the same value found in the raw data. So, similarly to A_1, items are represented as one-dimensional vectors whose single component is a nominal value (e.g. country name). For computing similarities between items under this representation we used the Occurrence Frequency (OF) metric [1] which besides being suitable for categorical data, exploits the frequency of items with regard to the associated features. This metric assigns 1 to items having the same feature value and different scores to mismatches. A mismatch between less frequent items regarding their features yields a lower value than a mismatch between more frequent items. For example, if we compare two artists from USA and England respectively, two countries with a large number of artists in the dataset, their similarity will be greater than artists of USA and Costa Rica, since Costa Rica has probably much less artists than USA. The idea is basically to avoid having zero similarities whenever a mismatch occurs.

The equation below defines OF (which is used as sim_{A_2}) of two items i, j regarding *Locality*:

$$OF_{A_2}(i,j) = \begin{cases} 1 & \text{if } \vec{i} = \vec{j} \\ \frac{1}{1+log\ \frac{|I|}{freq_{A_2}(i)} \times log\ \frac{|I|}{freq_{A_2}(j))}} & \text{otherwise.} \end{cases} \tag{5}$$

where $freq_{A_2}(i)$ returns the number of artists having the same feature value (country name in this case) as item i.

Regarding A_3, f_3 is analogous to f_2, but the item representation is slightly different. Since *Gender* is actually a combination of two aspects, items are represented by vectors containing two nominal values: (*type*,*gender*) where gender can be *male* or female if the artist type is *person*, and *neither* if the artist is associated to any other type. For calculating the similarity between items i, j we apply equation 5 separately for *type* and *gender* and take the average. More formally,

$$sim_{A_3}(i,j) = \frac{OF_{type}(i,j) + OF_{gender}(i,j)}{2} \tag{6}$$

As an example, the similarity between a male singer and a female singer should return a greater similarity than between a male singer and a band.

As an artist can be associated to multiple music genres, the feature extraction function for A_4 is the function $f_4 : A_4 \times I \rightarrow \{0,1\}$ that indicates the genres that are associated to a given artist. Each item is then represented by a binary vector of genres. To measure similarity between two items i, j regarding this aspect we use the well known cosine similarity function, i.e.,

$$sim_{A_4}(i,j) = cos(\vec{i}, \vec{j}). \tag{7}$$

Now, we can finally instantiate functions $g_a(\cdot)$ and $h_a(\cdot)$ introduced in section 2. For a given aspect $a \in A$, a recommendation list R and I_u:

$$g_a(R) = \sum_{(i,j)\in R\times R|i\neq j} sim_a(i,j) \tag{8}$$

$$h_a(R, I_u) = \sum_{(i,j)\in R\times I_u|i\neq j} sim_a(i,j) \tag{9}$$

5. EVALUATION

In this section we evaluate the effectiveness of MOAD for music recommendation. All code for the evaluation is available publicly online [5].

5.1 Data Collection and Preparation

We used three publicly available data sources: Last.fm, Music Brainz and DBpedia. Last.fm is a social network where users share data about their listening habits. In particular, we have used a recent Last.fm dataset published and made available by Schedl [13].

For extracting the aspects about the artists available in the Last.fm dataset, we have used Music Brainz, a music encyclopedia that provides rich metadata about artists and albums. From Music Brainz we extracted *Contemporaneity*, *Gender*, and *Locality*. Finally, we used DBpedia to extract the *Music Genre(s)* of each artist.

After enriching the artists with the aforementioned aspects, genres associated with less than 5 artists were removed, as well as artists with no genre at all. Finally, a sample of 1,000 users from the Last.fm dataset was randomly selected for the experiments. This number of users is in line with the size of other very well known and used Last.fm datasets in the music recommendation community, see for example the *Last-fm - 1K users dataset* [6] [3]. In our sample, 3 users had no history and were thus discarded. We also generated a train/test time split where the first 80% of artists listened by each user was used for training and the remaining 20% for testing.

Our sample includes the following statistics: 14,415 artists, a median of 140 artists listened per user, 10 genders, 437 localities, 847 genres, and contemporaneity ranging from 1212 to 2014. Figure 1 shows a flow chart summarizing our approach.

5.2 Evaluation Protocol and Metrics

Ideally, the aspects to keep and the ones to be diversified should be provided by the users themselves. Since this kind of online experiment can be very demanding, it will be left for future work. In this paper, we will simulate some

[5] https://github.com/ricooliveira/moad.git

[6] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html

possible scenarios and evaluate the extent to which MOAD can cope with them. The evaluation scenario is the following: one aspect is chosen to be diversified while the others are kept constant. Since we have four aspects, we end up with four evaluation scenarios. For example, a given user wants artist recommendations that are diverse regarding locality (artists countries) while maintaining genre, gender and contemporaneity constant (i.e. similar to previous listened artists w.r.t. these aspects).

Regarding evaluation metrics, we use *diversity* and *affinity* defined in equations 1 and 2 respectively. *Diversity* is actually related to a popular diversity metric known as ILD (Intra-List Diversity) [19] while *affinity* tries to assess the relevance of the recommendations regarding the aspects that were held constant. While *diversity* is only evaluated on the final recommendation list, *affinity* is evaluated on the test set.

#	Diversify	Maintain affinity
1	Cont.	Gender, Locality, Genre
2	Gender	Cont., Locality, Genre
3	Locality	Cont., Gender, Genre
4	Genre	Cont., Gender, Locality

Table 1. Evaluated recommendation scenarios

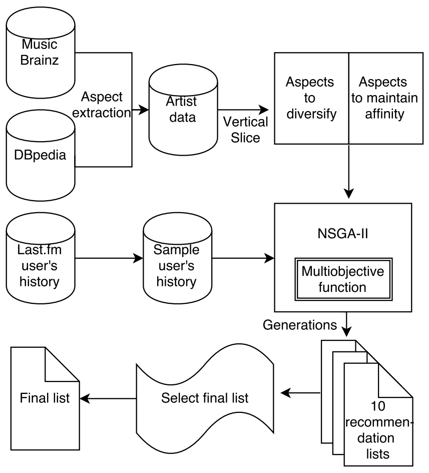


Figure 1. Flow chart of MOAD.

5.3 Baseline Algorithms

We have chosen baselines that are well known for promoting diversification without degrading accuracy. Since all of them compute item similarities in order to select the items that will compose the final recommendation list, we used the same similarity measures defined in section 4.4. This means that each baseline focused on the diversification of the same aspect, depending on the recommendation scenario chosen, as MOAD.

More specifically, we compare our approach to the following baselines:

- **Topic Diversification (TD):** Receives an initial recommendation list of 50 items where the first item of this list goes to the recommendation final list in order to preserve accuracy. Next, items are selected in an iterative and greedy fashion based on their rankings in the initial list and the similarity to the items already in the final list regarding the aspect of interest [19].
- **Relevance-based eXplicit Query Aspect Diversification (RxQUAD):** Performs a re-ranking over 50 precomputed items. During the greedy iterative step each item receives a score based on two factors: given an input aspect, the relevance of the aspect for the user and the relevance of the aspect for the item [15].
- **User-Based Collaborative Filtering (UBCF):** We also included a standard user-based collaborative filtering based on k-nearest neighbors. Notice that this algorithm is not aimed to promoting diversification.

We have used the RankSys tool where these three algorithm are implemented [14, 15].

5.4 Parameter Tuning

For determining suitable values to the NSGA-II parameters such as the number of generations, size of the population and probability of mutation, we extracted a subsample of 30 users from the Last.fm experimental dataset and determined a fixed scenario to perform some executions of our approach. We use the third scenario of Table 1 and $N = 10$, $nGen = 10$, $mProb = 0.1$ and $cProb = 0.9$ as default values. For tuning a particular parameter, we fixed the other parameters to its default values and varied the target parameter until no significant changes were found in the evaluation metrics. Due to the non-normality of the data, Wilcoxon non-parametric test was used. The tests determined that the ideal values to the parameters are $N = 10$, $nGen = 50$ and the default values to $mProb$ and $cProb$.

6. RESULTS AND DISCUSSION

For assessing MOAD variability across different executions, we run MOAD 10 times on scenario 3 of Table 1 and performed a Kruskal-Wallis test, which reported that there are no significant changes within the executions. We thus assume that other scenarios will follow a similar trend and thus only make one run of the algorithm for each subsequent scenario. The baseline algorithms are deterministic, so running them multiple times is not necessary.

For assessing the results we calculated *diversity* and *affinity* for all users in the experimental dataset. As mentioned earlier *affinity* was computed in the test set of each user. The boxplots of the results for all scenarios in Table 1 are shown in Figure 2. Notice that MOAD achieved better results than the baselines in all scenarios, considering both evaluation metrics, with a small variability across users. Wilcoxon tests are performed comparing MOAD to each baseline and all the differences are statistically significant albeit small in some cases.

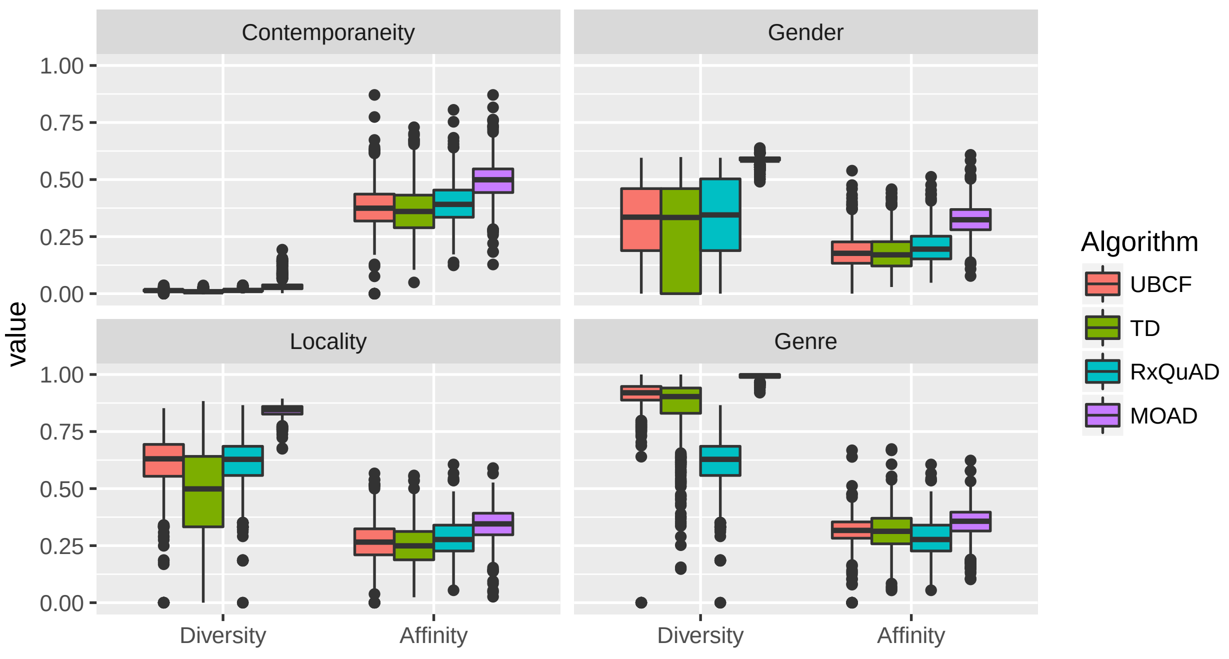


Figure 2. Comparison of MOAD and baselines

When diversifying *contemporaneity*, *diversity* shows very low results for all algorithms. This may be explained by the range of the aspect, mentioned on subsection 5.1, which turns the time difference, even between artists from different decades, very small when normalized. Gender is the scenario where the smallest differences between MOAD and the compared baselines are observed. A possible explanation to this is the fact that Gender aspect has only 10 possible values which does not leave much room for diversification. Locality and genre are the scenarios where we observed the highest gains, which is probably associated to the large number of possible values for these aspects.

Table 2 shows an example of a real Last.fm experimental user, receiving recommendations of three algorithms, based on scenario 3: UBCF; TD, the best baseline in scenario 3 and MOAD. In the simulations, MOAD obtained an improvement of 23.7% in diversity compared to TD. This means that MOAD may bring from 2 to 3 more artists from different countries than TD.

	Artist	**England**	**USA**	**Sweden**	**Iceland**	**Canada**	**Italia**	**India**	**Mexico**	**Norway**
UBCF	Pink Floyd	X								
	In Flames			X						
	Dream Theater		X							
	Iron Maiden	X								
	Megadeth		X							
	Coldplay	X								
	Björk				X					
	The Beatles	X								
	The Cure	X								
	Motörhead	X								
TD	Pink Floyd	X								
	Björk				X					
	Johnny Cash		X							
	In Flames			X						
	Clint Mansell	X								
	Zoë Keating					X				
	Dream Theater		X							
	Iron Maiden	X								
	Megadeth		X							
	Coldplay	X								
MOAD	Films of Colour	X								
	Ondskapt			X						
	Beautiful Sin									X
	I Ribelli						X			
	Planes Mistaken for Stars		X							
	Banda Machos								X	
	Jeff Healey					X				
	Cole Swindell		X							
	Mubarak Begum							X		
	Fuck Buttons	X								

Table 2. Top-10 recommendations for a real Last.fm user

7. CONCLUSIONS AND FUTURE WORK

In this paper we proposed MOAD, an approach for music recommendations that are at the same time diverse, regarding certain aspects, and similar to user preferences concerning other aspects. We cast this problem as a multi-objective optimization task and use an efficient algorithm based on evolutionary algorithms for solving it. We have defined specific similarity functions for each considered aspect and performed several simulations using real world data to assess MOAD performance. We have compared MOAD to other well known baselines from the literature and show that it provides better results in all evaluated scenarios.

As future work, we intend to run MOAD in all possible combinations of aspects to diversify and to hold constant and with more generations. We also intend to perform an online experiment with real users. Finally, we intend to approach the same problem under the perspective of MIR replacing the user's history by a set of input artists, allowing the user to discover new artists based on her instantaneous information needs.

Acknowledgement: This work was partially funded by the EU-BR BigSea project (MCTI/RNP 3rd Coordinated Call).

8. REFERENCES

[1] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 243–254. SIAM, 2008.

[2] Rocío L Cecchini, Carlos M Lorenzetti, Ana G Maguitman, and Nélida B Brignole. Multiobjective evolutionary algorithms for context-based search. *Journal of the Association for Information Science and Technology*, 61(6):1258–1274, 2010.

[3] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[4] Laizhong Cui, Peng Ou, Xianghua Fu, Zhenkun Wen, and Nan Lu. A novel multi-objective evolutionary algorithm for recommendation systems. *Journal of Parallel and Distributed Computing*, 2016.

[5] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[6] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. Adaptive diversity in recommender systems. In *IIR*, 2015.

[7] Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A Konstan, and Paul Schrater. I like to explore sometimes: Adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 19–26. ACM, 2015.

[8] Onur Küçüktunç, Erik Saule, Kamer Kaya, and Ümit V Çatalyürek. Diversified recommendation on graphs: pitfalls, measures, and algorithms. In *Proceedings of the 22nd international conference on World Wide Web*, pages 715–726. ACM, 2013.

[9] Ali Ouni, Raula Gaikovina Kula, Marouane Kessentini, Takashi Ishio, Daniel M German, and Katsuro Inoue. Search-based software library recommendation using multi-objective optimization. *Information and Software Technology*, 83:55–75, 2017.

[10] Elias Pampalk and Masataka Goto. Musicsun: A new approach to artist recommendation. In *ISMIR*, pages 101–104, 2007.

[11] Marco Tulio Ribeiro, Anisio Lacerda, Adriano Veloso, and Nivio Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 19–26. ACM, 2012.

[12] Marco Tulio Ribeiro, Nivio Ziviani, Edleno Silva De Moura, Itamar Hata, Anisio Lacerda, and Adriano Veloso. Multiobjective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):53, 2015.

[13] Markus Schedl. The lfm-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ICMR '16, pages 103–110, New York, NY, USA, 2016. ACM.

[14] Saul Vargas, Pablo Castells, and David Vallet. Intent-oriented diversity in recommender systems. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1211–1212, New York, NY, USA, 2011. ACM.

[15] Saúl Vargas, Pablo Castells, and David Vallet. Explicit relevance models in intent-oriented information retrieval diversification. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, pages 75–84, New York, NY, USA, 2012. ACM.

[16] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145–155, 2016.

[17] Mi Zhang and Neil Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 123–130. ACM, 2008.

[18] Yadong Zhu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Shuzi Niu. Learning for search result diversification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 293–302. ACM, 2014.

[19] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.

[20] Yi Zuo, Maoguo Gong, Jiulin Zeng, Lijia Ma, and Licheng Jiao. Personalized recommendation based on evolutionary multi-objective optimization [research frontier]. *IEEE Computational Intelligence Magazine*, 10(1):52–62, 2015.

A STUDY ON LSTM NETWORKS FOR POLYPHONIC MUSIC SEQUENCE MODELLING

Adrien Ycart and Emmanouil Benetos
Centre for Digital Music, Queen Mary University of London, UK
{a.ycart, emmanouil.benetos}@qmul.ac.uk

ABSTRACT

Neural networks, and especially long short-term memory networks (LSTM), have become increasingly popular for sequence modelling, be it in text, speech, or music. In this paper, we investigate the predictive power of simple LSTM networks for polyphonic MIDI sequences, using an empirical approach. Such systems can then be used as a music language model which, combined with an acoustic model, can improve automatic music transcription (AMT) performance. As a first step, we experiment with synthetic MIDI data, and we compare the results obtained in various settings, throughout the training process. In particular, we compare the use of a fixed sample rate against a musically-relevant sample rate. We test this system both on synthetic and real MIDI data. Results are compared in terms of note prediction accuracy. We show that the higher the sample rate is, the better the prediction is, because self transitions are more frequent. We suggest that for AMT, a musically-relevant sample rate is crucial in order to model note transitions, beyond a simple smoothing effect.

1. INTRODUCTION

Recurrent neural networks (RNNs) have become increasingly popular for sequence modelling in various domains such as text, speech or video [7]. In particular, long short-term memory networks (LSTMs) [10] have helped make tremendous progress in natural language modelling [15]. Those so-called *language models* can, in turn, be combined with *acoustic models* to improve speech recognition systems. Many recent improvements in this field have stemmed from better language models [8].

Automatic music transcription (AMT) is to music what speech recognition is to natural language: it is defined as the problem of extracting a symbolic representation from music signals, usually in the form of a time-pitch representation called *piano-roll*, or in a MIDI-like representation. Despite being one of the most widely discussed topics in music information retrieval (MIR), it remains an open problem, in particular in the case of polyphonic music [2]. While there has been extensive research on acoustic models for music transcription, music language models (MLMs) have received little attention until quite recently.

In this paper, we propose a study on the use of LSTM neural networks for symbolic polyphonic music modelling, in the form of piano-roll representations. We evaluate the impact of various parameters on the predictive performance of our system. Instead of relying on ever more complex architectures, we choose to use an LSTM with only one layer, and see how each parameter influences the final result, namely, the number of hidden nodes, the learning rate, the sampling rate of the piano-roll, and doing data augmentation. We also compare the use of time frames of fixed length against the use of beat-quantised time frames of fixed musical length (such as a 16th note). We evaluate the predictive performance of our system in terms of precision, recall and F-measure, and we monitor the evolution of these metrics throughout the learning process. We also conduct proof-of-concept experiments on AMT by post-processing the output of an existing acoustic model with our predictive models. We show that time-based time steps yield better results in terms of prediction because self-transitions are more frequent. This results in a simple smoothing effect when used in the context of transcription. On the other hand, note-based time steps perform worse for prediction, but show interesting behaviour that might be crucial for transcription, in particular the ability to model note transitions and some basic rhythmic features. To the best of our knowledge, such a study has not yet been done in the context of polyphonic music prediction.

The remainder of the paper is organised as follows. In section 2, we review existing works on symbolic music sequence modelling. In section 3, we describe the neural network architecture chosen for the experiments. In section 4, we describe the two employed datasets, one made of synthetic MIDI data, the other of real-life MIDI data. In section 6, we describe the various experiments performed for prediction and their results. In section 7, we show preliminary results on the application of the language model in the context of AMT. Finally, in section 8, we discuss the results obtained and their implications for future work.

2. STATE OF THE ART

Although MLMs have been discussed for quite a long time [14], they have not been specifically used in audio analysis until quite recently. Temperley [18] was one of the first to propose a joint probabilistic model for harmony, rhythm and stream separation, and suggested its use for AMT. Since then, several other audio analysis systems, such as [16], have used probabilistic models of high-level musical knowledge in order to improve their performance.

More recently, some approaches have used neural networks to post-process the output of an acoustic model. Indeed, it seems that neural networks are more suitable to model polyphonic sequences compared to probabilistic models such as hidden Markov models (HMMs). In [4], a neural network architecture combining a RNN with a Restricted Boltzman Machine (RBM) was proposed to estimate at each time-step a pitch distribution, given the previous pitches. This architecture was later integrated in various systems. In [17], it was integrated in an end-to-end neural-network for multi-pitch detection in piano music, coupled with a variety of neural-network-based acoustic models. In all these models, the time-step is of the order of 10 ms, which is small compared to the typical duration of a music note. Moreover, this time-step is constant, and does not depend on the tempo of the input music signal.

Some systems have also modelled symbolic music sequences for other purposes. Pachet et al. [9] proposed an architecture consisting of four joint neural networks in order to generate Bach chorales. In [11], another architecture using reinforcement learning to enforce musical rules in a RNN was proposed for music generation.

All the above neural architectures rely on sophisticated combinations of neural networks, and have many parameters, which means that they need a lot of training data, and can be prone to over-fitting. In this study, we focus on a simple architecture, and try to build from that using an experimental method to assess the importance of various hyper-parameters. A study similar to the present has been conducted in [13] on chord sequence modelling (thus on modelling monophonic sequences instead of polyphonic ones). In this previous study, the advantage of RNNs over HMMs is questioned in the context of chord sequence modelling. In particular, it is argued that when the frame rate is high (order of 10 fps), the RNN only has a smoothing effect, and thus is no more efficient than simpler models such as an HMM. On the other hand, it is suggested that on the chord-level (ie. one symbol per chord, no matter how long), the RNN used is significantly better than an HMM. We aim at investigating similar questions in the context of polyphonic note sequence modelling in the current study.

3. MODEL

In this section, we describe the model we used in the experiments. This model is trained to predict the pitches present in the next time frame of a piano-roll, given the previous ones.

3.1 Data Representation

The input is a piano-roll representation, in the form of an $88 \times T$ matrix M, where T is the number of timesteps, and 88 corresponds to the number of keys on a piano, between MIDI notes A0 and C8. M is binary, such that $M[p,t] = 1$ if and only if the pitch p is active at the timestep t. In particular, held notes and repeated notes are not differentiated. The output is of the same form, except it only has $T-1$ timesteps (the first timestep cannot be predicted since there is no previous information). We use two different timesteps:

- a fixed time step of 10 milliseconds, that we refer to as *time-based*
- a variable time step with a fixed musical length of a sixteenth note, referred to as *note-based*.

3.2 Network Architecture

Our primary goal is to study the influence of various parameters, namely the number of hidden nodes, the learning rate, the use of data augmentation, and the time steps used. In order to assess the influence of those parameters as accurately as possible, without being influenced by other parameters, we deliberately choose to use the most simple LSTM architecture possible. In particular, we choose not to use multiple layers, nor to use dropout or any other regularisation method during training. These will be investigated in future work.

We thus use an LSTM with 88 inputs, one single hidden layer, and 88 outputs. The number of hidden nodes is chosen among: $64, 128, 256, 512$. The network is trained using the Adam optimiser [12], using the cross-entropy between the output of the network and the ground truth as cost function. The learning rate is kept constant, and is chosen among: $0.01, 0.001, 0.0001$.

The output of the network is then sent through a sigmoid, and thresholded to obtain a binary piano-roll. The threshold is determined by choosing the one that gives the best results on the validation dataset (see section 4).

4. DATASETS

We use two different datasets for training and testing our models. One is a synthetic dataset, the other is a dataset made of real music pieces. Both datasets were split into training, validation and test datasets with the following respective proportions: 70%-10%-20%.

4.1 Synthetic MIDI Dataset

The synthetic MIDI dataset used in this study consists of combinations of notes and chords in the C major key. It contains only notes on the C major scale, between C3 and C5. The chords are either a note and the note a third above (major or minor, such that the second note is also in C major), or a note, the note a third above, and the note a fifth above. All generated files are 3sec long, with a tempo of 60, so each file is 3-quarter-notes long. All notes have a duration multiple of a quarter note, so note changes can occur after 1 second, 2 seconds, both or neither. We take all possible combinations of 3 notes or chords and we allow repetition. When a note or a chord is repeated we create two distinct files, one corresponding to the note being held, one corresponding to the note being played again. Overall, the dataset contains more than 36,000 files, representing 30 hours of data and will be referred to as *Synth dataset*.

4.2 Piano-midi.de Dataset

We use the Piano-midi.de dataset [1] as real-world MIDI data. This dataset currently holds 307 pieces of classical piano music from various composers. It was made by manually editing the velocities and the tempo curve of quantised MIDI files in order to give them a natural interpretation and feeling. This mode of production is the main reason why we chose it: it sounds real, with an expressive timing, and at the same time, the rhythmic ground truth is readily available. We can thus easily compute note-based time steps, without having to rely on a beat and meter detection algorithm.

This dataset holds pieces of very different durations (from 20 seconds to 20 minutes). In order to avoid excessive zero-padding for neural network training and to be more computationally efficient, we only keep the first minute from each file (and we zero-pad the shorter files). The resulting dataset is 5 hours long, and will be referred to as the *Piano dataset*. We also augment our dataset by transposing every piece in all keys from 7 semitones below to 6 semitones above. This increases the size of the dataset 12-fold, up to 60 hours. That way, all tonalities are equally represented, without shifting the range of notes too much.

5. EVALUATION METRICS

To evaluate the performance of our system, we compute several metrics following the MIREX Multiple-F0 Estimation task [1], namely the precision, recall and F-measure. Those metrics are computed for each file, and then averaged over a dataset. The progress throughout learning is computed on the validation dataset, and the performance of the trained model is computed on the test dataset.

6. PREDICTION

In this section, we compare the results obtained in various configurations, both quantitatively and qualitatively.

6.1 Number of Hidden Nodes and Learning Rate

We trained on the Synth dataset a series of models, with various numbers of hidden nodes in the hidden layer (*n_hidden*), and various learning rates (*learning_rate*). We tried all possible combinations with n_hidden $\in \{64, 128, 256, 512\}$ and learning_rate $\in \{0.0001, 0.001, 0.01\}$. We trained each model for 50 epochs, with a batch size of 50. All the implementations were made in Python, using Tensorflow [6]. The code necessary for the experiments can be found at: `http://www.eecs.qmul.ac.uk/~ay304/code/ismir17`.

In each case, the model converges to the same state: at epoch 50, all the models get the same precision, recall and F-measure with 10^{-2} precision. The only difference among all the models is the convergence speed. Similar observations were made for the other numbers of hidden nodes.

Quite expectedly, the parameter that has the most influence on convergence speed is the learning rate. Generally speaking, the larger the number of nodes is, the quicker the model converges (we could not compare when the learning rate is 0.01 since all the models converge in one epoch). Those empirical observations are consistently observed in all the other experiments as well (on the Piano dataset, with or without note-based time steps, with or without data augmentation).

When inspecting the output of the network before going through the sigmoid, we can notice some interesting features. All the notes that are outside the scale of C (that never appear in the training set) have a very low output, showing that the network is able to learn which notes might appear. This can be double-edged: notes outside the key are not mistakenly detected, but if they appear (which happens), they will not be detected either. In Section 7 we attempt to run this model on a real-life input file, and those findings are confirmed: the prediction clearly masks out every note that was not in the set of notes seen during training.

Considering the results of this preliminary experiment, we decide to keep for the rest of the experiments only n_hidden $\in [128, 256]$ and learning_rate $\in [0.001, 0.01]$. Indeed, 512 hidden nodes is too heavy computationally (around 20% longer training time compared to all the other configurations) without any clear improvement over 256 nodes. Similarly, a learning rate of 0.0001 converges too slowly compared to the others, with no noticeable advantage in the end result. We nevertheless choose to keep several models, not only the best one, because the best model on this dataset will not necessarily be the best one on another.

6.2 Data Augmentation

On the Piano dataset, we compare the performance of the model trained on the original 5-hour dataset, and on the augmented 60-hour dataset. The evolution of the F-measure on the validation dataset with and without data augmentation can be found in Figure 1. Results show that data augmentation improves greatly the results. All models trained on augmented data converge more quickly in terms of number of epochs, which is understandable since 12 times more data is processed at each epoch. However, in both cases, the results obtained after 50 epochs are approximately the same in terms of metrics.

6.3 Time Step

We compare the behaviour of the network when using time-based and note-based time steps, on both datasets. A comparison of the evolution of the F-measure on the Piano validation dataset with time-based or note-based time steps can be found in Figure 1.

With time-based time steps, we find that all the models seem to achieve similar results: with data augmentation, all

[1] `http://piano-midi.de/`

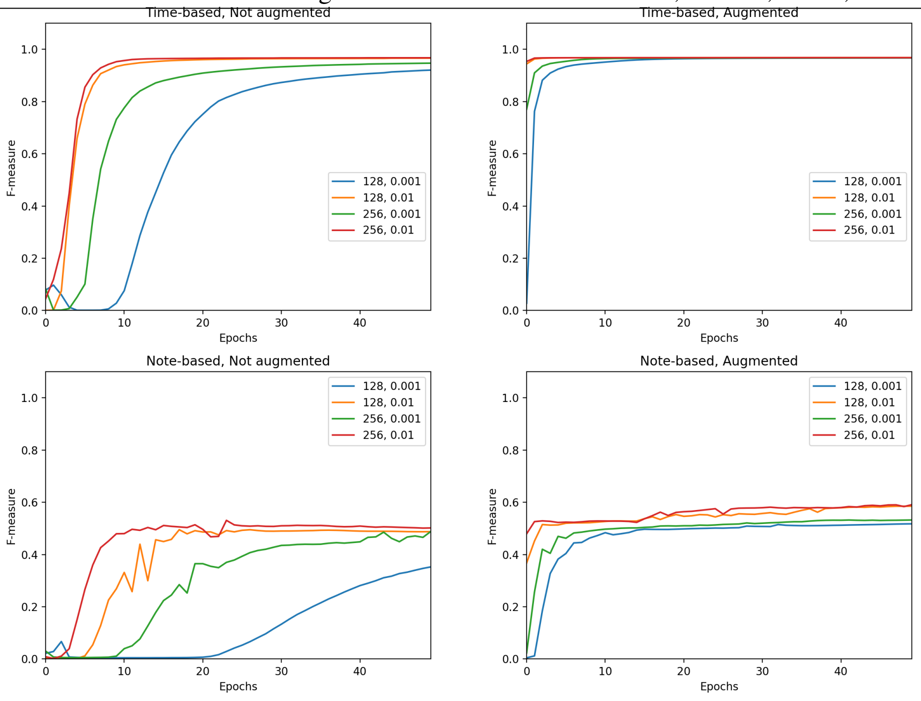


Figure 1. Comparison of the evolution of the F-measure across epochs, on the Piano validation dataset, with time-based or note-based timesteps, with or without data augmentation. A threshold of 0.5 is used.

	F-Measure	Precision	Recall
Without augmentation			
128, 0.001	0.451	0.409	0.509
256, 0.001	0.513	0.484	0.549
128, 0.01	0.548	0.536	0.560
256, 0.01	**0.553**	**0.544**	**0.562**
With augmentation			
128, 0.001	0.558	0.557	0.560
256, 0.001	0.571	0.552	0.592
128, 0.01	0.597	0.61	0.588
256, 0.01	**0.601**	**0.615**	**0.592**

Table 1. Prediction performance computed with note-based time steps on the Piano test dataset.

the models achieve a F-measure score of 0.966. Without data augmentation, the models trained with a learning rate of 0.01 achieve the same performance, with a learning rate of 0.001, the 128-hidden-node model achieves 0.917, and the 256-hidden-node model achieves 0.944. This might be due to the fact that they haven't fully converged after 50 epochs. All those scores were computed with a threshold of 0.5.

We also compute the precision, recall and F-measure on the Piano test dataset with note-based time steps. These results can be found in Table 1. We observe that this time, higher learning rate, higher number of nodes and data augmentation not only lead to quicker convergence, but also to better results.

For both datasets, the predictive accuracy is better in time-based configurations, since the frame rate is much higher, and thus there are more self-transitions (ie. notes are prolongated from the previous time steps). It seems indeed that in both cases, the system is uncertain when there are note changes, but learns to repeat the ongoing notes. The difference in performance can thus be attributed to the fact that note changes are much more frequent in the note-based case (once every 4 time steps versus once every 100 timesteps for the Synth dataset).

However, the note-based model shows very interesting behaviour at the uncertain time steps (ie. at each beat). On the Synth dataset, when the note changes, it gives a small probability to every note of the scale (the notes that might appear in the next frame), and rules out all the out-of-scale notes. Moreover, even when the note is kept for more than one beat, the model still shows the same "uncertain" behaviour, which does not happen with the time-based model. This is an error (which partly explains the worse scores), because the note should be held, but it has some very interesting consequences in terms of music modelling. This shows that the note-based model has learned that periodically, notes have a strong probability to change. This can be related to the rhythmic structure of music, as note changes are more frequent on strong metric positions. An example prediction output before thresholding is shown in Figure 2. We can see those "uncertain" time-steps in position 3 and 7, which correspond to time-steps 4 and 8 in the input (ie note changes).

We also find this behaviour with the Piano dataset, however only appearing with data augmentation. It is not clear if this is specific to data augmentation, or if it is simply because models without data augmentation were undertrained. In this case, the "uncertain" behaviour occurs at every eighth note, and with stronger probabilities at each quarter note. This suggests that the model behaves this way at the smallest metrical level, and not only at strong metrical positions. This might be a problem in the future, since it might encourage transitions too frequently. However, a small probability is only given to notes of the cor-

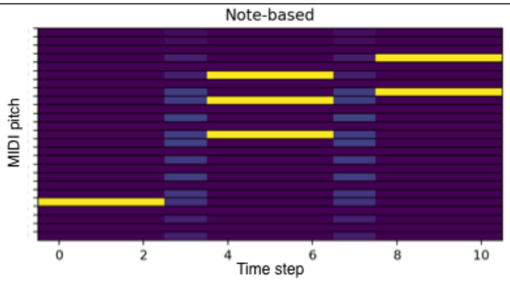


Figure 2. The prediction system output (n_hidden = 256, learning_rate=0.01) with note-based time steps after going through the sigmoid, before thresholding. The ground truth is: E3, C4E4G4, F4A4. At each note change, a small probability is given to all notes in C major scale.

rect scale, which shows that the model is able to get the tonal context of a piece to some extent. An example output before thresholding is shown in Figure 3.

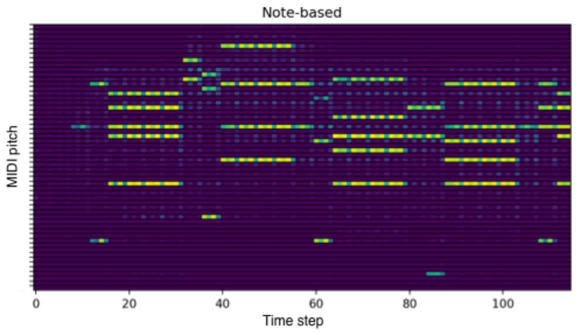


Figure 3. The output of the prediction system (n_hidden = 256, learning_rate=0.01) with note-based time steps after going through the sigmoid, before thresholding. The ground truth is Chopin's Prelude, No. 7, Op. 28 in A Major. At each eighth note, a small probability is given to some notes in A major, the tonality of the piece.

7. AUDIO TRANSCRIPTION

A preliminary experiment on assessing the potential of prediction models in the context of AMT is carried out. For this experiment, we use a single piece taken from the Piano dataset: Chopin's Prelude No. 7, Op. 28 in A Major.

7.1 Acoustic Model

For the experiment on integrating AMT with polyphonic music prediction, we use the system of [3], which is based on Probabilistic Latent Component Analysis. The system decomposes an input spectrogram into several probability distributions for pitch activations, instrument contributions and tuning deviations, using a dictionary of pre-extracted spectral templates. For this experiment, a piano-specific system is used, trained using isolated notes from the MAPS database [5]. The output of the acoustic model is a posteriogram of pitch activations over time.

7.2 Method

We synthesise the MIDI file with GarageBand, using the default Steinway Grand Piano soundbank. We analyse 3 different audio files:

- The full audio file, with expressive timing.
- The right-hand of the piece, transposed in C. In this case, predictive models trained both on the Synth and Piano dataset are evaluated.
- The full audio file, with quantised timing. The output of the transcription system is then downsampled to obtain a time step of a 16th note.

We take the posteriogram output by the previously described acoustic model, and feed it to various polyphonic prediction models, in various conditions:

- The raw posteriogram, with positive values theoretically unbounded (*raw_post*)
- The raw posteriogram thresholded in order to have a binary piano-roll (*raw_piano*)

The output of our predictive model is then thresholded using the value determined on the validation dataset in the experiments described in Section 6.3. The resulting piano-roll is compared to the ground truth using the accuracy metrics described in Section 5. An example of output of the baseline transcription system is shown in Figure 4.

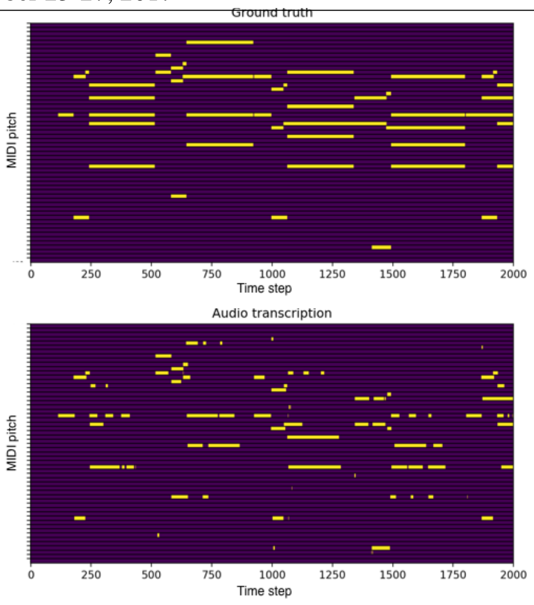


Figure 4. The first 20 seconds of the thresholded output of the baseline AMT system, compared with the ground truth.

7.3 Results

Results in terms of multi-pitch detection are shown in Table 2. Although we tested every configuration with all our models, we only report the results of the most meaningful experiments.

In the vast majority of cases, the results with the predictive model are below those of the acoustic model only, without post-processing. The only cases where the post-processing step improves the results is when the prediction is made on the whole piece, with time-based time steps, in *raw_piano* configuration. Otherwise, the results are at best equivalent to those of the baseline system. In the case where the results are improved, we inspect what improvements are made by the predictive model. It seems that the only improvements were few isolated frames that are temporally smoothed. We do not notice any missing notes be-

	F-Measure	Precision	Recall
Full audio, raw_piano			
Baseline	0.455	0.960	0.299
128, 0.001	0.458	0.938	0.303
256, 0.001	0.458	0.941	0.303
128, 0.01	0.460	0.959	0.303
256, 0.01	**0.460**	**0.961**	**0.303**
Right hand in C, raw_post, Synth			
Baseline	**0.670**	0.898	**0.535**
128, 0.001	0.556	0.955	0.393
256, 0.001	0.607	**0.966**	0.442
128, 0.01	0.522	0.834	0.380
256, 0.01	0.527	0.877	0.377
Full note-based, raw_piano			
Baseline	**0.526**	**0.963**	**0.361**
128, 0.001	0.434	0.624	0.332
256, 0.001	0.440	0.651	0.332
128, 0.01	0.478	0.852	0.332
256, 0.01	0.481	0.875	0.332

Table 2. Some results on transcription from audio, compared to the output of the baseline acoustic model.

ing added, and very few spurious notes are removed.

When using the Synth-dataset-trained models on the right hand transposed in C, the results are mixed. The precision measure is improved, due to the fact that many spurious notes are removed. On the other hand, some notes went completely missing because they were not in the C major scale, which leads to a lower recall score. Overall, the F-measure is lower than that of the acoustic model .

When using frame-based time steps, in every configuration, the results are worse. We have identified two reasons for that. The first is that sometimes, the system would add evenly distributed noise at the beginning of the prediction. This is probably due to the fact that the network takes a few frames to build a memory good enough to make correct predictions. More training removes that problem (the problem does not appear with models trained with a learning rate of 0.01). The second reason is that the system has some latency: a note is only activated one frame after it is seen in the input, and it is only deactivated one frame after it disappears of the input. When comparing the output of the system shifted one frame back with the output of the baseline system, the results were very close, and in some cases, identical (though never better).

8. DISCUSSION

In this study, we compare the influence of various parameters of an LSTM network on modelling polyphonic music sequences with respect to the training process and prediction performance. Those parameters are: the learning rate, the number of hidden nodes, the use of data augmentation by transposition, and the use of time-based time steps against note-based time steps. We found that with a given time step and a given dataset, the learning rate is the most important factor for learning speed, and that the more hidden nodes there are, the quicker the convergence is. We also found that data augmentation greatly improves both the convergence speed and the final performance.

When it comes to the choice of the time steps, it appears that time-based time steps yield a better prediction performance, because self transitions are more frequent. On the other hand, note-based time steps seem to show better musical properties. When trained on synthetic data containing only notes of the scale, it seems rather evident that notes that are have never been obeserved are very unlikely. More interestingly, when trained with real data in all tonalities, the system can still detect the scale of the piece : we can see with the example in Figure 3 that only notes of the correct tonality are given a some probability. We can also see that the system has learned the places where note changes might occur, and that the note changes are more frequent at beat positions than at half-beat positions.

We also use our system to post-process the output of an acoustic model for multi-pitch detection, in a proof-of-concept experiment. The first thing that we can notice from this experiment is that a good prediction performance does not necessarily translate to a good audio transcription performance. However, the order of performance for prediction seem to be kept for transcription: models with more nodes and higher learning rate tend to perform better.

The poor performance of our the predictive model for improving AMT performance is understandable: the input presented to the system in the transcription process is very different from those the models were trained on. Future work will include training predictive models not with piano-rolls, but with acoustic model posteriograms.

From all the above experiments, we can conclude that with time-based time steps, what the LSTM does is a simple smoothing, albeit a good one, since it improves transcription performance in some cases. The very fact that post-processing the output of the acoustic model with our system can improve the transcription performance, even though our language model was trained on a completely different kind of data, shows that it has in fact not learned much from the data it was fed, except temporal smoothing similar to e.g. a median filter. Since we aim at modelling the complex vertical and horizontal dependencies that exist within music, this behaviour is not sufficient.

On the other hand, we found some very interesting features in the output of the note-based models: they are able to learn when note changes might occur and what note might be activated which is very promising in terms of polyphonic music modelling. The downside of such models is that they would rely on meter detection algorithms when applied to audio, which might lead to error propagation. Future work will focus on investigating the possibilities of those models in the context of AMT, assuming that the meter and tempo are given as a first step.

Finally, we will extend this study in future work by gradually increasing the complexity of our model, and studying how the performance varies. In particular, we will study the result of adding more hidden nodes, and using more sophisticated regularisation techniques. We will also further investigate the results by visualising the learned parameters, as well as the activations of the hidden nodes.

9. ACKNOWLEDGEMENTS

A. Ycart is supported by a QMUL EECS Research Studentship. E. Benetos is supported by a RAEng Research Fellowship (RF/128).

10. REFERENCES

[1] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[2] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Society for Music Information Retrieval Conference (ISMIR)*, pages 701–707, 2015.

[4] N. Boulanger-Lewandowski, P. Vincent, and Y. Bengio. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *29th International Conference on Machine Learning*, pages 1159–1166, 2012.

[5] 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*, 18(6):1643–1654, August 2010.

[6] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[8] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649. IEEE, 2013.

[9] G. Hadjeres and F. Pachet. DeepBach: a steerable model for Bach chorales generation. *arXiv preprint arXiv:1612.01010*, 2016.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] N. Jaques, S. Gu, R. E. Turner, and D. Eck. Tuning Recurrent Neural Networks with Reinforcement Learning. *5th International Conference on Learning Representations (ICLR)*, pages 1722–1728, 2017.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

[13] F. Korzeniowski and G. Widmer. On the Futility of Learning Complex Frame-Level Language Models for Chord Recognition. In *AES International Conference on Semantic Audio*, 2017.

[14] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[15] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[16] S. A. Raczyński, E. Vincent, and S. Sagayama. Dynamic Bayesian networks for symbolic polyhonic pitch modeling. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1830 – 1840, 2013.

[17] S. Sigtia, E. Benetos, and S. Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, May 2016.

[18] D. Temperley. A Unified Probabilistic Model for Polyphonic Music Analysis. *Journal of New Music Research*, 38(1):3–18, 2009.

AUDIO TO SCORE MATCHING BY COMBINING PHONETIC AND DURATION INFORMATION

Rong Gong, Jordi Pons and Xavier Serra
Music Technology Group, Universitat Pompeu Fabra, Barcelona
rong.gong@upf.edu, jordi.pons@upf.edu, xavier.serra@upf.edu

ABSTRACT

We approach the singing phrase audio to score matching problem by using phonetic and duration information – with a focus on studying the jingju a cappella singing case. We argue that, due to the existence of a basic melodic contour for each mode in jingju music, only using melodic information (such as pitch contour) will result in an ambiguous matching. This leads us to propose a matching approach based on the use of phonetic and duration information. Phonetic information is extracted with an acoustic model shaped with our data, and duration information is considered with the Hidden Markov Models (HMMs) variants we investigate. We build a model for each lyric path in our scores and we achieve the matching by ranking the posterior probabilities of the decoded most likely state sequences. Three acoustic models are investigated: *(i)* convolutional neural networks (CNNs), *(ii)* deep neural networks (DNNs) and *(iii)* Gaussian mixture models (GMMs). Also, two duration models are compared: *(i)* hidden semi-Markov model (HSMM) and *(ii)* post-processor duration model. Results show that CNNs perform better in our (small) audio dataset and also that HSMM outperforms the post-processor duration model.

1. INTRODUCTION

The ultimate goal of our research project is to automatically evaluate the jingju a cappella singing of a student in the scenario of jingju singing education – see Figure 1. Jingju, a traditional Chinese performing art also known as Peking or Beijing opera, is extremely demanding in the clear pronunciation and accurate intonation of each syllabic or phonetic singing unit. To this end, during the initial learning stages, students are required to completely imitate tutor's singing. Therefore, the automatic jingju singing evaluation system we envision is based on this training principle and measures the intonation and pronunciation similarities between the student's and the tutor's singings. Before measuring similarities, the singing phrase should be automatically segmented into syllabic or phonetic units in order to capture the temporal details. Jingju music scores, which contain the phonetic and duration information for each singing syllable, will be beneficial for this segmentation. In the application scenario, the score of a query audio could be selected from the database by the user itself. However, to avoid manual intervention and improve the user experience, we tackle the problem of automatically finding the corresponding music score for a given query audio (bold in Figure 1). Note that achieving successful methods for audio to score matching might be beneficial for several music informatics research (MIR) tasks, such as score-informed automatic syllable/phoneme segmentation [5] or score-informed source separation [14]. The objective of this research task is to find the corresponding score for a given singing audio query. We restrict this research to the "matching" scope by pre-segmenting both the singing audios and the music scores into the phrase units.

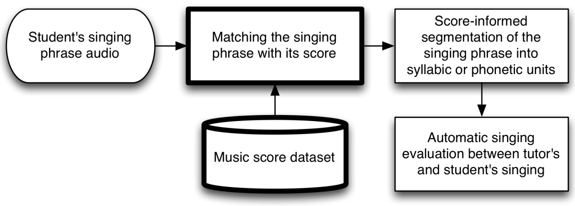

Student's singing phrase audio
Matching the singing phrase with its score
Score-informed segmentation of the singing phrase into syllabic or phonetic units
Music score dataset
Automatic singing evaluation between tutor's and student's singing

Figure 1. System design framework of the entire research project. The modules with bold border lines are addressed in this paper.

Xipi and *erhuang* are the main modes in jingju music. Each has two basic melodic contours – an opening phrase and a closing phrase. Each basic melodic contour is constructed upon characteristic pitch progressions for each mode [22]. Therefore, singing phrases from different arias sharing the same mode are likely to have a similar melodic contour. Figure 2 shows an example of this fact. However, melodic information tends to be intuitively used for such matching tasks. For example in Query-by-Singing/Humming (QBSH) [15], melodic similarities can be obtained by comparing the distance between the F0 contour of the query audio and those synthesized from the candidate scores. Then, the best matched music score can be retrieved by selecting the most similar melody. But note that using this approach for jingju music would bring matching errors since the melodic contours of the same mode are similar in this sense. In this case, it is more ap-

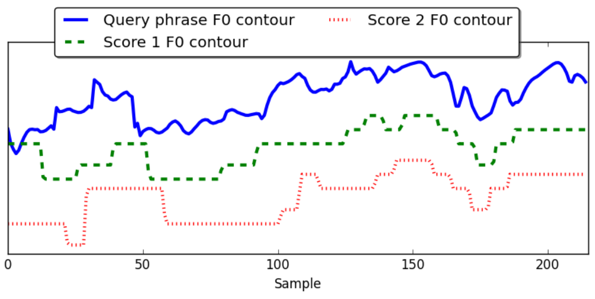


Figure 2. An example of different phrases having a similar *Xipi* melodic contour in our score dataset. The lengths of these contours are normalized to a fixed sample size.

propriate to use another notion of similarity. We propose using the lyrics since the stories narrated in different jingju arias are distinctive and lyrics tend to change through different jingju arias – even when they share the same mode. Therefore, phonetic information might be useful to identify a similar score given a query audio.

QBSH is the most related research task to our study, which retrieves a song by singing a portion of itself. Most of the studies use melody information as the only cue. The typical process of such systems was introduced by Molina *et al.* [15]: firstly, the F0 contour and/or a note-level transcription for a given singing query are extracted; and then, a set of candidate songs are retrieved from a large database using a melodic matcher module. The most successful QBSH system, which obtained the best results in MIREX 2016 contest, is based on the method of multiple similarity measurements fusion [21]. This system proposed a melodic matcher which combines several similarities that are note-based and frame-based. The authors claim that the fusion mechanism improves the query performance because no similarity measurement is perfect. Therefore, information sources that are complementary to each other might be beneficial for this approach. Very few studies have explored the capability of the phonetic information for QBSH. Guo *et al.* [8] and Wang *et al.* [20] both used a lyric recognizer based on Hidden Markov Models (HMMs). Their recognition networks [1] were constructed with the phonetic information from the query candidates database. They used frame-based MFCCs to create the acoustic models with GMMs. Then, the Viterbi algorithm was executed over the recognition networks to either obtain the most likely phonetic state sequence (for Wang *et al.* [20]) or the posterior probability of each possible decoding path (for Guo *et al.* [8]). The final score of a query candidate is either based on semantic similarity [20] or based on the posterior probability of its corresponding lyrics [8].

Another research task related to our study is singing keyword spotting. The main goal of this task is to search for one or more keywords in a singing query. The system proposed by Kruspe [12] searches for a specific singing keyword on the resulting phoneme observations. A keyword-filler HMMs is employed for this purpose. She used two phoneme duration models: the HSMM and the post-processor duration model.

Finally, both phonetic and duration information extracted from the score have been extensively used in alignment-related tasks, such as audio-to-score alignment and audio-to-lyrics alignment. For example, Gong *et al.* [4] construct a left-to-right HSMM using phonetic and duration information. Or Dzhambazov *et al.* [3] use a similar approach for aligning polyphonic audio. Analogously, the proposed approach explores the use of both phonetic and duration information (available in scores) to tackle the matching ambiguity problem existing in jingju music.

The remainder of this paper is organized as follows: the used dataset is introduced in section 2, section 3 explains the modules of the proposed approach – detailing how to incorporate phonetic and duration information. Experiments and results are reported in section 4, and section 5 concludes and points out future work.

2. DATASET

The jingju a cappella singing dataset is composed of two overlapping parts: *(i)* audio and *(ii)* score datasets.

The audio dataset [1] used for this study consists of two role-types singing: *dan* (young woman) and *laosheng* (old man). The *dan* part of this dataset has 42 recordings sung by 7 singers and the *laosheng* part has 23 recordings sung by 7 *laosheng* singers. The boundary annotations of the audio dataset have been done in Praat format (textgrid) considering a hierarchy of three levels: phrase, syllable and phoneme – using Chinese characters, pinyin notations and X-SAMPA notations, respectively. 32 phoneme classes are used in the phoneme-level annotation. Two Mandarin native speakers and a jingju musicologist have been devoted to this annotating work. Annotations and more detailed information can be found online [2]. Some statistics about the dataset are reported in Table 2. The average phrase, syllable and voiced phoneme length of *dan* singing are ostensibly greater than those of *laosheng* singing (bold numbers in Table 2), which might indicate that *dan* singing tends to have more pitch variation and ornamentation – as we could observe empirically by listening to the data.

	Num.	Avg. len (s)	Std. len (s)
Phrases	325	**16.42**	14.11
Syllables	2933	**1.58**	2.82
Voiced phonemes	7198	**0.61**	0.97
Unvoiced phonemes	2014	0.10	0.67
Phrases	247	9.47	8.14
Syllables	2289	0.88	1.48
Voiced phonemes	4948	0.39	0.78
Unvoiced phonemes	1454	0.07	0.05

Table 1. Detailed information of the jingju a cappella singing audio dataset: *dan* (top), *laosheng* (bottom).

[1] The topology of the HMM is defined by the recognition network.

[2] http://doi.org/10.5281/zenodo.344932

The audio dataset, along with their boundary annotations, is split into three parts: training set, development (dev) set and test set. We define the training set to be the non-overlapping part with the score dataset, see Figure 3. The training set will be used for calculating the phonetic duration (duration information) and training acoustic models (phonetic information). After taking the training set out, we define the development set to be the half of the remaining phrases in the audio dataset (randomly selected) – it will be used for parameters optimization. The test set consists on the remaining phrases of the audio dataset – it will be used for testing the acoustic models performance and the matching performance.

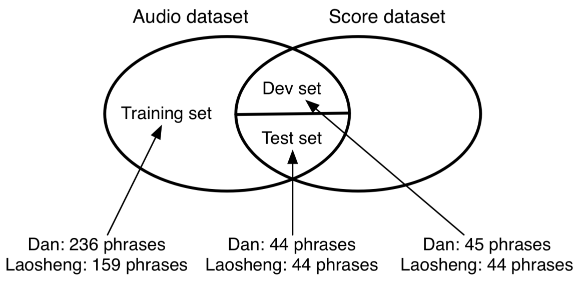


Figure 3. The intersection between the audio and the score datasets. The partition of the audio dataset.

On the other hand, the score dataset contains 435 *dan* phrases and 481 *laosheng* phrases. The scores have been typed in stave notation (including lyrics in Chinese characters) using MuseScore from different printed sources in jianpu notation. Since tempo is usually not clearly noted in the printed score, we do not include this information in the dataset. The relative syllabic durations are indicated by the note durations corresponding to the lyrics, which will be used to calculate the phonetic duration (duration information) and the matching network. The whole score dataset will be used as candidates for testing the matching performance and for parameter optimization.

3. APPROACH

The proposed approach aims to match the query audio to its score by using phonetic and duration information. During the training process (red boxes in Figure 4): the acoustic models of each phoneme are shaped by using the audio training set and its phonetic boundary annotations; the score dataset is used to construct a matching network; and phoneme duration distributions are estimated by using both audio training set and scores. During the matching process (green boxes in Figure 4): two duration models –HSMM and post-processor– are explored for the Viterbi decoding step. Finally, the best-matched phrase is found by ranking the decoded state sequence probabilities.

3.1 Acoustic models

Here presented acoustic models aim to represent the relationship between an audio signal and the 32 phoneme

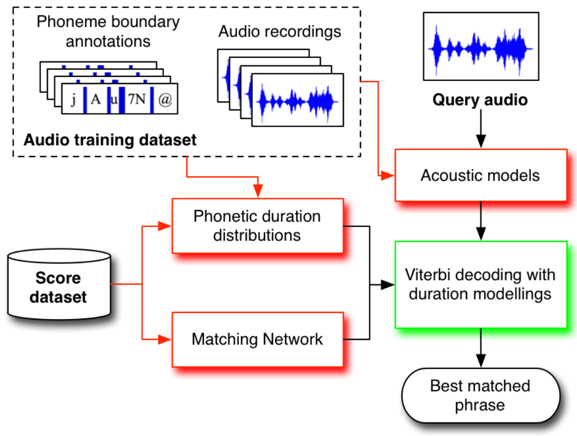


Figure 4. Diagram of the proposed approach.

classes present in our dataset. The output of these models yield probability scores for each phoneme class.

The most popular way to approach acoustic modeling is by using GMMs and MFCCs features [8, 20]. For that reason, we set as baseline a 40-component GMM with the following input vector: 13 MFCCs, their deltas and delta-deltas. Moreover, DNNs have been found very useful for acoustic modeling [9, 13]. Therefore, we propose an additional baseline: a DNN with 2 hidden layers followed by the 32-way softmax output layer – the input is set to be a log-mel spectrogram.

However, DNNs are very prone to over-fitting and the available dataset is relatively small. For that reason we propose using CNNs since these are more robust against over-fitting – note that CNNs allow parameter sharing. Additionally, Pons *et al.* [17] have successfully used spectrograms-based CNNs for learning music timbre representations from small datasets. Given that timbre is an important feature for acoustic modeling, we propose using the same architecture: a single convolutional layer with filters of various sizes [16, 17]. The input is set to be a log-mel spectrogram. We use 128 filters of sizes 50×1 and 70×1, 64 filters of sizes 50×5 and 70×5, and 32 filters of sizes 50×10 and 70×10 – where the first and second numbers denote the frequential and temporal size of the filter, respectively. A max-pool layer of $2\times N'$ is followed by a 32-way softmax output layer with 30% dropout – where N' denotes the *temporal* dimension of the resulting feature map. $2\times N'$ max-pool layer was chosen to achieve time-invariant representations while keeping the frequency resolution. And *same* padding is used to preserve the dimensions of the feature maps so that these are concatenable. Filter shapes are designed so that filters can capture the relevant time-frequency contexts for learning timbre representations – according to the design strategy proposed by Pons *et al.* [17]

Log-mel spectrograms are of size 80×21 – the network takes a decision for a frame given its context: ±10 frames, 21 frames in total. Activation functions are ELUs [2] for all deep learning models and these are optimized with stochastic gradient descent (batch size: 64), using ADAM [11] and early stopping – when validation loss (categorical cross-entropy) does not decrease after 10 epochs.

Spectrograms are computed from audio recordings sampled at 44.1 kHz. STFT is performed using a window length of 25ms (2048 samples with zero-padding) with a hop size of 10ms. The 80 log-mel bands energies are calculated on frequencies between 0Hz and 11000Hz and these are standardized to have zero mean and unit variance.

The acoustic models are trained separately for each role-type and their performance is reported in section 4.2.

3.2 Matching network

The matching network defines the topology of the hidden Markov model. By using each candidate phrase in the score dataset as an isolated unit, isolated-phrase matching networks can be constructed. Figure 5 shows the structure of this matching network, which has $K = 916$ lyric paths.

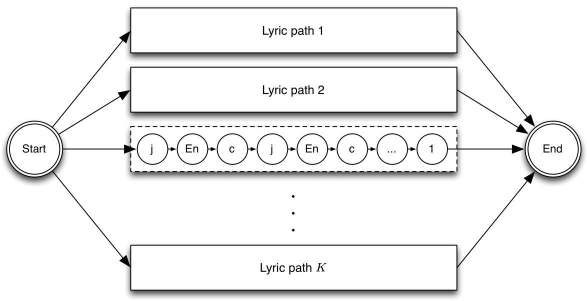


Figure 5. The structure of the K paths isolated-phrase matching network. Path 3 shows an example of the left-to-right state chain structure of an individual lyric path.

The matching network uses HMMs or HSMMs, depending on how the internal duration is modeled. Each path is a left-to-right state chain which represents the phoneme transcription of its lyrics. In order to construct the lyric path, pinyin lyrics are segmented into phonetic units and transcribed into X-SAMPA notations by using a predefined dictionary. For example, a path which has the lyrics *yan jian de hong ri* in pinyin is a chain consisting of 12 states: j, En, c, j, En, c, 7, x, UN, N, r\', 1 in X-SAMPA notation. When the decoding process has finished, each lyric path can get a posterior probability which will be used as the similarity measure between the query phrase and the candidate phrase.

3.3 Phonetic duration distributions

Phonetic duration information comes from two sources: the boundary annotations of audio training dataset and the score dataset. The phonetic duration is not directly indicated in the score. However, it is indispensable for modeling the phonetic duration distribution for each state in the matching network. The syllable, of which duration can be deduced by the corresponding note(s), is used to restrict the durations of the phonemes.

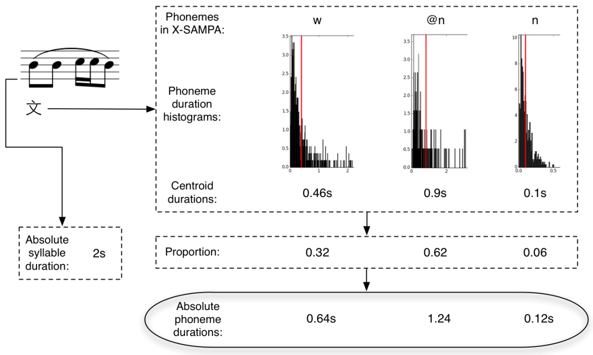


Figure 6. Flowchart example of estimating the phonetic durations of a syllable.

In the following, we propose a method for estimating the absolute phonetic duration given: *(i)* the score, and *(ii)* the phonemes duration histograms computed from the audio dataset annotations. First, we omit silence parts in the query audio (with a simple voice activity detection method [19]) and also in the score by removing the rest notes. Second, we compute the duration histogram and its duration centroid for each phoneme class – by aggregating the phonetic durations indicated in the boundary annotations of the audio training dataset. Then, we segment each syllabic duration in the score dataset into phonetic durations according to the proportion of their duration centroids. Finally, as the scores do not contain tempo, we normalize the phonetic durations of each phrase such that their summation is equal to the duration of the query audio. See Figure 6 for an equivalent graphic explanation. In Figure 6, the centroid durations of these three phonemes are: 0.46s, 0.9s and 0.1s, summing: 1.46s – alternatively, these can be expressed as a proportion of 1.46s: 0.32, 0.62 and 0.06. With these proportions and the absolute syllable duration (2s), we can compute the absolute phoneme durations: 0.32·2s = 0.64s, 0.62·2s=1.24s and 0.06·2s=0.12s.

The phonetic duration distribution needs to be calculated for each state in the matching network in order to incorporate the *a priori* phonetic duration information. We model it by Gaussian distributions:

$$\mathcal{N}(x; \mu_l, \sigma_l^2) = \frac{1}{\sqrt{2\pi}\sigma_l} \exp\left(-\frac{(x-\mu_l)^2}{2\sigma_l^2}\right). \quad (1)$$

where μ_l is the duration of the phoneme l deduced by the above method and the standard deviation σ_l is proportional to μ_l: $\sigma_l = \gamma\mu_l$. The proportionality constant γ will be optimized in section 4.3 for each role-type.

3.4 Duration modeling

Standard Markovian state does not impose explicitly duration distribution, instead, imposing an implicit state occupancy distribution which corresponds to a "1-shifted" geometric distribution [6]:

$$d_j(u) = (1 - \tilde{p}_{jj})\tilde{p}_{jj}^{u-1} \tag{2}$$

where u denotes the occupancy or sojourn time in a Markovian state j and $\tilde{p}_{jj}$ denotes the self-transition probability of the state j. Because of the implicity of the Markovian state occupancy, the phonetic duration distribution introduced in section 3.3 can not be imposed. Kruspe [12] presents two duration modeling techniques for HMMs: Hidden semi-markov model (HSMM) and post-processor duration model.

3.4.1 Hidden semi-markov model

Guédon [6] defined a semi-Markov chain S_t with finite state space $0, ..., J-1$ by the following parameters:

- initial probabilities $\pi_j = P(S_0 = j)$ with $\sum_j \pi_j = 1$
- transition probability of semi-Markovian state j: for each $k \neq j, p_{jk} = P(S_{t+1} = k | S_{t+1} \neq j, S_t = j)$ with $\sum_{k \neq j} p_{jk} = 1$ and $p_{jj} = 0$

An explicit occupancy distribution is attached to each semi-Markovian state:

$$\begin{aligned} d_j(u) = P(S_{t+u+1} \neq j, S_{t+u-v} = j, v = 0, ..., u-2 \\ |S_{t+1} = j, S_t \neq j), u = 1, ..., M_j \end{aligned} \tag{3}$$

where M_j denotes the upper bound to the time spent in state j. $d_j(u)$ defines the conditional probability of leaving state j at time $t+u+1$ and entering state j at time $t+1$.

To apply HSMMs to the matching network, we first use the matching network as the HSMMs topology. Thus the state occupancy distribution is set to its corresponding phonetic duration distribution. Then the probabilities of each left-to-right state transition are set to 1 because all self-transition probabilities in HSMMs are 0. The goal is to find the most likely sequence of hidden states for each lyric path and collect its posterior probability. The Viterbi algorithm meets this specific goal and its complete implementation is provided in [7].

3.4.2 Post-processor duration model

The post-processor duration model was first introduced by Juang *et al.* [10]. It was then experimentally proved in Kruspe's paper [12] that this duration model works better than HSMMs for the keyword spotting task in English pop singing voice. The post-processor duration model uses the original HMMs Viterbi algorithm – therefore, during the decoding process, no explicit occupancy duration distribution is imposed.

The log posterior probability of the decoded most likely state sequence is augmented by the log duration probabilities:

$$\log \hat{f} = \log f + \alpha \sum_{j=1}^{N} \mathcal{N}(u_j; \mu_j, \sigma_j^2) \tag{4}$$

where f is the HMMs posterior probability, α is a weighting factor which will be optimized in section 4.3, $j = 1, ..., N$ is the decoded state number in the most likely state sequence, and $\mathcal{N}(u_j; \mu_j, \sigma_j^2)$ is the occupancy probability of being in state j for the occupancy u_j.

4. EXPERIMENTS AND RESULTS

4.1 Performance metrics

Two experiments[3] are performed: the first is to evaluate the performance of the acoustic models, and the second is to evaluate the proposed matching approaches. For the first task, we use one simple evaluation metric: the overall classification accuracy which is defined as the fraction of instances that are correctly classified. For the second task, our goal is to evaluate the ability to match the ground-truth phrase in the score dataset to the query one, which is almost identical to the goal of a QBSH system: *"finding the ground-truth song in a song database from a given singing/humming query"*. Therefore, we borrow the standard performance metrics used in QBSH task to evaluate our approaches: Top-M hit and Mean Reciprocal Rank (MRR) [8]. The Top-M hit rate is the proportion of queries for which $r_i \leq M$, where r_i denotes the rank of the ground-truth score phrase. MRR is the average of the reciprocal ranks across all queries, n is the number of queries, and $rank_i$ is the posterior probability rank of the ground-truth phrase corresponding to the i-th query.

$$MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{rank_i} \tag{5}$$

4.2 Acoustic models

CNN, DNN and GMM acoustic models yield probability scores for each phoneme class. In order to evaluate the classification accuracy, we choose the phoneme class with the maximum probability score as the prediction. Table 4.2 reports the performance of CNN, DNN and GMM acoustic models evaluated on the test set.

	dan(#parameters)	*laosheng*(#parameters)
CNNs	0.484(222k)	0.432(222k)
DNNs	0.284(481k)	0.282(430k)
GMMs	0.290(-)	0.322(-)

Table 2. Overall classification accuracies of CNN and baseline acoustical models for *dan* and *laosheng* datasets.

The relatively low classification accuracies for all three models show that modeling the phonetic characteristics of jingju singing voice is a challenging problem. Our best results are achieved with CNNs – and GMMs perform better than DNNs. Interestingly, these results contrast with the literature where Hinton *et al.* [9] describe that DNNs acoustic models largely outperform GMMs for automatic speech recognition, and Maas *et al.* [13] showed that CNNs

[3] Code:https://goo.gl/1XB6j1

perform worse than DNNs for building speech acoustic models. First, we argue that in our case DNNs perform worse than GMMs and CNNs because a small amount of training data is available. DNNs require a lot of training data to achieve good performance and note that large amounts of training data are typically not available for most MIR tasks. And second, note the CNNs used here are specifically designed to efficiently learn timbre representations [17] while Maas *et al.* [13] used small square filters, which proved successful in computer vision tasks. These results show that using CNN architectures designed for the task at hand is especially beneficial in small data scenarios. A CNN model is used in the following experiments.

4.3 Parameters optimization

The parameters which need to be optimized for *dan* and *laosheng* role-types are: the weighting factor α for the post-processor duration model, and the proportionality constant γ for both models: HSMMs and post-processor duration model. Table 4.3 reports the optimal values we obtained by doing grid search on the development set – MRR metric was maximized.

Parameters	Search bounds	Optimal values
α	$[0.25, 2]$ with step 0.25	1.0 / 1.0
γ HSMMs	$[0.1, 2]$ with step 0.1	0.1 / 0.1
γ post-processor	$[0.1, 2]$ with step 0.1	0.7 / 1.5

Table 3. Parameters to be optimized, search bounds and resulting optimal values (*dan* / *laosheng*).

4.4 Duration modeling

To highlight the advantage of using duration modeling methods for audio to score matching, a standard HMM without explicitly imposing the occupancy distribution is used as a baseline. Results in Figure 7 show that its performance is inferior to the HSMM duration model.

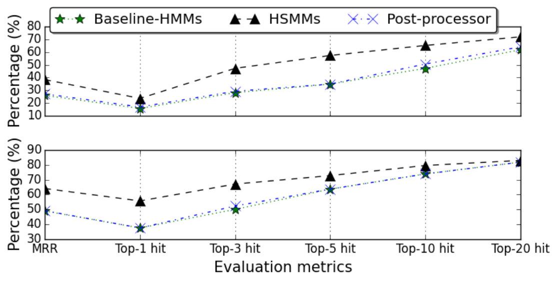


Figure 7. Phrase matching performance of HSMM and post-processor duration model with CNN acoustic model: *dan* (top), *laosheng* (bottom).

One can also observe in Figure 7 that HSMM performs the best, improving the baseline MRR metric performance by 13.2% for *dan* role-type and 15.1% for *laosheng* role-type. This means that HSMMs explicit duration modeling can help achieve a better audio to score matching by using phonetic information.

The post-processor duration model does not significantly improve the baseline performance. This result contrasts with the literature, where the post-processor duration model worked better than HSMMs for singing voice keyword spotting [12]. This inconsistency might result from *(i)* the length difference of the matching unit (singing-words vs. singing-phrases), and *(ii)* the large standard deviation of the jingju singing phonemes length. First, in Kruspe's work [12], the matching unit is the singing keyword – which usually contains fewer phonemes than a singing phrase (as in our case). And second, the vowel length standard deviation of the a cappella dataset used by Kruspe [12] (around 0.3s) is much short than in our dataset (*dan*: 0.97s, *laosheng*: 0.78s) – denoting less vowel duration variance than in our study case. Moreover, a significant deficiency of the post-processor duration model is that it does not provide the most likely state sequence by internally considering the durations, but it computes a *new* weighted likelihood given the obtained sequence [12]. If the most likely state sequence is decoded poorly, it can't be restored by the post-processor duration model.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an audio to score matching approach that uses phonetic and duration information.

We explored two duration models: HSMM and post-processor duration model. HSMMs achieved better results than the post-processor duration model – probably due to *(i)* the matching units length, *(ii)* the large standard deviation of the considered phonemes, and *(iii)* because for the post-processor duration model it is hard to recover a decoding mistake. Moreover, HSMMs achieved a better matching performance than the baseline-HMMs approach, which only took into account phonetic information, denoting the utility of using duration information.

We also compared CNN, DNN and GMM acoustic models, and CNNs have shown to be superior in our small singing voice audio dataset. The used CNN architecture was specifically designed to learn timbral representations efficiently [17] – this being the key factor for enabling CNNs (a deep learning method requiring large amounts of data) to perform so well on such a small dataset.

There are many possibilities to improve our approach. It has been shown in the speech research field that LSTM RNNs achieved the best acoustic modeling performance [18]. However, this method requires a large training dataset in order prevent from over-fitting. Another possibility to improve our acoustic model is to go deeper with the current single-layer CNN architecture, but this will also require more training data. We plan to collect more jingju a cappella singing recordings and perform data augmentation to leverage the capability of the acoustic models. Furthermore, in order to take advantage of the melodic information existing in both audio and score datasets, we also plan to investigate methods which can fuse melodic, phonetic and duration information.

6. ACKNOWLEDGEMENTS

We are grateful for the GPUs donated by NVidia. This work is partially supported by the Maria de Maeztu Programme (MDM-2015-0502) and by the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583).

7. REFERENCES

[1] D. A. A. Black, M. Li, and M. Tian. Automatic Identification of Emotional Cues in Chinese Opera Singing. In *ICMPC*, Seoul, South Korea, August 2014.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv:1511.07289*, 2015.

[3] G. B. Dzhambazov and X. Serra. Modeling of phoneme durations for alignment between polyphonic audio and lyrics. In *SMC*, Maynooth, Ireland, 2015.

[4] R. Gong, P. Cuvillier, N. Obin, and A. Cont. Real-Time Audio-to-Score Alignment of Singing Voice Based on Melody and Lyric Information. In *Interspeech*, Dresden, Germany, September 2015.

[5] Rong Gong, Nicolas Obin, Georgi Dzhambazov, and Xavier Serra. Score-informed syllable segmentation for jingju a cappella singing voice with mel-frequency intensity profiles. In *International Workshop on Folk Music Analysis*, Málaga, Spain, June 2017.

[6] Y. Guédon. Hidden hybrid markov/semi-markov chains. *Computational Statistics & Data Analysis*, 49(3):663 – 688, 2005.

[7] Yann Guédon. Exploring the state sequence space for hidden markov and semi-markov chains. *Computational Statistics & Data Analysis*, 51(5):2379–2409, 2007.

[8] Z. Guo, Q. Wang, G. Liu, J. Guo, and Y. Lu. A music retrieval system using melody and lyric. In *2012 IEEE International Conference on Multimedia and Expo Workshops*, pages 343–348, July 2012.

[9] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] B. Juang, L. Rabiner, S. Levinson, and M. Sondhi. Recent developments in the application of hidden markov models to speaker-independent isolated word recognition. In *ICASSP*, volume 10, pages 9–12, Apr 1985.

[11] D. Kingma and J. B. Adam. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[12] A. M. Kruspe. Keyword spotting in singing with duration-modeled hmms. In *EUSIPCO*, pages 1291–1295, August 2015.

[13] Andrew L Maas, Peng Qi, Ziang Xie, Awni Y Hannun, Christopher T Lengerich, Daniel Jurafsky, and Andrew Y Ng. Building dnn acoustic models for large vocabulary speech recognition. *Computer Speech & Language*, 41:195–213, 2017.

[14] Marius Miron, Julio J. Carabias-Orti, Juan J. Bosch, Emilia Gómez, and Jordi Janer. Score-informed source separation for multichannel orchestral recordings. *JECE*, 2016, December 2016.

[15] E. Molina, L. J. Tardón, I. Barbancho, and A. M. Barbancho. The importance of f0 tracking in query-by-singing-humming. In *ISMIR*, Taipei, Taiwan, 2014.

[16] Jordi Pons and Xavier Serra. Designing efficient architectures for modeling temporal features with convolutional neural networks. In *ICASSP*, New orleans, USA, 2017.

[17] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *arxiv.org:1703.06697*, 2017.

[18] Hasim Sak, Andrew W. Senior, Kanishka Rao, and Françoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. *CoRR*, abs/1507.06947, 2015.

[19] J. Sohn, N. S. Kim, and W. Sung. A statistical model-based voice activity detection. *IEEE signal processing letters*, 6(1):1–3, 1999.

[20] C. C. Wang and J. S. R. Jang. Improving query-by-singing/humming by combining melody and lyric information. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(4):798–806, April 2015.

[21] L. Wang, S. Huang, S. Hu, J. Liang, and B. Xu. An effective and efficient method for query by humming system based on multi-similarity measurement fusion. In *International Conference on Audio, Language and Image Processing*, pages 471–475, July 2008.

[22] E. Wichmann. *Listening to Theatre: The Aural Dimension of Beijing Opera*. University of Hawaii Press, 1991.

AUTOMATIC INTERPRETATION OF MUSIC STRUCTURE ANALYSES: A VALIDATED TECHNIQUE FOR POST-HOC ESTIMATION OF THE RATIONALE FOR AN ANNOTATION

Jordan B. L. Smith
National Institute of Advanced Industrial Science and Technology (AIST), Japan
jordan.smith@aist.go.jp

Elaine Chew
Queen Mary University of London
elaine.chew@qmul.ac.uk

ABSTRACT

Annotations of musical structure usually provide a low level of detail: they include boundary locations and section labels, but do not indicate what makes the sections similar or distinct, or what changes in the music at each boundary. For those studying annotated corpora, it would be useful to know the rationale for each annotation, but collecting this information from listeners is burdensome and difficult. We propose a new algorithm for estimating which musical features formed the basis for each part of an annotation. To evaluate our approach, we use a synthetic dataset of music clips, all designed to have ambiguous structure, that was previously used and validated in a psychology experiment. We find that, compared to a previous optimization-based algorithm, our correlation-based approach is better able to predict the rationale for an analysis. Using the best version of our algorithm, we process examples from the SALAMI dataset and demonstrate how we can augment the structure annotation data with estimated rationales, inviting new ways to research and use the data.

1. INTRODUCTION

Listeners perceive structure in music, and trying to predict the structures they perceive is a popular task in the MIR community [14]. Since the perception of structure is a complex phenomenon, the community focuses on a simpler, operational version: we imagine that structure, as perceived, can be characterized as a set of time points regarded as boundaries, and a set of labels that indicate which of the intervening segments repeat similar material. This simplification is not made naïvely: those who create annotations of musical structure are aware of its limitations, and the methodologies for annotating [1, 16, 21] and evaluating [7, 9, 11] structural analyses have become their own important subtopics in MIR.

Still, the simplification is unfortunate because musical similarity is multi-dimensional. If a listener declares that two excerpts are "similar", they could mean with respect to melody, contour, rhythm, timbre, or any combination of these or other musical attributes. This is in addition to the issue that structure itself is multi-dimensional; as pointed out in [16], boundaries may be perceived for reasons of musical similarity, musical function, or instrumentation.

Thus, in the transition from structure perception to structure annotation, we usually fail to capture *why* a listener has included a boundary or chosen a label. This information, if preserved (or reconstructed), would help us to understand the content of the annotations, and could lead to fairer evaluations of structure segmentation algorithms. It would also provide more meaningful data to analyze in musicology or music perception research.

How feasible is it to collect this information? As we found in [23], to transcribe the rationale for every aspect of an annotation is difficult and requires prolonged self-interrogation. Even before that, it is difficult to decide what information to collect, and how to collect it: should the data be collected after a listener has provided the segmentation, in the manner of music perception experiments [2]? Or should each piece be annotated several times, each time with a focus on a single feature [19]? No matter how it is done, collecting this information is burdensome.

A more practical possibility is to estimate this information automatically from existing annotations, which was our motivation in [22]. Our algorithm compared self-distance matrices (SDMs) for different features to the ground truth annotation, and found which parts of the feature-based SDMs best re-created the annotation-based SDM. While [22] presented some examples to demonstrate the plausibility of the approach, we offered no experimental validation.

Validation requires paired responses: a set of listeners' analyses, and the listeners' justifications for each analysis. Producing this data is time-consuming and burdensome for the reasons described above. However, we recently produced data suited to this purpose for a music perception study [20]. The goal of that study was to determine what role attention plays in the perception of structure.

In this article, we make three main contributions: first, we test whether the approach described in [22] can effectively predict the attention of the listeners, based on the dataset created for [20]. Second, we explain some shortcomings of the previous approach, and suggest and test two improvements. Third, we demonstrate how the validated algorithm can be used to analyze and to augment real-world data with new information layers.

The next two sections recap the studies on which this article builds. In Section 2, we briefly recall [22]'s algorithm, point out some shortcomings, and introduce a refined approach. In Section 3, we summarize the results of the experiment in [20], and describe in more detail the data developed for that study and used in this one. In Sections 4 and 5, we outline the validation experiment and discuss the results, and in Section 6 we use the algorithm to create new information layers for examples from SALAMI [21]. We close with a few observations on the limitations of the present work and recommendations for future research.

2. AN ALGORITHM FOR ESTIMATING FEATURE RELEVANCE

In [22] we estimated the relevance of musical features to a listener's analysis section-by-section by finding the weighted sum of feature-derived SDMs that best matched the analysis. The analysis is represented as a binary SDM, expanded to the same timescale as the feature SDMs. A number n of feature matrices are computed; from each, we derive m single-section SDMs by taking only the rows and columns associated with that single segment, as defined by the annotation. (This row and column selection is done by multiplying the SDM with a segment mask.) This gives $n \cdot m$ component matrices. A quadratic program (QP) is used to find the weights for these components whose sum optimally reproduces the annotation-derived SDM; these weights, the reconstruction coefficients, are taken to indicate feature relevance.

The method is illustrated in Fig. 1. The sound example has the form ABAB with respect to harmony, AABB with respect to rhythm, and ABBA with respect to timbre. If a listener gives the analysis ABAB, segmenting the audio at the 1/4, 2/4 and 3/4 marks, we obtain the four segment masks given in the top row. We compute four audio features, each related to a different musical attribute (see Section 4.1 for details), which are pointwise multiplied by the masks to give 8 potential components. The QP finds the optimal combination of components to reproduce the annotation in the top-left corner, and gives the coefficients shown above each component. In this case, the algorithm has successfully identified that bass chroma is the feature that best justifies the analysis.

2.1 Algorithm Improvements

One limitation of this approach is that none of the feature matrices may properly reflect the homogeneity of a given section. We could include additional SDMs that have been smoothed at different timescales (as demonstrated in [22]), but the smoothing can blur the boundaries between sections even as they make the sections more homogeneous. We could use stripe-based instead of block-based masks in order to capture repetitions of feature sequences, but in

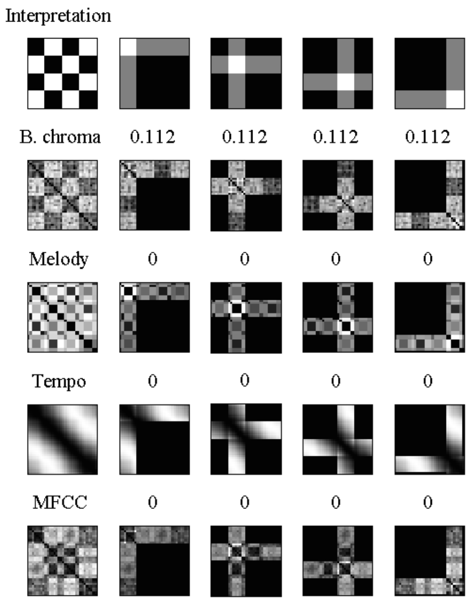


Figure 1. Illustration of component-building for QP algorithm. Four beat-indexed feature matrices (at left) are multiplied by the masks (top) given by the segmentation, which here is ABAB. The number above each component is the QP's estimate of the component's importance.

non-square blocks (which occur whenever two segments have unequal lengths), it is not easy to guess the best orientation or placement of the stripes.

A second problem is that it is unclear how to interpret some aspects of the QP. Should the individual reconstruction coefficients, or their sum, be bounded? Leaving them unbounded can lead to unconstrained solutions, but if bounds are imposed, how should they be interpreted?

A third problem is that by finding the single optimal *sum* of matrix components, some good explanations may be ignored. For example, if there are two matrix components which both justify a particular part of the analysis, the QP may find that only one is necessary. Thus, we cannot conclude that features omitted from the solution are necessarily irrelevant, which is a big limitation.

For the first problem, we propose that instead of using the original SDMs, with all their heterogeneities, we reduce them to segment-indexed SDMs, a common practice since [4]. Similar to [13], we may take the distance between each pair of segments to be the average distance of all the pixels in the submatrix over which the segments intersect. The segment-indexed SDM can then be analyzed with the QP as before, although with a substantial reduction in complexity.

A second way to address the problem is to use a diagonal stripe-based mask instead of a block-based mask. Since the diagonals are the most salient portions of the SDM, it makes sense to focus on reconstructing this portion of the SDM. Emphasizing stripes is a common SDM analysis technique, and a comparison of block and stripe features found that when boundaries were given, stripe fea-

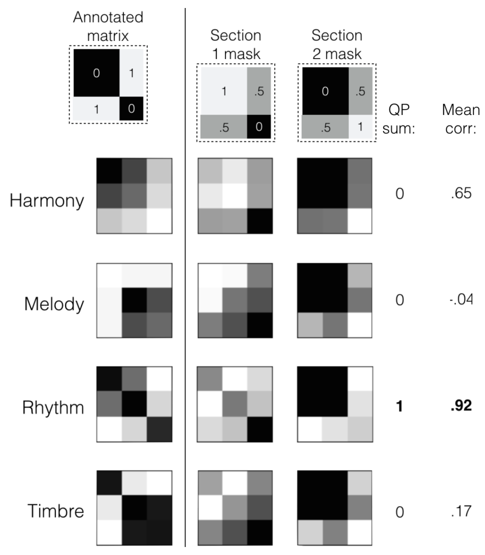


Figure 2. Illustration of proposed segment-indexed approach, with both QP- and correlation-based estimates of feature relevance.

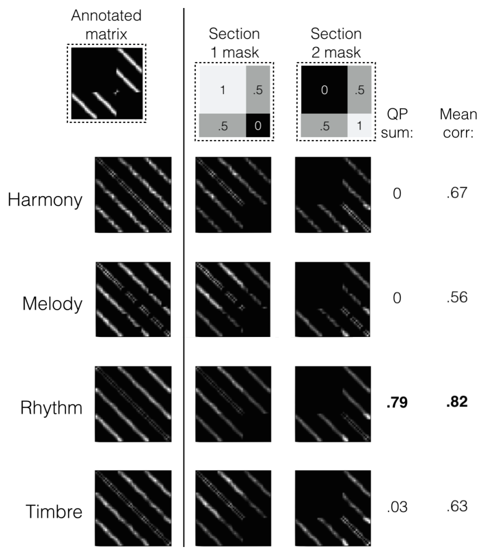


Figure 3. Illustration of proposed stripe approach with correlation. Like in Fig. 2, QP coefficients are in the middle column, correlations on the right.

tures were more effective [12]. We remember the caveat above, that repeated segments with different durations pose a problem for creating a stripe-based mask, but we can still test it in cases where this is not an issue.

A third proposal, which addresses the second and third problems above, is to dispense with the QP altogether and simply take the element-wise Pearson correlation between the feature-derived and annotation-derived matrices. (The same section-by-section method still applies.) Correlations are perhaps more intuitive than QPs and reconstruction coefficients, and using them would permit second-place features to be more readily identified in the solution.

Fig. 2 shows the output for an example of a three-part stimulus, using the suggested improvement of segment-indexing: the features have been averaged over the blocks given by the segmentation. The sum of the reconstruction coefficients obtained using the QP method are given in the "QP sum" column, and the mean point-wise correlation between the masked regions is shown in the "Mean corr." column. Fig. 3 shows the output for the same example but using the stripe-based mask. The mask is constructed by drawing a diagonal line across each block of the original beat-indexed SDM, and then applying a 2D convolution with a Gaussian kernel of width 5 beats.

To sum up, we suggest three improvements to the algorithm: (1) using the correlation between submatrices, instead of QP, to estimate their relevance; (2) using a segment-indexed version of the SDM; and (3) applying a stripe mask to the SDM, instead of using the blocks.

3. A DATASET OF VALIDATED ANALYSES

Researchers in music psychology, like those in MIR, are invested in modeling how listeners perceive structure. (For one discussion, see [15].) The goal of [20] was to determine whether listeners could be influenced to perceive different structures by manipulating the musical feature to which they paid attention. In order to test this, we composed a set of artificial musical stimuli in which four different features (harmony, melody, rhythm and timbre) were systematically changed at different times, creating musical passages with ambiguous forms. These four features were chosen because they figured most prominently in studies where listeners were asked to justify why they perceived a given boundary, such as [2].

The three-part stimuli had two potential structures, AAB or ABB, with different features changing at different times. For example, the passage in Fig. 4a has form AAB with respect to harmony, and form ABB with respect to melody. The four-part stimuli had three potential structures, AABB, ABAB or ABBA, so that at every boundary there were two features that changed. For example, in the passage in Fig. 4d, the rhythm and harmony both change after the second measure.

As stated above, validating the algorithm requires musical examples where listeners' analyses are paired with their justifications—i.e., with the musical attributes to which they were paying attention. Many datasets of structural analyses exist, but none indicate which musical attributes justify the analyses. Also, in typical pieces of music, attributes change frequently, to different extents, and often simultaneously. To validate this algorithm we should use music with known, controlled changes. Hence, artificial stimuli such as these are valuable resources to validate the algorithm: each passage contains precise change points related to known musical attributes; and the link between the attributes and the different forms has been affirmed by listeners in an experimental setting.

More artificial stimuli could be generated and tested in future work; this may be a convenient way to provide deep-learning algorithms with the quantity of labelled data

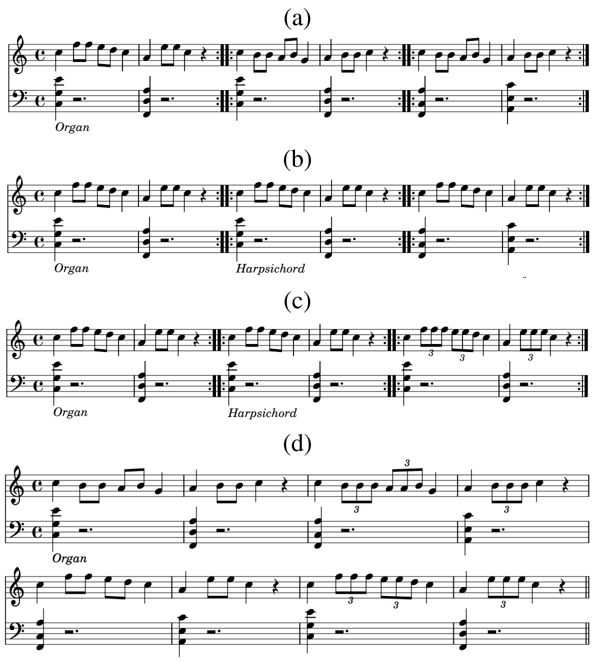


Figure 4. (a) Example stimulus with harmonic form AAB and melodic form ABB. (b) Harmonic form AAB, timbral form ABB. (c) Rhythmic form AAB, timbral form ABB. (d) Example four-part stimulis with melodic form AABB, rhythmic form ABAB, and harmonic form ABBA.

they require. However, it is not as simple as sonifying a symbolic score, since scores must be annotated in order to know the perceived structure and the musical features that motivate that analysis. The stimuli in our study are rare in that they were (1) composed so that musical features varied systematically, and (2) used in a listening experiment to validate that the intended structures were perceived, for the intended reason.

3.1 Stimulus Details

For [20], we composed three sets of stimuli. Each stimulus contains two voices, and in each set of stimuli, each voice potentially expresses changes in two different features. The examples in Fig. 4 are from the "HT-MR" set, where one voice expresses changes in harmony and timbre, and the other, changes in melody and rhythm. "HM-RT" and "HR-MT" sets were also composed.

Since in each set of stimuli, certain features are "convolved," some incorrect answers are less wrong than others. For instance, the feature that changes at the second boundary of the example in Fig. 4c is rhythm, but if an algorithm said that the boundary was justified by melody, it would be partially right.

The stems for the stimuli were composed using a Digital Audio Workstation with standard instrument patches. The 8 stems for each set were systematically recombined to generate 192 three-part stimuli and 384 four-part stimuli, for a total of 1728 stimuli among all sets. Efforts were made to keep constant all musical features other than harmony, melody, rhythm and timbre: the tempo of all stimuli is 140 bpm, and the loudness of each voice and each passage is approximately equal. The stimuli are now freely available on Github. [1]

4. EXPERIMENT

4.1 Features

The stimuli manipulated four different musical attributes (in three environments): harmony, melody, rhythm and timbre. We want to extract audio features that match each of these attributes independently. Each audio feature should change when the related musical feature changes, and be robust to changes in other musical features. We selected two audio features for each musical feature, all available as Vamp plugins [2] and listed in Tab. 1. We used ground truth beat locations, and median feature values were taken for each beat. Each dimension was normalized (independently for each stimulus) to zero mean, unit variance. All features were extracted using Sonic Annotator [3] using the default settings. For some features, we performed additional processing:

Chords: Chord labels were estimated from Chordino and reconverted back to a chroma-like representation. This feature is thus based on the same information as *bass chroma*, but refined with the chord-estimation algorithm.

Melody: The chroma of the estimated melody, and the interval between the current steady-state note and the previous one, each a 12-dimensional feature per frame. We also used the register of the melody: low, middle or high.

Autocorrelation: this was computed on an onset detection function with a sliding window.

Low level features: a concatenation of loudness, RMS amplitude, rolloff, sharpness, smoothness, tristimulus, zero-crossing rate, and the centroid, kurtosis, skewness, and slope of the spectrum.

Feature	***Vamp plugins used to obtain feature***
Harmony	*Bass chroma*, from Chordino and NNLS Chroma plugin [8] *Chord notes* [8]
Melody	*Treble chroma* [8] *Melody*, based on MELODIA [18]
Rhythm	*Cyclic tempogram* [6] *Autocorrelation*, based on UAPlugins's Note Onset Detector [17]
Timbre	*MFCCs* (2nd to 13th), from Chris Cannam and Jamie Bullock's LibXtract library *Low level features*, a set of fifteen one-dimensional descriptors from LibXtract

Table 1. List of features chosen, and Vamp plugins used to obtain them

[1] `https://github.com/jblsmith/music-structure-stimuli.`
[2] `vamp-plugins.org`

4.2 Results

We applied the algorithms, discussed in Section 2, on the stimuli discussed in Section 3. For each three-part stimulus, we ran the algorithms twice: once with analysis AAB, once with ABB. Likewise, we ran the algorithm thrice on each four-part stimulus to find the best justifications for forms AABB, ABAB and ABBA. Each algorithm takes one of these analyses as input. The output of each algorithm is a matrix of feature relevance values $x_{i,j}$: one per section i, per feature j. The importance of feature j is the sum across all the sections: $s_j = \sum_i x_{i,j}$. The importance of each musical attribute a is the sum of the two values s_j related to that feature: $y_a = s_{a1} + s_{a2}$. We end up with four values y_a.

We test whether the maximum value correctly predicts the feature related to the analysis with $\arg\max_a y_a$. The fraction of trials with correct guesses is the accuracy. Each trial has one focal pattern and three potential wrong answers, so the random baseline performance is 25%.

The five algorithm options were: whether to use cosine or Euclidean distance (in either case, the values were rescaled between 0 and 1); whether to compute beat-indexed or segment-indexed SDMs; whether to apply stripe-based masks to the SDMs; whether to use the QP or correlation-based approach; and finally, if using QP, what constraint to use. We tested three constraints: (a) $\sum_{i,j} x_{i,j} = C$ (the sum of the coefficients over the entire piece has a fixed value); (b) $\sum_j x_{i,j} = C$ (the sum of the coefficients for each section in the piece has a fixed value); and (c) $0 \leq x_{i,j} \leq 1$. These options were tested in a full-factorial design, replicated across three variables that were not part of the algorithm: the relevant feature; the music environment; and the stimulus length (3 or 4 sections).

We fit a linear model to the results and used ANOVA to interpret the eight factors. With 268,512 trials, three factors were insignificant ($p > 0.05$): stimulus length, distance metric, and QP constraint. The other five factors all had $p < 0.0001$, and main effect plots for each are shown in Fig. 5. They show that performance varied greatly among the music examples and features. However, the three proposed changes to the original algorithm—using correlation instead of QP, using stripe masks, and using segment-indexed SDMs—all saw improvements, albeit a minor one in the case of segment indexing.

Tab. 2 gives the accuracy for different parameter settings. It shows that although the main effects appear modest in Fig. 5, their impact is additive: the original approach achieved 47% accuracy, and the three changes (using correlation, segment-indexing, and applying a stripe mask) together raised the accuracy to nearly 70%.

These are the accuracies for choosing the most correct answer, but not all errors are equally bad: guessing a feature that was convolved in the stimulus with the correct one is sometimes a fair mistake. However, Tab. 2 shows that the "convolved-with-correct" answer was not given any special weight by the algorithms. There are 3 features besides the correct one, so the chance of randomly guessing the convolved feature is 33%. In all cases, fewer than a third of the incorrect answers related to the convolved feature.

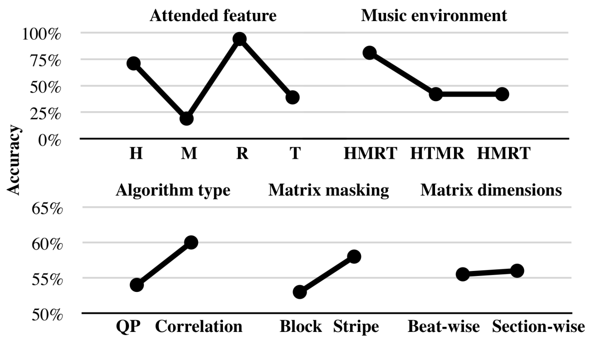


Figure 5. Main effect of significant factors on accuracy (i.e., rate of correct guesses).

Method:	**Quad. Prog.**		**Correlation**	
Settings:	Correct	Conv.	Correct	Conv.
Regular	47.1	13.7	52.3	12.8
Seg.-indexing	46.9	16.3	60.6	8.9
Stripe mask	52.4	13.5	62.4	11.9
Seg. and stripes	59.6	12.8	69.6	7.2

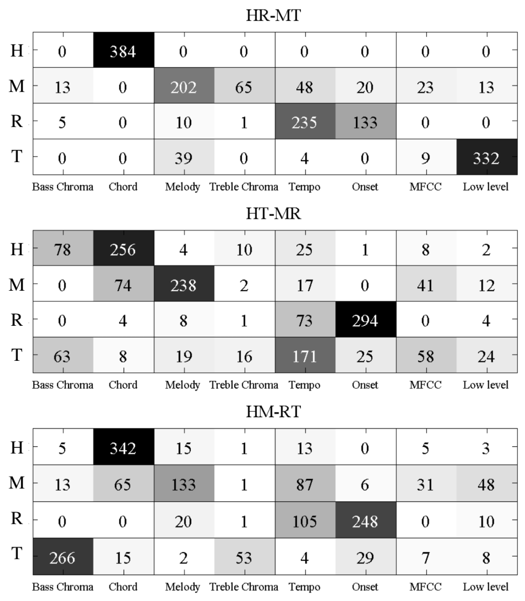
Table 2. Comparison of QP-based and correlation-based algorithms. Columns indicate how often the guessed feature was correct ("Correct") or convolved with the correct feature ("Conv."). For example, in the HT-MR environment, if the correct feature for a trial is timbre, guessing harmony could be half-right.

Prediction accuracy varied greatly among the features, as can be seen in the confusion matrices for the algorithms. Three are shown in Fig. 6, one for each music environment. These are the results for the best-performing algorithm. For harmony, we can observe that chord notes were more effective than bass chroma, the feature from which they derive. Bass chroma were especially misled in the HM-RT setting, possibly due to the difference in bass drum between the two timbre settings. With melody, it was also the case that the 2nd-order feature (the estimated predominant pitch and interval) was better than the lower-level feature (treble chroma).

5. DISCUSSION

The results validate the algorithm proposed in [22]. However, they also show that a simpler correlation-based approach is better at predicting how best to justify an analysis: it outperformed the QP approach by roughly 10%. Two other refinements, the stripe-based mask and the segment indexing, increased accuracy by roughly another 10%.

However, the confusion matrices revealed great disparities between the features we chose to use: some, such as Chordino, were effective; others, such as the tempogram and MFCCs, were often wrong. Arguably, it is naïve for us to presume that off-the-shelf features can detect the types of musical changes we created in the stimuli. Perhaps it is no accident that the four features we tweaked or assem-



HR-MT

	Bass Chroma	Chord	Melody	Treble Chroma	Tempo	Onset	MFCC	Low level
H	0	384	0	0	0	0	0	0
M	13	0	202	65	48	20	23	13
R	5	0	10	1	235	133	0	0
T	0	0	39	0	4	0	9	332

HT-MR

	Bass Chroma	Chord	Melody	Treble Chroma	Tempo	Onset	MFCC	Low level
H	78	256	4	10	25	1	8	2
M	0	74	238	2	17	0	41	12
R	0	4	8	1	73	294	0	4
T	63	8	19	16	171	25	58	24

HM-RT

	Bass Chroma	Chord	Melody	Treble Chroma	Tempo	Onset	MFCC	Low level
H	5	342	15	1	13	0	5	3
M	13	65	133	1	87	6	31	48
R	0	0	20	1	105	248	0	10
T	266	15	2	53	4	29	7	8

Figure 6. Confusion matrix for algorithm using correlation, stripe masks, and segment-indexed SDMs. The rows gives the correct musical attribute; the column indicates the audio feature with maximum relevance.

bled for this purpose (chord notes, MELODIA-based feature, onset autocorrelation and low-level features) tended to outperform their off-the-shelf rivals.

Still, the underperformance is surprising, since the stimuli are highly constrained: in the study for which the stimuli were created, listeners identified the attribute that changed at a boundary with 85% accuracy [20]. It seems reasonable to expect that, say, MFCCs will change more when a trumpet is swapped for a flute than when a trumpet plays a different melody; or that when the harmony changes, bass chroma will be more affected than the tempogram. Yet these are among the errors made by the features in this study. The results thus remind us of the utility of carefully-designed features, such as timbre-invariant chroma [10].

An alternative to testing hand-crafted features is to learn features with deep learning, but as mentioned earlier, this would require building a much larger, more representative stimulus set—more stimuli than can easily be validated in a listener study. The small set used here is suitable for testing existing features, but not learning new ones.

6. APPLICATION

The correlation algorithm can be used to interpret annotations in the SALAMI corpus [21]. We used the segment-indexing setting but not the stripe-masking, which (as noted in Section 2.1) is not applicable when unequal segment lengths give rectangular blocks. The audio processing was the same except that BeatRoot [5] was used to locate beats.

Fig. 7 visualizes a listener's analysis of "We Are The Champions" by Queen at the long and short timescales. Each vertical slice corresponds to a single section, and the brightness of each cell indicates the correlation of that feature to that section. We can see that on a long timescale, the verse sections (A) were characterized by their harmonic and melodic content, while the chorus sections (B) were characterized more by their timbre. However, on a short timescale, subsection a was also characterized by timbre, and many of the subsections of B were more strongly characterized by harmony and melody compared to B itself.

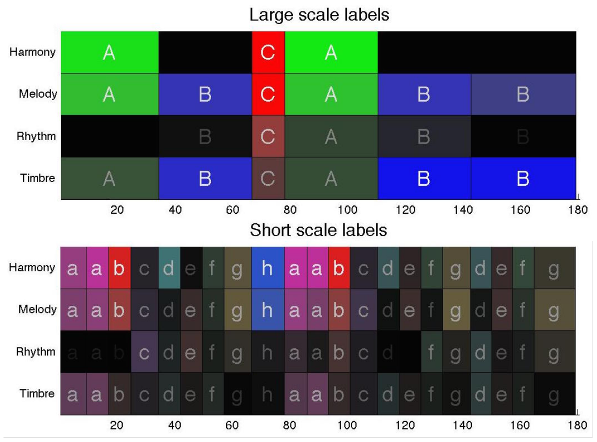


Figure 7. Example augmented annotation for the song "We Are The Champions" by Queen. The letters and colors both encode the section labels. Brightness indicates a feature's relevance to a section.

This, it turns out, is an accurate description of the song: in a, Freddie Mercury sings above a piano and bass only; the electric guitar enters quietly in b, but the drums come in with c in a raucous crescendo to the chorus. The timbral inconsistency of A means that timbre would be a poor feature to use to justify grouping the first four subsections into a larger unit.

On the other hand, the timbre of the choruses is relatively homogeneous; this makes it a good feature to justify grouping the B sections together, but also makes it a poor feature to justify giving the subsections of B different labels. The fact that subsections d, e, f and g have different labels must therefore reflect their pitch content.

7. CONCLUSION

We have validated the algorithm proposed by [22], and proposed three modifications to improve its effectiveness. Although we restricted this study to stimuli that were validated in a psychology experiment, it would be possible to generate large amounts of artificial music, with more complicated patterns of repetition and variation, and changes in more musical parameters, like loudness, tempo, syncopation, dissonance, and so on.

The accuracy of the algorithms fell short of human performance. Given the disparities among the features, this must be due in part to the mismatch between the audio features we chose and the musical attributes manipulated in the stimuli. Despite this, the algorithm is useful for visualizing the structure of pieces in a new way: by highlighting the musical features that explain the annotation.

8. REFERENCES

[1] Frédéric Bimbot, Emmanuel Deruty, Gabriel Sargent, and Emmanuel Vincent. Semiotic structure labeling of music pieces: Concepts, methods and annotation conventions. In *Proceedings of ISMIR*, pages 235–240, Porto, Portugal, 2012.

[2] Michael Bruderer, Martin McKinney, and Armin Kohlrausch. The perception of structural boundaries in melody lines of Western popular music. *Musicæ-Scientæ*, 13(2):273–313, 2009.

[3] Chris Cannam, Michael O. Jewell, Christophe Rhodes, Mark Sandler, and Mark d'Inverno. Linked data and you: Bringing music research software into the semantic web. *Journal of New Music Research*, 39(4):313–325, 2010.

[4] Matthew Cooper and Jonathan Foote. Summarizing popular music via structural similarity analysis. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 127–30, New Paltz, NY, United States, 2003.

[5] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30(1):39–58, 2001.

[6] Peter Grosche, Meinard Müller, and Frank Kurth. Cyclic tempogram - a mid-level tempo representation for music signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Dallas, TX, USA, 2010.

[7] Hanna Lukashevich. Towards quantitative measures of evaluating song segmentation. In *Proceedings of ISMIR*, pages 375–380, Philadelphia, PA, USA, 2008.

[8] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of ISMIR*, pages 135–140, Utrecht, Netherlands, 2010.

[9] Brian McFee, Oriol Nieto, and Juan Pablo Bello. Hierarchical evaluation of segment boundary detection. In *Proceedings of ISMIR*, Málaga, Spain, 2015.

[10] Meinard Müller and Sebastian Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):649–662, 2010.

[11] Oriol Nieto, Morwaread Farbood, Tristan Jehan, and Juan Pablo Bello. Perceptual analysis of the f-measure to evaluate section boundaries in music. In *Proceedings of ISMIR*, Taipei, Taiwan, 2014.

[12] Jouni Paulus and Anssi Klapuri. Acoustic features for music piece structure analysis. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 309–312, Espoo, Finland, 2008.

[13] Jouni Paulus and Anssi Klapuri. Music structure analysis using a probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech & Language Processing*, 17(6):1159–1170, 2009.

[14] Jouni Paulus, Meinard Müller, and Anssi Klapuri. Audio-based music structure analysis. In *Proceedings of ISMIR*, pages 625–636, Utrecht, The Netherlands, 2010.

[15] Marcus T. Pearce, Daniel Müllensiefen, and Geraint A. Wiggins. The role of expectation and probabilistic learning in auditory boundary perception: A model comparison. *Perception*, 39:1367–1391, 2010.

[16] Geoffroy Peeters and Emmanuel Deruty. Is music structure annotation multi-dimensional? A proposal for robust local music annotation. In *Proceedings of the International Workshop on Learning the Semantics of Audio Signals*, pages 75–90, Graz, Austria, 2009.

[17] Antonio Pertusa and José Manuel Iñesta. Note onset detection using one semitone filter-bank for mirex 2009. In *MIREX Audio Onset Detection*, Kobe, Japan, 2009.

[18] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20:1759–1770, 2012.

[19] Chris Sanden, Chad R. Befus, and John Z. Zhang. A perceptual study on music segmentation and genre classification. *Journal of New Music Research*, 41(3):277–293, 2012.

[20] Jordan B. L. Smith. Explaining listener differences in the perception of musical structure. September 2014.

[21] Jordan B. L. Smith, J. Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J. Stephen Downie. Design and creation of a large-scale database of structural annotations. In *Proceedings of ISMIR*, pages 555–560, Miami, FL, United States, 2011.

[22] Jordan B. L. Smith and Elaine Chew. Using Quadratic Programming to estimate feature relevance in structural analyses of music. In *Proceedings of the ACM International Conference on Multimedia*, pages 113–122, Barcelona, Spain, 2013.

[23] Jordan B. L. Smith, Isaac Schankler, and Elaine Chew. Listening as a creative act: Meaningful differences in structural annotations of improvised performances. *Music Theory Online*, 20(3), 2014.

AUTOMATIC PLAYLIST SEQUENCING AND TRANSITIONS

Rachel M. Bittner, Minwei Gu, Gandalf Hernandez, Eric J. Humphrey,
Tristan Jehan, P. Hunter McCurry, Nicola Montecchio
Spotify Inc., USA

ABSTRACT

Professional music curators and DJs artfully arrange and mix recordings together to create engaging, seamless, and cohesive listening experiences, a craft enjoyed by audiences around the world. The average listener, however, lacks both the time and the skill necessary to create comparable experiences, despite access to same source material. As a result, user-generated listening sessions often lack the sophistication popularized by modern artists, *e.g.* tracks are played in their entirety with little or no thought given to their ordering. To these ends, this paper presents methods for automatically sequencing existing playlists and adding DJ-style crossfade transitions between tracks: the former is modeled as a graph traversal problem, and the latter as an optimization problem. Our approach is motivated by an analysis of listener data on a large music catalog, and subjectively evaluated by professional curators.

1. INTRODUCTION

DJs are modern artists that carefully select, sort, and combine recordings in order to enhance the music listening experience over simpler forms, such as albums or playlists. They traditionally create *mixes* or *sets* that flow seamlessly from one song to the next by sequencing styles, matching keys and tempos, and smoothly transitioning between musical ideas. Importantly, the ordering of tracks or samples and the quality of the transitions between them are fundamentally linked: it can be very difficult to create an enjoyable transition between songs that significantly differs in style, tempo, or key. Transitioning between a slow, smooth jazz piece and a high energy, fast electronic track, for example, will likely feel awkward or unnatural and create an abrupt change in the listening experience.

Though listening to DJ mixes is not a new phenomenon, modern music streaming services indicate that there is significant appetite among users for curating their own sets, having produced over 2 billion playlists in the last decade on Spotify alone. [1] To develop a vague sense of how many users might be aspiring "DJs" in the home or car, we find that roughly 1% of the public playlists available through Spotify's Web API contain "party" in the title. [2] Even through coarse extrapolation, this suggests that some 20M playlists are candidates for DJ-style production.

Therefore, given that so many users are actively exercising their curatorial skills, the steady advance of machine listening technology offers promise that the more technical challenges of creating a DJ mix could be achieved computationally. In this paper we focus specifically on the two hurdles faced in creating a DJ set from a given playlist: compute an optimal sequencing, and create song-to-song transitions between sequenced tracks. One of the challenges of building a model for these two tasks is defining how to evaluate performance. Because quality of a song sequence and of a song-to-song transition is highly subjective, we rely on user studies to evaluate the performance of our systems.

2. RELATED WORK

Several commercial products (e.g., Serato DJ [3] and Native Instruments' Traktor line [4]) are designed to assist DJs with digital mixing on a laptop. These are mainly tools for enthusiasts and professionals who already have experience in mixing, and as such these tools tend to replicate with software their original analog counterparts. Automatic audio analysis techniques are sometimes exploited to let the user sort playlists by tempo and key, however by design it is up to the DJ to make a final selection and decide on where to transition: the software's role is to assist with time-stretching and facilitating the execution of beat-aligned transitions. This paper is concerned with the automation of the entire experience, demanding less involvement by the users; examples of commercial software in this category include Algoriddim DJay [5] , Pacemaker [6] , and Serato Pyro [7] .

Sequential ordering is the primary concern of [6], that uses an audio similarity metric built on Gaussian models of MFCCs. However, the approach does not constrain the problem to a pre-selected set of songs and instead generates playlists from a large pool. In analyzing the order-

[1] https://press.spotify.com/us/about/

[2] `https://developer.spotify.com/web-api/playlist-endpoints/`
[3] `https://serato.com/dj`
[4] `https://www.native-instruments.com/en/products/traktor`
[5] `https://www.algoriddim.com/`
[6] `https://pacemaker.net/`
[7] `https://seratopyro.com/`

ing of songs in professionally-made DJ sets, [11] presents evidence that timbral factors play an important role in sequencing. In [3], consideration is given to "tempo trajectories" over time as a way of modeling human DJs' ability to structure the rise and fall of energy levels in the music as the sequence of songs progresses. Ishizaki, et al. [9] focus on making smooth tempo adjustments to songs with the goal of minimizing abrupt changes that could cause listener discomfort. In choosing optimal mixing regions between two songs, [8] employs section similarity metrics derived from chroma information, along with beat and tempo features. Similarly, [7] proposes a model for measuring the perceptual consonance for different transition regions given two tracks. In [15], a more complete DJ simulation method is proposed, which performs song selection, ordering and cross-fading for electronic music. A closely related problem is the automatic creation of musical "mash-ups", for which a number of algorithms have recently been proposed [5, 12].

3. SEQUENCING

Given a playlist, the goal of a sequencing algorithm is to order the tracks it contains in a way as to make the music "flow smoothly" from each song to the next. Cunningham et al. performed an in-depth study of how individual users sequence tracks, and concluded that the task is "more of an art than a science" [4]. Thus, the notion of flow and its attainment is ultimately an aesthetic phenomenon; a DJ may want the tempo to stay relatively constant or neighboring songs to be acoustically similar as a function of creative intent, as illustrated in Figure 1. If songs are to be cross-faded, proper sequencing can ensure that consecutive pairs of songs have similar keys and tempos, allowing for less abrupt transitions. Understandably, the scope of this work entails a more calculated approach than that of an expert DJ, and we identify artist-quality sequencing as a broader aim of this research area. It is important to note that this problem is related to, but different from the task of *generating* playlists, for example as in [2] – in this task we are given a list of tracks and the task is to reorder them, rather than to find a list of coherent tracks from a large corpus.

Examples of playlists sequenced using the proposed approach can be found online. [8] [9]

3.1 Method

The problem of sequencing tracks lends itself well to be formulated in a graph theory setting. The central step consists in mapping acoustic features into a Euclidean space so that songs that are fit to be sequenced next to each other are also close together in the feature space. Finding a good sequencing involves finding the shortest non-repeating path between all the songs.

[8] https://open.spotify.com/user/rabitt3/playlist/6a4lxKlqWZwKQgV3VhRMjX

[9] https://open.spotify.com/user/rabitt3/playlist/0Cl1BNwnWxmLkfUn8YQZVS

Original Playlist		
Title	**Artist**	**Tempo**
All Star	Smash Mouth	104
...Baby One More Time	Britney Spears	92
Bills, Bills, Bills	Destiny's Child	127
Every Morning	Sugar Ray	109
Genie In A Bottle	Christina Aguilera	175
I Want It That Way	Backstreet Boys	99
Livin' la Vida Loca	Ricky Martin	178
Miami	Will Smith	108
No Scrubs	TLC	92
Smooth	Santana, Rob Thomas	115

Sequenced by Tempo & Timbre		
Title	**Artist**	**Tempo**
...Baby One More Time	Britney Spears	92
I Want It That Way	Backstreet Boys	99
All Star	Smash Mouth	104
Every Morning	Sugar Ray	109
Smooth	Santana, Rob Thomas	115
Livin' la Vida Loca	Ricky Martin	178
Genie In A Bottle	Christina Aguilera	175
Bills, Bills, Bills	Destiny's Child	127
No Scrubs	TLC	92
Miami	Will Smith	108

Figure 1: Example playlist sequencing by tempo and timbre.

3.1.1 Constructing the Feature Space

Several acoustic aspects of a song are exposed so that they can be combined differently:

- *acoustic vectors* are created by first using a convolutional neural network [16] trained to reproduce collaborative-filtering vectors in ($\mathbb{R}^{2048}$). The acoustic vectors are low dimensional embeddings ($\mathbb{R}^{2048} \Rightarrow \mathbb{R}^{8}$) of the output of the convolutional neural network, where the embedding was trained to minimize the Euclidean distance between artists. These features mostly capture the timbral character of a song.
- *key and mode* information from the Echonest analyzer is mapped into points in $\mathbb{R}^3$ so that adjacent keys in the circle of fifths and relative major/minor keys are equidistant, as pictured in Figure 2: Left.
- *tempo* (originally in beats per minute estimated from the Echonest analyzer) is represented in a base-2 logarithmic scale. In certain applications it is desirable to preserve tempo-octave invariance: in that case tempo is represented as a unit vector whose polar angle is mapped into a tempo octave, as in Figure 2: Right.

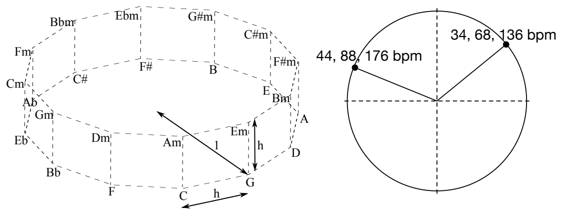


Figure 2: Left: key/mode mapping to Euclidean space. Right: octave-invariant tempo mapping to Euclidean space.

A feature vector is finally constructed by concatenating the above individual feature vectors, each feature is optionally weighted according to the application (e.g., in dance playlist, tempo coherence is important, thus tempo would have a large weight).

3.1.2 Solution as a Graph Problem

Let us consider the complete symmetric graph in which each song is associated to a vertex, and edges are weighted by the Euclidean distance between the corresponding songs' features.

A *Hamiltonian Path* is a path that visits each vertex in a graph exactly once. The optimal sequencing of a playlist corresponds to the *Shortest* Hamiltonian Path in the (complete) graph, a problem which is unfortunately NP-Complete (the total cost of an ordering is the sum of all the weights of the edges in the path). Several approximation strategies, shown below, have been considered; their computational cost is dominated by the construction of the weight matrix, quadratic in the length of the playlist.

A straightforward greedy approximation (which we denote by *HAM-1*) consists of iteratively selecting the closest non-visited vertex, starting from a given *seed* vertex. An improvement (*HAM-2*) can be made by selecting the closest non-visited vertex from either the tail or the head of the partial sequencing.

Empirically, both methods give satisfying results; the total cost of a *HAM-2* sequencing is virtually always lower (better) than its *HAM-1* counterpart, although the seed track does not end up as the head of the sequencing anymore (which could be itself a desirable feature). An undesirable artifact is the presence of poor track pairings at the tail of the sequencing for *HAM-1* and at both ends for *HAM-2*, due to the greedy nature of the algorithm.

A different solution is given by the Shortest Hamiltonian *Cycle*, an NP-complete problem (also known as the *Traveling Salesman Problem*) which however admits a polynomial 2-approximation [13]. The cost is usually higher than either of the greedy Hamiltonian Path solutions, but the resulting playlist will have smooth transitions, even when repeated in a loop, and is free from the head and tail artifacts described above.

3.2 Evaluation

To measure the effectiveness of the sequencing algorithm, we ran a pilot study in which professional curators blindly compared six sequenced vs. randomly sequenced playlists. Each of the six playlists contained 30 "Discover Weekly" playlists. The sequenced version of the playlist was created using *HAM-2* with acoustic vectors as features. For each of the six playlist pairs, the curators were instructed to (1) choose which playlist was sequenced better, and (2) list the pairs of tracks in each playlist that were deemed "abrupt" when played sequentially.

In the first task, for playlists 1, 2, and 5 the curators unanimously chose the sequenced playlist over the random playlist. For playlists 3 and 4, the curators were evenly split showing no preference, and for playlist 6, half preferred the sequenced, and half had either no preference or preferred the random playlist. Table 1 shows the average number of "abrupt" pairs of tracks across curators for each playlist. As expected there were more abrupt pairs in the random versions than in the sequenced versions. This is particularly drastic for playlist 5, probably due to the wider range of genres.

Playlist	Genres	Random	Sequenced
1	*Folk Pop, Country*	2.8 (1.8)	1.2 (1.3)
2	*Underground Hip-Hop, Funk*	3.8 (4.3)	1.2 (1.3)
3	*Abstract Hip-Hop, Indietronica*	2.7 (2.0)	3.3 (2.2)
4	*Indietronica, Indie Rock*	2.8 (1.9)	2.8 (3.7)
5	*Jazz, Classical, House*	9.3 (1.5)	2.7 (1.2)
6	*Folk Metal, Death Metal*	4.00 (3.6)	3.50 (2.3)
	Average	**4.2 (2.5)**	**2.4 (1.0)**

Table 1: Average number of song pairs (out of a total of 29 pairs) marked as "abrupt" across curators. The standard deviation is indicated in parentheses.

4. TRANSITIONS

Various streaming services provide, as a toggleable feature, a simple fixed-length crossfading between tracks; this however does not take content into account. About 95% of the users of Spotify forgo the option, and use standard end-to-end playback. To motivate the inclusion of transitions in a playlist, an A-B test was run on 10% of users of Spotify, where DJ-curated transitions were added to several popular playlists for the test group. The results showed that the percentage of people who returned to the playlists per day was 1.4 percentage points higher for the test group than control, suggesting that the listeners enjoyed the playlists with DJ curated transitions more and were thus more likely to listen again.

The goal of the algorithm we present is to create interesting DJ-like transitions between pairs of songs, which could be offered as an enhanced alternative to the existing crossfade. This involves choosing where in each track the transition will occur given a fixed transition length (in units of number of beats).

4.1 Method

Given a pair of tracks and a target transition length, our method selects transition start and end points in both songs, and uses this information to render the transition. Transition locations are restricted to downbeats, and are heavily weighted to occur on section boundaries, such as at the intersaection of a verse and a chorus. Additionally, we assume that regions of tracks that have similar timbre and pitch distributions will yield the smoothest transition. In this work, we only consider music in quadruple meter.

A symmetric crossfade, depicted in Figure 3, is arguably the most basic kind of transition: $t_1^{(A)}$ and $t_1^{(B)}$ denote the fade out start and end points in track 1, and $t_2^{(A)}$ and $t_2^{(B)}$ denote the fade in start and end points in track 2; the duration of the *transition region*, the interval $[t_i^{(A)}, t_i^{(B)}]$, for track 1 and 2 need not be equal.

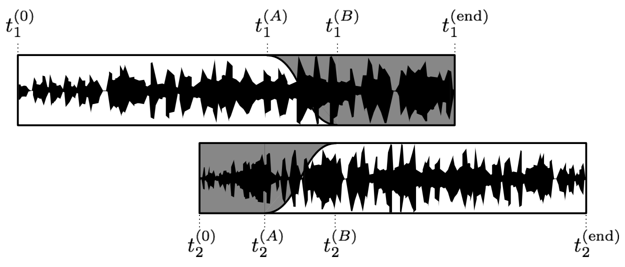


Figure 3: Sample crossfade transition. $t_1^{(A)}$ and $t_1^{(B)}$ mark the start and end points of the fade out for track 1. Similarly, $t_2^{(A)}$ and $t_2^{(B)}$ mark the start and end points of the fade in for track 2.

4.1.1 *Features*

Unless otherwise stated, each of the following features are computed for each track using the Echo Nest Analyzer, which is largely based on [10]. Let $\mathbf{b}$ be a list of estimated beat positions in seconds. Given the beat positions of each track, we compute several different types of event locations, each on the same time grid as the estimated beats. Let M be the set of indices of $\mathbf{b}$ which are downbeats. Similarly, let S be the set of indices of $\mathbf{b}$ which are section boundaries, and D be indices which are "drop" points. Section boundaries are computed using the method described in [14], and the "drop" point estimation is described in Section 4.1.2.

When choosing transition points, not all beats are created equal: the best transition points occur at strong structural boundaries. Each type of event location has a different level of structural significance in the track. The strongest structural boundaries, if they exist, are at the drop points. The next strongest points are section boundaries, followed by downbeats. Ideally, all drop points are section boundaries, and all section boundaries are downbeats, but this may not be the case.

In addition to these event locations, we compute several beat-synchronous features. Let N be the number of beats. Timbre features $\mathcal{T}$ are a (12 x N) matrix describing the spectral shape of each beat, and the chroma features C are a (12 x N) matrix giving the pitch class distribution for each beat. Loudness features ℓ and "vocalness" features v give the loudness and probability of vocals for each beat, and are each size (1 x N). Intuitively, transition regions with low loudness can often sound awkward and abrupt, and when vocals are present we risk overlapping vocals with the other track, or cutting over mid-sentence.

4.1.2 *Drop Point Estimation*

The goal in drop point estimation is to find the points in a track where the "drop" happens. The term "drop" is typically used in the context of specific types of electronic dance music, and refers to the point(s) in time where a drastic change in the song occurs following a large build. In our context, we are looking for points in a song where an exceptionally interesting event occurs. Rather than take a content-based solution similar to [17], we use a crowd-sourced approach following from the work described in P. Lamere's blog [10]. Lamere computes the points where users moved (scrubbed) the playhead while listening to a track. Typically users tend to move the playhead towards the most interesting points in the track. Figure 4 (top) shows an example of the aggregated playhead scrubbing data (blue) for *Skrillex: "First of the Year"*. The large peak occurring around 66 seconds accurately marks the first big drop, and the second smaller peak around 145 seconds marks the second big drop.

To identify these peak locations, we use a standard peak picking approach from the onset detection literature [1]: an adaptive threshold (shown in green) is computing using a median filter, then a detection function subtracts the adaptive threshold from the normalized scrub ratio and selects its peaks, as shown in Figure 4 (bottom). Choosing the closest downbeat that occurs before each resulting peak gives us our final drop index D. Note that in Figure 4 there is a small peak near the start of the track which is not a significant musical point. We correct for this by removing peaks that occur within the first 15 seconds of the track.

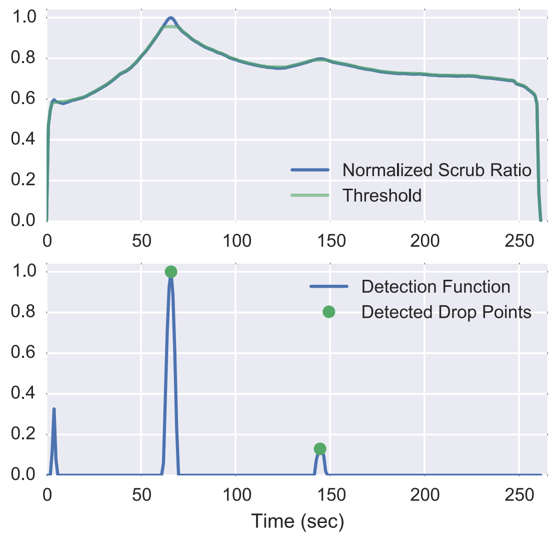


Figure 4: Drop point estimation intermediate steps for *Skrillex: "First of the Year (Equinox)"*. **Top**: Normalized scrub ratio and adaptive threshold. **Bottom**: Detection function and detected drop points. The first peak in the detection function is not a drop point because it occurs within the first 15 seconds of the track.

[10] `http://musicmachinery.com/2015/06/16/the-drop-machine/`

4.1.3 Selecting Transition Points

The procedure for selecting transition points between track 1 (T_1) and track 2 (T_2) of length n beats is outlined in Algorithm 1. The functions `beats` and `features` are described in Section 4.1.1.

Let $\mathbf{t}_1$ and $\mathbf{t}_2$ be the set of transition point candidates from which $t_1^{(A)}$ and $t_2^{(A)}$ will be selected. Since we are given a transition duration (in units of number of beats), $t_1^{(B)}$ and $t_2^{(B)}$ can be determined from the values of $t_1^{(A)}$ and $t_2^{(A)}$. Initially, we set $\mathbf{t}_1 = M_1$ and $\mathbf{t}_2 = M_2$.

We prune these sets to ensure that the transition points happen in reasonable portions of the track, removing obvious "bad" regions. The pruning is performed using the following rules:

- $t_1^{(B)}$ occurs before the fade out, $t_2^{(A)}$ is after the fade in
- $t_1^{(B)}$ occurs within the last 25% of the track, $t_2^{(A)}$ occurs within the first 20% of the track.

After pruning, the remaining points in $\mathbf{t}_1$ and $\mathbf{t}_2$ are considered valid candidates. These pruned sets are the output of the `candidates` function.

For each pair of points in $\mathbf{t}_1$ and $\mathbf{t}_2$, we compute pairwise comparisons along a series of different features over the entire overlapping region. For a transition of length n beats, the overlapping region begins at beats i and j, and ends at beats $i + n$ and $j + n$. In Algorithm 1 beginning at line 9, we use the notation $\mathcal{T}_1[i : i_n]$ to denote features within the region beginning at beat i and ending at beat i_n. Let Λ be the combined transition point cost matrix, where one axis represents the beat indices of track 1 and the second of track 2. Let Λ_x be the transition cost matrix for a particular feature comparison x. For timbre and chroma features, we compute $\Lambda_{\mathcal{T}}$ and Λ_C as the Euclidean distance between the features directly (Algorithm 1 lines 9, 10). Λ_ℓ (line 11) is computed as the sum of the average inverse loudness for each track, giving regions that are loud in both tracks a low transition cost. Similarly, Λ_v is the sum of the average "vocalness" probability, to assign transitions that both have vocals a high cost. Finally, we penalize transitions that do not end on a drop or a second boundary in both tracks (lines 13, 14), with a score of 2 if neither track's region ends on a boundary, and a score of 1 if only one track's region ends on a boundary.

Each feature's individual cost matrix Λ_x is standardized so that the minimum cost is 0 and the maximum cost is 1. The final cost matrix Λ is computed as a weighted sum of each feature's cost matrix after standardization. An example of each of feature's standardized matrix is shown in Figure 5, and the weighed combination is shown in Figure 6. The final transition points $t_1^{(A)}$ and $t_2^{(B)}$ are chosen as the times corresponding to the minimum cost entry in Λ.

Algorithm 1 Picking Transition Points

```
1:  procedure TRANSITION-POINTS($T_1, T_2, n$)
2:      $\mathbf{b}_1 \leftarrow \texttt{beats}(T_1), \mathbf{b}_2 \leftarrow \texttt{beats}(T_2)$
3:      $\mathcal{T}_1, C_1, \ell_1, v_1, M_1, D_1, S_1 \leftarrow \texttt{features}(T_1, \mathbf{b}_1)$
4:      $\mathcal{T}_2, C_2, \ell_2, v_2, M_2, D_2, S_2 \leftarrow \texttt{features}(T_2, \mathbf{b}_2)$
5:      $\mathbf{t}_1 \leftarrow \texttt{candidates}(T_1, M_1, S_1, D_1, \ell_1)$
6:      $\mathbf{t}_2 \leftarrow \texttt{candidates}(T_2, M_2, S_2, D_2, \ell_2)$
7:      for $i \in \mathbf{t}_1, j \in \mathbf{t}_2$ do
8:          $i_n \leftarrow i + n\; j_n \leftarrow j + n$
9:          $\Lambda_{\mathcal{T}}[i,j] \leftarrow \texttt{norm}(\mathcal{T}_1[i : i_n] - S_2[j : j_n])$
10:         $\Lambda_C[i,j] \leftarrow \texttt{norm}(C_1[i : i_n] - C_2[j : j_n])$
11:         $\Lambda_\ell[i,j] \leftarrow \texttt{avg}\,(2 - (\ell_1[i : i_n] + \ell_2[j : j_n]))$
12:         $\Lambda_v[i,j] \leftarrow \texttt{avg}(v_1[i : i_n]) + \texttt{avg}(v_2[j : j_n])$
13:         $\Lambda_D[i,j] \leftarrow 1_{i_n \notin D_1} + 1_{j_n \notin D_2}$
14:         $\Lambda_S[i,j] \leftarrow 1_{i_n \notin S_1} + 1_{j_n \notin S_2}$
15:     end for
16:     $\Lambda \leftarrow [\Lambda_{\mathcal{T}}, \Lambda_C, \Lambda_\ell, \Lambda_v, \Lambda_D, \Lambda_S]$
17:     for $k \in \Lambda$ do
18:         $k \leftarrow \texttt{standardize(k)}$
19:     end for
20:     $\Lambda \leftarrow \texttt{weightedAvg}(\Lambda_{\mathcal{T}}, \Lambda_C, \Lambda_\ell, \Lambda_v, \Lambda_D, \Lambda_S)$
21:     $i^*, j^* \leftarrow \texttt{argmin}(\Lambda)$
22:     $t_1^{(A)}, t_2^{(A)} \leftarrow \mathbf{b}_1[i^*], \mathbf{b}_2[j^*]$
23:     return $t_1^{(A)}, t_2^{(A)}$
24: end procedure
```

4.2 Rendering Transitions

Transitions are rendered such that the beats in the two tracks occur at the same time. In virtually every case, the tempos are not perfectly in sync, each beat is timestretched such that the tempo slowly changes from the tempo of track 1 to the tempo of track 2. For an N beat transition, if the nth beat in track 1 has duration d_1 and the beat in track 2 has duration d_2, the total duration of the new nth beat is $d_{\text{out}} = \frac{N-n}{N} d_1 + \frac{n}{N} d_2$. To achieve this, the nth beat in track 1 is time stretched by a factor of d_1/d_{out}, and the n^{th} beat in track 2 by d_2/d_{out}.

4.3 Evaluation

A selection of rendered transitions were evaluated by subjective human review. We randomly picked 48 pairs of tracks from a selection of popular music across multiple dance genres, using tempo constraints when picking the tracks to make sure the tempo difference between pairs was no more than 5 bpm.

For each of the pairs, we asked four professional curators to listen to the transition all the way through at least once and rate the quality. For subjective measurement, the overall quality is described as Good (3), OK (2) and Bad (1). Additionally, curators were asked to describe any potential problems they noticed within the transitions, such as "beats do not align" or "key clash". The results are shown in Tables 2 and 3, respectively.

A fairly large number (15%) of transitions were marked as "Bad" because the "beats do not align". Since we constrain transitions to align along estimated beats, we conclude that the "beats do not align" transitions occur as a result of errors in the beat estimation algorithm. Transitions labeled as "transitioning mid-vocals" are also likely a result of errors in our vocal activity detection algorithm. In

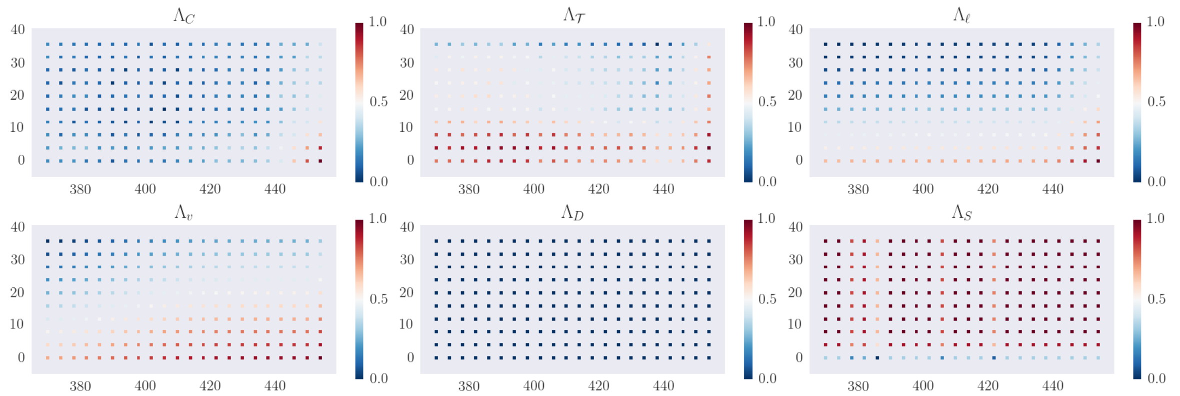


Figure 5: Transition matrices for each feature for a pair of songs. The x-axis show beat indices in Track 1, and the y-axis for Track 2. Many index pairs have no score because they are not part of the set of candidates. Dark blue points indicate good transition pairs for the feautre, while red indicates a poor pair. In this example, no drops were detected, so Λ_D is a uniform matrix.

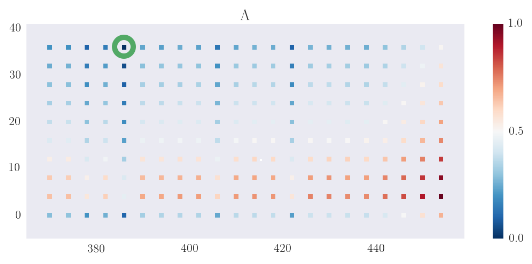


Figure 6: Weighted combination Λ of the individual feature matrices in Figure 5.The x-axis show beat indices in Track 1, and the y-axis for Track 2. The point with the lowest cost is circled in green.

Rating	Percentage
3 - Good	64.13%
2 - OK	28.26%
1 - Bad	7.61%
Average (Std) Rating	2.56 (0.38)

Table 2: Average percentage of quality rating for all track pairs and average rating of song pairs in rendered transition test set. The standard deviation is indicated in parentheses.

Reason	Percentage
Beats do not align	15.22%
Not on downbeat	2.17%
Key clash	0%
Awkward transition points	2.17%
Transitions mid-vocals	6.52%
Contrasting Songs	4.34%

Table 3: Average percentage of song pairs marked as the stated reason for bad quality transitions by curators.

both of these cases, as beat tracking and vocal activity detection algorithms improve, these transition quality issues should be mitigated. An interesting finding is that "key clash" is not marked as problematic by any of the curators for a single transition in either transition types.

5. CONCLUSIONS

This paper has presented systems for automatically sequencing and generating DJ-style transitions for a playlist of songs. Both systems were evaluated with the help of professional curators. Beat and downbeat tracking errors were found to be the primary bottleneck in the subjective performance of automated transitions.

A possible alternative approach for tackling the sequencing and transitioning problems entails the usage of Machine Learning approaches. Given a large number of (carefully curated) playlists and transition points between them, one might attempt to directly learn the mapping of low-level audio representation of recordings into their optimal sequencing and transitions. Such an methodology is certainly fascinating, and represents in fact a future research direction. However, the experiments above prove how just a few *interpretable* features are suitable for this problem to a remarkable extent. We chose then to investigate an approach that is heuristic in nature, but whose particular behavior can be customized by the user in an extremely intuitive manner (e.g., weighting acoustic similarity more than key and tempo might be preferred when constructing a playlist for a radio show, while the reverse is true in the case of a dancing playlist).

Finally, this work has focused on specific genres of music – namely "party" music. The constraints we imposed may not be necessary or sufficient for other genres of music, for example rap. However, the same framework could be applied substituting different features in the optimization problem. The exploration of how to apply this model to other genres is left as future work.

6. REFERENCES

[1] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler. A tutorial on onset detection in music signals. *Speech and Audio Processing, IEEE Transactions on*, 13(5):1035–1047, 2005.

[2] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722. ACM, 2012.

[3] Dave Cliff. Hang the DJ: Automatic sequencing and seamless mixing of dance-music tracks. *HP LABORATORIES TECHNICAL REPORT HPL*, 104, 2000.

[4] Sally Jo Cunningham, David Bainbridge, and Annette Falconer. 'more of an art than a science': Supporting the creation of playlists and mixes. In *ISMIR*, pages 240–245, 2006.

[5] Matthew EP Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. Automashupper: Automatic creation of multi-song music mashups. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(12):1726–1737, 2014.

[6] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist generation using start and end songs. In *ISMIR*, pages 173–178, 2008.

[7] Roman B Gebhardt, Matthew EP Davies, and Bernhard U Seeber. Psychoacoustic approaches for harmonic music mixing. *Applied Sciences*, 6(5):123, 2016.

[8] Tatsunori Hirai, Hironori Doi, and Shigeo Morishima. Musicmixer: Computer-aided dj system based on an automatic song mixing.

[9] Hiromi Ishizaki, Keiichiro Hoashi, and Yasuhiro Takishima. Full-automatic dj mixing system with optimal tempo adjustment based on measurement function of user discomfort. In *ISMIR*, pages 135–140, 2009.

[10] Tristan Jehan. *Creating music by listening*. PhD thesis, Massachusetts Institute of Technology, 2005.

[11] Thor Kell and George Tzanetakis. Empirical analysis of track selection and ordering in electronic dance music using audio feature extraction. In *ISMIR*, pages 505–510, 2013.

[12] Chuan-Lung Lee, Yin-Tzu Lin, Zun-Ren Yao, Feng-Yi Lee, and Ja-Ling Wu. Automatic mashup creation by considering both vertical and horizontal mashabilities. In *ISMIR*, pages 399–405, 2015.

[13] Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2):498–516, 1973.

[14] B. McFee and D. P. W. Ellis. Analyzing song structure with spectral clustering. In *ISMIR*, 2014.

[15] Jaume Parera. Dj codo nudo: a novel method for seamless transition between songs for electronic music. Master's thesis, Universitat Pompeu Fabra, Barcelona, 2016.

[16] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

[17] Karthik Yadati, Martha Larson, Cynthia CS Liem, and Alan Hanjalic. Detecting drops in electronic dance music: Content based approaches to a socially significant music event. In *ISMIR*, pages 143–148, 2014.

AUTOMATIC STYLISTIC COMPOSITION OF BACH CHORALES WITH DEEP LSTM

Feynman Liang
Department of Engineering
University of Cambridge
fl350@cam.ac.uk

Mark Gotham
Faculty of Music
University of Cambridge
mrhg2@cam.ac.uk

Matthew Johnson
Microsoft

Jamie Shotton
Microsoft

ABSTRACT

This paper presents "BachBot": an end-to-end automatic composition system for composing and completing music in the style of Bach's chorales using a deep long short-term memory (LSTM) generative model. We propose a new sequential encoding scheme for polyphonic music and a model for both composition and harmonization which can be efficiently sampled without expensive Markov Chain Monte Carlo (MCMC). Analysis of the trained model provides evidence of neurons specializing without prior knowledge or explicit supervision to detect common music-theoretic concepts such as tonics, chords, and cadences. To assess BachBot's success, we conducted one of the largest musical discrimination tests on 2336 participants. Among the results, the proportion of responses correctly differentiating BachBot from Bach was only 1% better than random guessing.

1. INTRODUCTION

Recent advances have enabled computational modeling to provide novel insights into a range of musical phenomena. One use case is *automatic stylistic composition*: the algorithmic generation of music in a style similar to a particular composer or repertoire. This study explores that goal, restricting its attention to *generative probabilistic sequence models* which are *learned from data*. This model is desirable because it can be applied to a variety of tasks, including: harmonizing a melody (by conditioning the model on the melody) and automatic composition (by sampling a sequence from the model).

The aim is to build a system capable of generating music in the style of Bach chorales such that *an average listener cannot distinguish it from original Bach.* While the method we develop is capable of modeling any multi-part music, we limit the scope of this work to Bach's chorales because: they provide a relatively large corpus, by a single composer, are well understood by music theorists, and are routinely used in the teaching of music theory.

1.1 Related Work

Two well-known difficulties in automatic composition are 1) learning the long-term dependencies required for plausible phrasing structure and motif distribution [31], and 2) evaluating the model's performance rigorously [34]. Addressing the first difficulty, more recent work has reported improvements in learning long-term dependencies by using LSTM [14, 13, 18]. Eck and Schmidhuber [14] used LSTM to model blues music and found that LSTM can indeed learn long-term aspects of musical structure such as repeated motifs without explicit modelling.

Evaluating model performance has proven to be more problematic. In recent work, researchers have begun conducting larger-scale human evaluations. Quick [35] evaluated her rule-based system's outputs on 237 human participants from Amazon's MTurk. Perhaps most relevant to the present study is Collins et al. [6]: a Markov chain expert system for automatic composition. The authors evaluated on 25 participants with a mean of 8.56 years of formal music training and found that only 20% of participants (5 out of 25) performed significantly better than chance. While these prior results are strong, both of these systems relied upon a large amount of expert domain knowledge encoded into the models. In contrast, BachBot leverages minimal prior knowledge and is evaluated on a significantly larger participant pool.

Bach chorales have been a popular corpus for previous work on automatic composition. Early deterministic systems included rule-based symbolic methods [7, 8, 12, 36], grammatical inference [9], and constraint logic programming [39]. Probabilistic models learned from data include the effective Boltzmann machine [3] as well as various connectionist models [37, 38, 24, 31, 15, 27].

Allan and Williams [1] used hidden Markov models to generate Bach chorale harmonizations and is one of the first studies to evaluate model performance quantitatively using cross-entropy on held-out data. They introduce the *JSB Chorales* dataset which has since become a standard benchmark routinely used to evaluate the performance of generative models on polyphonic music modelling [4, 33, 2, 21, 41]. However, *JSB Chorales* quantizes time to eighth notes, distorting 2816 notes (2.85% of the corpus). In contrast, BachBot eliminates this problem with 2× the time resolution (distorting no notes). Unfortunately, the higher resolution time quantization of Bach-

Bot's data as well as BachBot's sequential encoding format make direct comparison of cross-entropies against studies using this dataset difficult. On this dataset, the current state-of-the-art (as measured by cross-entropy validation loss) by Goel and Vohra [20] uses a deep belief network (DBN) which uses a LSTM to propagate temporal dynamics. While BachBot also utilizes a LSTM for capturing long range dependencies, BachBot uses a softmax distribution rather than a DBN to parameterize the probability distribution and hence does not require Monte Carlo sampling at each time step of training and inference.

A recent approach developed concurrent to BachBot was by Hadjeres and Pachet [23]. Their approach also uses an encoding which accounts for note articulations and fermatas and is similarly capable of harmonization under arbitrary constraints (e.g. a given Alto and Tenor part). However, their model utilizes LSTMs to summarize both past and future context within ± 16 time steps, limiting context to a temporally local region and inhibiting the learning of long-term structures such as motifs. Since future context is not always available, to generate samples the authors first randomly initialize a predetermined number of time steps followed by multiple iterations of MCMC. In contrast, BachBot's ancestral sampling method requires only a single forward pass and does not require the number of timestamps in the sample to be known in advance. The authors also evaluate their model using an online discrimination test, but on a smaller participant pool of 1272.

2. THE BACHBOT SYSTEM

2.1 Corpus Construction and Preprocessing

We took the full set of Bach chorales in MusicXML format as provided by Cuthbert and Ariza [10]. Following prior work [31, 14, 16, 17] preprocessing transposed all scores to C-major / A-minor and quantized time into sixteenth notes. Time quantization at this resolution does not distort any notes in the corpus.

2.2 Sequential Encoding of Polyphonic Music Scores

We encode the scores into sequences of tokens amenable for sequential processing by recurrent neural networks (RNNs). We limit the symbolic representation to pitch and rhythm. This is consistent with previous work [4, 33] and the practice of music theoretic pedagogy. Unlike some prior work [15, 14, 1], we avoid explicitly encoding music-theoretic concepts such as motifs, phrases, and chords / inversions, instead tasking the model to learn musically meaningful features with minimal prior knowledge (see section 3.4).

Our encoding represents polyphonic scores with sixteenth-note *frames*, encoding duration implicitly by the number of frames processed. Such an encoding requires the network to leverage memory to account for longer durations notes, a counting and timing task which LSTM is known to be capable of [19]. Consecutive frames are separated by a unique delimiter (`|||` in fig. 1).

Within each frame, we represent individual notes rather than entire chords, reducing the vocabulary size from $O(128^4)$ down to $O(128)$. Prior work modeling characters versus words in language modeling tasks suggests that this has negligible impact [22]. Each frame consists of four (Soprano, Alto, Tenor, and Bass) $\langle \texttt{Pitch}, \texttt{Tie} \rangle$ tuples where $\texttt{Pitch} \in \{0, 1, \cdots, 127\}$ represents the MIDI pitch of a note and $\texttt{Tie} \in \{\texttt{True}, \texttt{False}\}$ distinguishes whether a note is tied with a note at the same pitch from the previous frame or is articulated at the current timestep. We *order notes within a frame in descending MIDI pitch and neglects crossing voices*; potential consequences of doing so are discussed in section 3.2.

For each score, a unique `START` symbol and `END` symbol are added. This enables initialization of the trained model prior to ancestral sampling of a token sequence by providing a `START` token and also allows us to determine when a sampled composition ends. In addition, our encoding also includes fermatas (represented by `(.)`), which Bach used to denote ends of phrases. Significantly, we found that adding this additional notation to the input resulted in more realistic phrase lengths in generated output.

2.3 Model Architecture, Training, and Sampling

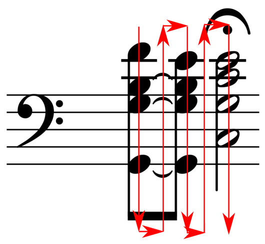
We use a RNN with LSTM memory cells and the following hyperparameters:

1. `num_layers` – the number of memory cell layers
2. `rnn_size` – the number of hidden units per memory cell (*i.e.* hidden state dimension)
3. `wordvec` – dimension of vector embeddings
4. `seq_length` – number of frames before truncating back-propagation through time (BPTT) gradient
5. `dropout` – the dropout probability

Our model first embeds the inputs $\boldsymbol{x}_t$ into a `wordvec`-dimensional vector-space, compressing the dimensionality down from $|V| \approx 140$ to `wordvec` dimensions. Next, `num_layers` layers of memory cells followed by batch normalization [28] and dropout [26] with dropout probability `dropout` are stacked. The outputs $\boldsymbol{y}_t^{(\texttt{num\_layers})}$ are followed by a fully-connected layer mapping to $|V| = 108$ units, which are passed through a softmax to yield a predictive distribution $P(\boldsymbol{x}_{t+1}|\boldsymbol{h}_{t-1}, \boldsymbol{x}_t)$: the probability distribution over the next token $\boldsymbol{x}_{t+1}$ given the current token $\boldsymbol{x}_t$ and the previous RNN memory cell state $\boldsymbol{h}_{t-1}$.

Models were trained using the Adam optimizer [29] with a minibatch size of 50 and an initial learning rate of 2×10^{-3} decayed by 0.5 every 5 epochs. The back-propagation through time gradients were clipped at ± 5.0 [32] and truncated after `seq_length` frames.

We minimize cross-entropy loss between the predicted distributions $P(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$ and the actual target distribution $\delta_{\boldsymbol{x}_{t+1}}$. During training, the correct token $\boldsymbol{x}_{t+1}$ is treated as the model output even if the most likely prediction $\operatorname{argmax} P(\boldsymbol{x}_{t+1}|\boldsymbol{h}_t, \boldsymbol{x}_t)$ differs. Williams and Zipser



(a) Three musical chords in traditional music notation. The red arrows indicate the order in which notes are sequentially encoded.

```
START           (59, True)      (55, False)
(65, False)     (55, True)      (48, False)
(59, False)     (43, True)      |||
(55, False)     |||             END
(43, False)     (.)
|||             (64, False)
(64, False)     (60, False)
```

(b) A corresponding sequential encoding of the three chords in an eighth-note time-quantization (for illustration, broken over three columns). Each line within a column corresponds to an individual token in the encoded sequence. `|||` delimit frames and `(.)` indicate a fermata is present within the corresponding frame.

Figure 1: Example encoding of three musical chords ending with a fermata ("pause") chord.

[40] refers to this as *teacher forcing*, which is performed to aid convergence because the model's predictions may not be reliable early in training. During inference, we perform ancestral sampling and reuse the actual token $\hat{\boldsymbol{x}}_t$ sampled from $P(\boldsymbol{x}_t|\boldsymbol{h}_{t-1}, \boldsymbol{x}_{t-1})$ to compute $P(\boldsymbol{x}_{t+1}|\boldsymbol{h}_t, \boldsymbol{x}_t)$ for sampling $\hat{\boldsymbol{x}}_{t+1}$. Unlike MCMC, which requires running multiple iterations to obtain a single sample, ancestral sampling requires only a single forward pass.

2.4 Harmonization with Greedy 1-best Search

Chorale harmonization involves providing accompaniment parts to an existing melody. This is a musical task with ecological validity undertaken by many composers including Bach himself. Many of Bach's chorales are harmonizations by Bach of pre-existing melodies (not by Bach) and certain melodies (by Bach or otherwise) form the basis of multiple chorales with different harmonizations.

We extend this harmonization task to the completion of chorales for a wider number and type of given parts. Let $\boldsymbol{x}_{(1:T)}$ be a sequence of tokens representing an encoded musical score, $\alpha \subset \{1, 2, \dots, T\}$ a multi-index, and suppose $\widehat{\boldsymbol{x}}_\alpha$ correspond to some fixed token values to be harmonized (e.g. a provided Soprano line).

We are interested in solving the following optimization:

$$\boldsymbol{x}^*_{(1:T)} = \underset{\boldsymbol{x}_{(1:T)}}{\operatorname{argmax}}\, P(\boldsymbol{x}_{(1:T)}|\boldsymbol{x}_\alpha = \widehat{\boldsymbol{x}}_\alpha) \tag{1}$$

First, any proposed solution $\tilde{\boldsymbol{x}}_{1:T}$ must satisfy $\tilde{\boldsymbol{x}}_\alpha = \widehat{\boldsymbol{x}}_\alpha$, so the decision variables are $\tilde{\boldsymbol{x}}_{(1:T)\setminus\alpha}$. Hinton and Sejnowski [25] refer to this constraint as "clamping" the generative model. We propose a simple greedy strategy for choosing $\tilde{\boldsymbol{x}}_{(1:T)\setminus\alpha}$:

$$\tilde{\boldsymbol{x}}_t = \begin{cases} \widehat{\boldsymbol{x}}_t & \text{if } t \in \alpha \\ \operatorname{argmax}_{\boldsymbol{x}_t} P(\boldsymbol{x}_t|\tilde{\boldsymbol{x}}_{1:t-1}) & \text{otherwise} \end{cases} \tag{2}$$

where the tilde on the previous tokens $\tilde{\boldsymbol{x}}_{1:t-1}$ indicate that they are equal to the actual previous argmax choices. This corresponds to a greedy 1-best search at each time t without any accounting of future constraints (e.g. $\boldsymbol{x}_\tau$ if $\tau > t$ and $\tau \in \alpha$). This is sub-optimal, and we leave more sophisticated search strategies such as beam search [30] for future work.

3. EXPERIMENTS

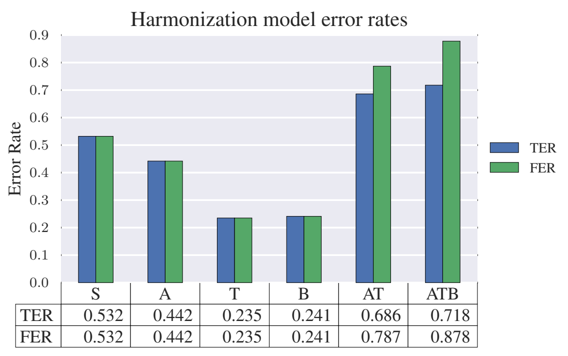
3.1 Sequence Modelling

With the BachBot model, we performed a grid search through the parameter grid in table 1 and found `num_layers` $= 3$, `rnn_size` $= 256$, `wordvec` $= 32$, `seq_length` $= 128$ `dropout` $= 0.3$ achieves the lowest cross-entropy loss of 0.477 bits on a 10% held-out validation corpus.

Parameter	Values Searched
`num_layers`	$\{1, 2, 3, 4\}$
`rnn_size`	$\{128, 256, 384, 512\}$
`wordvec`	$\{16, 32, 64\}$
`seq_length`	$\{64, 128, 256\}$
`dropout`	$\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$

Table 1: The grid of hyperparameters searched over while optimizing RNN structure

3.2 Harmonization



	S	A	T	B	AT	ATB
TER	0.532	0.442	0.235	0.241	0.686	0.718
FER	0.532	0.442	0.235	0.241	0.787	0.878

Figure 2: Token error rates (TER) and frame error rates (FER) for various harmonization tasks

For the parts to harmonize (i.e. $\boldsymbol{x}_{(1:T)\setminus\alpha}$), we considered the following test cases:

1. One part: Soprano (S), Alto (A), Tenor (T), or Bass (B).

2. The inner parts (AT). Completion of the inner parts corresponds to a musically-valid exercise common in Baroque composition (including some Bach chorales) where only the outer voices are specified (with or without figured bass to indicate the chord types).

3. All parts except Soprano (ATB): the most common form of *harmonization* exercise.

It is widely accepted that these tasks successively increase in terms of difficulty [11].

We deleted the different subsets of parts from a validation corpus and used eq. (2) to fill in the missing parts. Our model's error rates for predicting individual tokens (token error rate, TER, % of errors in individual token predictions) as well as all tokens within frames (frame error rate, FER, % of errors in frame predictions where any token prediction errors within a frame counts as a frame error) are reported in fig. 2.

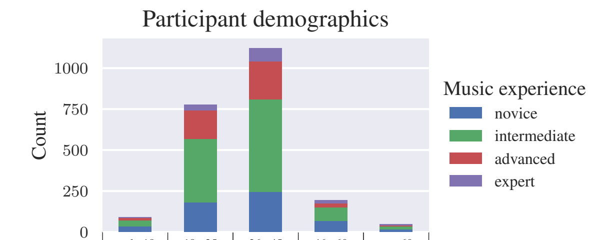
Surprisingly, error rates were higher for S/A than for T/B. One possible explanation for this result is our design decision in section 2.2 to order notes within a frame in SATB order. As a result, the model must predict the Soprano part for each frame without any knowledge of the other parts. When predicting the Bass part, however, it has already seen all of the other parts and can leverage this harmonic context. To assess this idea, we propose as future work an investigation of different part orderings in the encoding.

3.3 Musical Discrimination Test

To measure BachBot's success in this task, we developed a publicly accessible musical discrimination test at `bachbot.com`. Unlike prior studies which leverage paid services like Amazon MTurk for human feedback [35], we offered no such incentive and promoted the study only through social media.

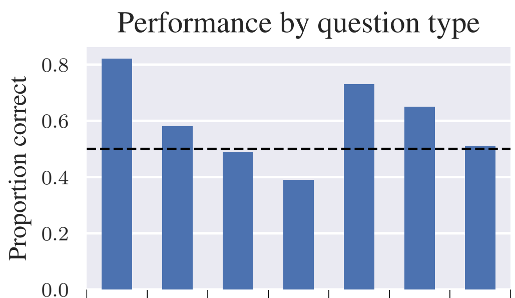
Participants were first surveyed for their age group and prior music experience (fig. 3a). Next, they are presented five discrimination tasks which presented two audio tracks (an original Bach composition and a synthetic composition by BachBot) and ask them to identify the Bach original. Each audio track contains an entire composition from start to end. The music score for the audio was not provided. Participants were granted an unlimited amount of time and allowed to replay each track an arbitrary number of times. Participants could only see the next question after submitting the current one and were not allowed to modify their responses after submitting.

The five questions comprised of three harmonizations (S/A/T/B, one AT, one ATB), and two original compositions. To construct the questions, harmonizations were paired along with the original Bach chorales the fixed parts were taken from. No such direct comparison is possible for the SATB case, so these synthetic compositions were paired with a randomly selected Bach chorale in a somewhat different comparative listening task. Harmonizations



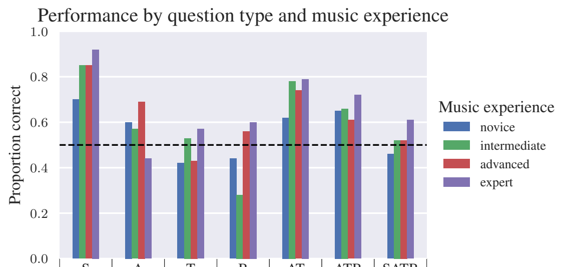
	under18	18to25	26to45	46to60	over60
novice	34	181	244	66	16
intermediate	36	387	565	85	18
advanced	17	176	233	23	5
expert	5	34	81	21	9

(a) Demographics of respondents; self-reported music experience defined as follows — *Novice*: casual listener, *Intermediate*: plays an instrument, *Advanced*: formally studied music composition, *Expert*: music teacher/researcher.



	S	A	T	B	AT	ATB	SATB
Proportion	0.82	0.58	0.49	0.39	0.73	0.65	0.51

(b) Proportion of responses correctly discriminating BachBot from Bach for different question types. The SATB column shows that BachBot's generated compositions can be differentiated from Bach only 1% better than random guessing.



	S	A	T	B	AT	ATB	SATB
novice	0.7	0.6	0.42	0.44	0.62	0.65	0.46
intermediate	0.85	0.57	0.53	0.28	0.78	0.66	0.52
advanced	0.85	0.69	0.43	0.56	0.74	0.61	0.52
expert	0.92	0.44	0.57	0.6	0.79	0.72	0.61

(c) Figure 3b segmented by self-reported music experience. As expected, more experienced listeners generally produced more correct responses, though not for the 'B' condition.

Figure 3: Results collected from a web-based musical discrimination test.

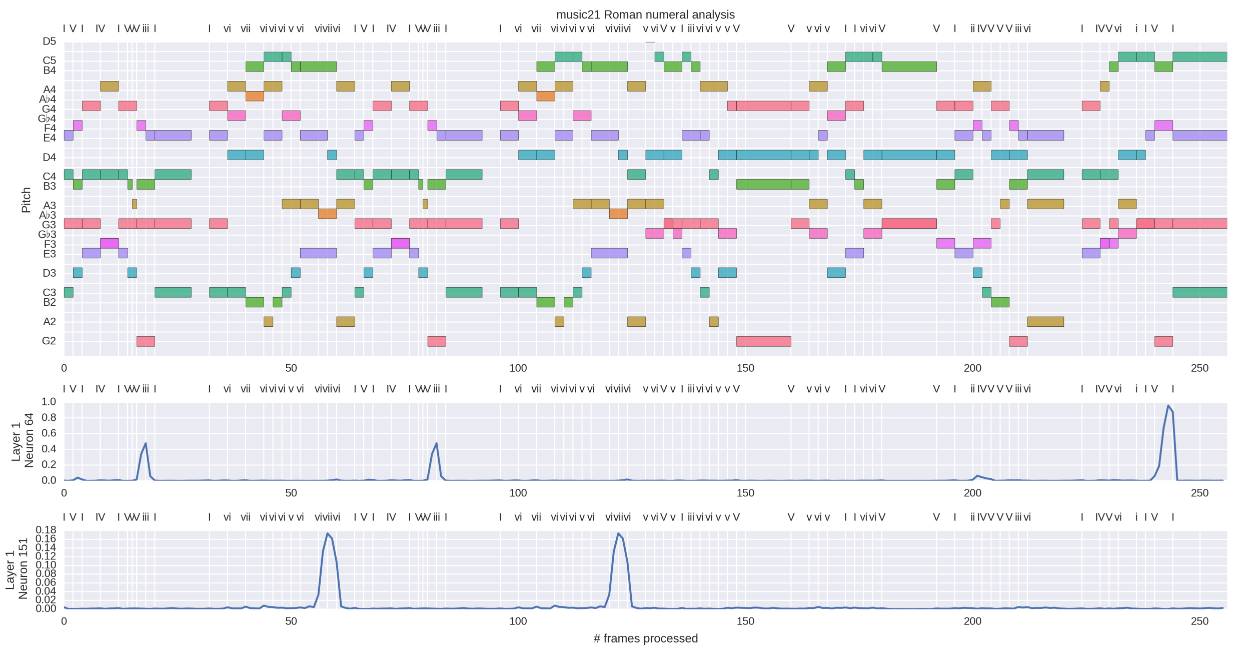


Figure 4: Activation profiles suggesting that neurons have specialized to become detectors of musically relevant features. *Layer 1, neuron 64*: strongly correlates with the use of dominant seventh chords in the main, tonic key (C major, originally D major). These are the main non-triadic harmony, are strongly key defining, and have a important function in the harmonic closure of phrases in this style. *Layer 1, neuron 151*: fires with the equivalent dominant seventh chord for the two cadences in the relative minor (a minor, originally b minor) that end phrases 2 and 4. These are the only two appearances in the chorale of the pitch G# which is foreign to C major, and strongly key defining in a minor.

were synthesized by extracting part(s) from a randomly selected Bach chorale and filling in the remaining parts of the composition using the method previously described in section 2.4. Original compositions (questions labelled SATB) were generated by providing a `START` symbol followed by ancestral sampling as previously described in section 2.3 until an `END` symbol is reached. The final audio provided in the questions were obtained by rendering the compositions using the Piano instrument from the Fluid R3 GM SoundFont.

We only considered the first response per IP address of participants who had played both choices in every question at least once and completed all five questions. This totaled 2336 participants at the time of writing, making our study one of the largest subjective listening evaluation of an automatic composition system to date.

Figure 3b shows the performance of BachBot on various question types. The SATB column shows that, for the novel synthetic compositions, participants on average successfully discriminated Bach from BachBot only 51%: *average human listeners could only perform* 1% *better than random guessing*. To assess statistical significance, we choose significance level $\alpha = 0.05$ and conducted a one-tailed binomial test (446 successes in 874 trials) to find that the probability of a discrimination rate higher than 51% has p-value $0.282 > \alpha$. Thus, we conclude that there does not exist sufficient evidence that the discrimination rate between Bach and BachBot is significantly different (at $\alpha = 0.05$) than the rate achieved by random guessing random guessing .

The weaker performance of BachBot's outputs on most harmonization questions (fig. 3b other than SATB) compared to automatic composition questions (SATB) is counterintuitive: one would expect the provided parts to aid the model in creating more Bach-like music. This result may be explained by the shortcomings of our greedy 1-best harmonization method (discussed above) and/or by the possible benefit of consistent origins, with all-Bach and all-BachBot being preferred over hybrid solutions.

Across the S/A/T/B and AT/ATB conditions, the results vary significantly. The ease of discrimination appears to correlate with the position in the texture from highest (S, easiest) to lowest (B, hardest). This may be due to the S part's importance in carrying the melody in chorale style, or (more likely) due once again to the BachBot's lower error rates for completing bass parts as compared with other parts (fig. 2), which in turn is probably due to the sequential encoding (fig. 1) of bass notes last within each frame, giving it a harmonic context to work with. Another possibility is that most listeners focus more on the top melody, neglecting the bass part and any potential deviations there. In any case, the relatively poor performance of expert listeners for the B-only condition (see fig. 3c) is noteworthy, and not explained by any aspect of the process.

3.4 Do Neurons Specialize to Music-Theoretic Concepts?

Research in convolutional networks has shown that neurons within computer vision models specialize to detect high-level visual features [42]. Similarly, convolutional networks trained on audio spectrograms have been shown to possess neurons which detect high-level aural features [5]. Following these results, one might expect the BachBot model to possess neurons which detect features within symbolic music which have music theoretic relevance.

To investigate this further, one could look at the activations over time of individual neurons within the LSTM memory cells to see if neuron activity correlates with recognized musical processes. An informal analysis suggests that while some neurons are ambiguous to interpretation, other neurons correlate significantly with recognized music-theoretic objects, particularly chords (see fig. 4). To our knowledge, *this is the first reported evidence for an LSTM optimized for automatic composition learning music-theoretic concepts without explicit prior information*. This invites a follow-up study testing the statistical significance of these observations.

4. DISCUSSION

The data generated by `bachbot.com` shows that subjects distinguished BachBot from Bach only 51% of the time, suggesting that BachBot successfully composes and completes music that cannot be distinguished from Bach significantly above the chance level. Additionally, BachBot's design involves no explicit encoding of musical parameters beyond the notation, so the results reflects its ability to acquire music knowledge independently from data.

As discussed, the higher time resolution of our custom encoding scheme enabled the model to learn about Bach's use of sixteenth notes, which is not possible for models trained on *JSB Chorales*. Unfortunately, this improved encoding means that we are unable to compare quantitative performance metrics such as log likelihood against other literature values reported for polyphonic modeling on the *JSB Chorales* [1] dataset.

Using this sequential encoding scheme, we train a deep LSTM sequential prediction model and discover that it learns music theoretic concepts without prior knowledge or explicit supervision. We then propose a method to utilize the sequential prediction model for harmonization tasks. We acknowledge that our method is not ideal and discuss better alternatives in future work. Our harmonization results reveal that this issue is significant and should be a priority for any follow-up work.

Finally, we leveraged our model to generate harmonizations as well as novel compositions and used the generated music in a web-based music discrimination test. Our results here confirm the success of our project.

While many opportunities for extension are highlighted, we conclude that our stated research aims have been reached. In other words, generating stylistically successful Bach chorales is now a more closed (as a result of BachBot) than open problem.

5. CONCLUSION

In this paper, we:

- introduce a sequential encoding scheme for music which achieves time-resolution $2\times$ that of the commonly used *JSB Chorales* [1] dataset.

- performed the largest (to the best of our knowledge at time of publication) musical discrimination test of an automatic composition system, which demonstrated that high quality data can be collected from voluntary internet surveys.

- demonstrate that a deep LSTM sequential prediction model trained on our encoding scheme is capable of composing music that can be distinguished only 1% better than random guessing, a statistically insignificant difference

- provide the first evidence that neurons in the LSTM model appear to model common music-theoretic concepts without prior knowledge or supervision.

In addition, we have open sourced the code for BachBot [1] as well as our music discrimination test framework [2]. The Magenta project of Google Brain has recently implemented the BachBot model for their polyphonic RNN model [3].

6. REFERENCES

[1] Moray Allan and Christopher KI Williams. allan2005. *Advances in Neural Information Processing Systems*, 17:25–32, 2005.

[2] Justin Bayer, Christian Osendorfer, Daniela Korhammer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*, 2013.

[3] Matthew I Bellgard and Chi-Ping Tsang. Harmonizing music the boltzmann way. *Connection Science*, 6 (2-3):281–297, 1994.

[4] Nicolas Boulanger-Lewandowski, Pascal Vincent, and Yoshua Bengio. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. *Proc. of the 29th International Conference on Machine Learning (ICML-12)*, (Cd):1159–1166, 2012.

[1] `https://github.com/feynmanliang/bachbot`

[2] `https://github.com/feynmanliang/subjective-evaluation-server` and `https://github.com/feynmanliang/subjective-evaluation-client`

[3] `https://github.com/tensorflow/magenta/tree/master/magenta/models/polyphony_rnn`

[5] Keunwoo Choi, George Fazekas, Mark Sandler, and Jeonghee Kim. Auralisation of deep convolutional neural networks: Listening to learned features. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, pages 26–30, 2015.

[6] Tom Collins, Robin Laney, Alistair Willis, and Paul H Garthwaite. Developing and evaluating computational models of musical style. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 30(01):16–43, 2016.

[7] David Cope. Experiments in music intelligence. In *Proc. of the International Computer Music Conference*, 1987.

[8] David Cope. Computer modeling of musical intelligence in emi. *Computer Music Journal*, 16(2):69–83, 1992.

[9] Pedro P Cruz-Alcázar and Enrique Vidal-Ruiz. Learning regular grammars to model musical style: Comparing different coding schemes. In *International Colloquium on Grammatical Inference*, pages 211–222. Springer, 1998.

[10] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. 2010.

[11] James Denny. *The Oxford school harmony course*, volume 1. Oxford University Press, 1960.

[12] Kemal Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3): 43–51, 1988.

[13] D. Eck and J. Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. *Neural Networks for Signal Processing - Proc. of the IEEE Workshop*, 2002-Janua: 747–756, 2002. ISSN 0780376161. doi: 10.1109/NNSP.2002.1030094.

[14] Douglas Eck and Jürgen Schmidhuber. A 1st Look at Music Composition using LSTM Recurrent Neural Networks. *Idsia*, 2002. URL `http://www.idsia.ch/{~}juergen/blues/IDSIA-07-02.pdf`.

[15] Johannes Feulner and Dominik Hörnel. Melonet: Neural networks that learn harmony-based melodic variations. In *Proc. of the International Computer Music Conference*, pages 121–121. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1994.

[16] Judy A Franklin. Recurrent neural networks and pitch representations for music tasks. In *FLAIRS Conference*, pages 33–37, 2004.

[17] Judy A Franklin. Jazz melody generation from recurrent network learning of several human melodies. In *FLAIRS Conference*, pages 57–62, 2005.

[18] Judy A Franklin. Recurrent neural networks for music computation. *INFORMS Journal on Computing*, 18(3):321–338, 2006.

[19] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[20] Kratarth Goel and Raunaq Vohra. Learning temporal dependencies in data using a dbn-blstm. *arXiv preprint arXiv:1412.6093*, 2014.

[21] Kratarth Goel, Raunaq Vohra, and JK Sahoo. Polyphonic music generation by modeling temporal dependencies using a rnn-dbn. In *International Conference on Artificial Neural Networks*, pages 217–224. Springer, 2014.

[22] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[23] Gaëtan Hadjeres and François Pachet. Deepbach: a steerable model for bach chorales generation. *arXiv preprint arXiv:1612.01010*, 2016.

[24] Hermann Hild, Johannes Feulner, and Wolfram Menzel. Harmonet: A neural net for harmonizing chorales in the style of js bach. In *NIPS*, pages 267–274, 1991.

[25] Geoffrey E Hinton and Terrence J Sejnowski. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:282–317, 1986.

[26] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[27] Dominik Hörnel. Melonet i: Neural nets for inventing baroque-style chorale variations. In *NIPS*, pages 887–893, 1997.

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[29] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Xunying Liu, Yongqiang Wang, Xie Chen, Mark JF Gales, and Philip C Woodland. Efficient lattice rescoring using recurrent neural network language models. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4908–4912. IEEE, 2014.

[31] Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 6(2-3):247–280, 1994.

[32] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proc. of The 30th International Conference on Machine Learning*, (2):1310–1318, 2012. ISSN 1045-9227. doi: 10.1109/72.279181. URL `http://jmlr.org/proceedings/papers/v28/pascanu13.pdf`.

[33] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[34] Marcus Pearce and Geraint Wiggins. Towards a framework for the evaluation of machine compositions. In *Proc. of the AISB'01 Symp. on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 22–32. Citeseer, 2001.

[35] Donya Quick. *Kulitta: A Framework for Automated Music Composition*. PhD thesis, YALE UNIVERSITY, 2014.

[36] Randall R Spangler, Rodney M Goodman, and Jim Hawkins. Bach in a box-real-time harmony. 1998.

[37] Peter Todd. A sequential network design for musical applications. In *Proc. of the 1988 connectionist models summer school*, pages 76–84, 1988.

[38] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4): 27–43, 1989.

[39] Chi Ping Tsang and Melanie Aitken. Harmonizing music as a discipline in contraint logic programming. In *Proc. of the International Computer Music Conference*, pages 61–61. INTERNATIONAL COMPUTER MUSIC ACCOCIATION, 1991.

[40] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[41] Wojciech Zaremba. An empirical exploration of recurrent network architectures. 2015.

[42] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

CLUSTERING EXPRESSIVE TIMING WITH REGRESSED POLYNOMIAL COEFFICIENTS DEMONSTRATED BY A MODEL SELECTION TEST

Shengchen Li
Beijing University of
Posts and Telecommunications
shengchen.li@bupt.edu.cn

Simon Dixon
Queen Mary University of London
s.e.dixon@qmul.ac.uk

Mark D. Plumbley
University of Surrey
m.plumbley@surrey.ac.uk

ABSTRACT

Though many past works have tried to cluster expressive timing within a phrase, there have been few attempts to cluster features of expressive timing with constant dimensions regardless of phrase lengths. For example, used as a way to represent expressive timing, tempo curves can be regressed by a polynomial function such that the number of regressed polynomial coefficients remains constant with a given order regardless of phrase lengths. In this paper, clustering the regressed polynomial coefficients is proposed for expressive timing analysis. A model selection test is presented to compare Gaussian Mixture Models (GMMs) fitting regressed polynomial coefficients and fitting expressive timing directly. As there are no expected results of clustering expressive timing, the proposed method is demonstrated by how well the expressive timing are approximated by the centroids of GMMs. The results show that GMMs fitting the regressed polynomial coefficients outperform GMMs fitting expressive timing directly. This conclusion suggests that it is possible to use regressed polynomial coefficients to represent expressive timing within a phrase and cluster expressive timing within phrases of different lengths.

1. INTRODUCTION

In performed classical piano music, small variations of the beat length serving music expression is known as *expressive timing*. Expressive timing can be represented by *tempo curves* that connects the value of tempo on each beat to form a curve. A common method [3, 5, 8, 11] of analysing expressive timing within a phrase in performed classical piano music is to cluster expressive timing. One of the possible unit used for clustering expressive timing is *phrase* [5] that contains a certain beats forming a sensible music structure. The length of phrase, or *phrase length* (defined as the number of beats contained in a phrase), is expected to be identical throughout a piece of music by most algorithms such as Li et al. [5]. Such strong restrictions make the large-scale applications of existing algorithms almost impossible because the phrase lengths are not constant in most pieces. This paper proposes a way to cluster expressive timing regardless of phrase length.

In past research [14], polynomial functions, especially parabolic functions, are used to regress tempo curves. Regressing a tempo curve, the coefficients of the resulting polynomial function is called *regressed polynomial coefficients* for a tempo curve. Given an order of polynomial function to be regressed to, each tempo curve can be represented by a fixed number of regressed coefficients. In this paper, we propose to cluster regressed polynomial coefficients instead of clustering expressive timing directly in order to enable the clustering of expressive timing without a pre-defined unit possible. A model selection test is shown in this paper demonstrating the Gaussian Mixture Models (GMMs) fitting regressed polynomial coefficients outperform the GMMs fitting expressive timing directly.

For simplicity, the GMMs fitting the expressive timing are represented by GMM_o, whereas the GMMs fitting the regressed polynomial coefficients are represented as GMM_r. There are multiple ways to compare two GMMs. Because the two types of GMMs fitting two different sets of data in this paper, the traditional model selection criteria (such as Bayesian Information Criterion [1, Ch. 3] used by Li et al. [5]) based on model likelihood cannot be used. Although comparing the clustering results with a ground truth is a more general way to evaluate model performance, the clustering of expressive timing has no consensus or well-recognised "ground truth" by the musicologists. The performance of GMMs is evaluated by the approximation of each tempo curve by their corresponding centroids as this principle is a general evaluation for clustering algorithms.

To make the clustering of expressive timing and the regressed polynomial coefficients comparable, the pieces we selected in this paper still have constant phrase lengths. However, the expressive timing in various phrases can be regressed to the polynomial function of a single order regardless of phrase lengths. The three pieces of music are two pieces of Chopin's Mazurkas (Op. 24, No. 2 and Op. 30, No. 2) used in the previous works [10, 11] and Berekrev's *Islamey* dataset, which Li et al [5] used. Although the music analysed is classical music, the proposed algorithm for clustering expressive timing may be potentially used for other forms of music such as jazz music.

This paper is organised as follows: relevant literatures are reviewed first, then we describe how the standardised tempo curves and the regressed polynomial coefficients are clustered. Next, we will show how the performance of models are represented and the results are presented. A discussion comparing the differences of the GMMs precedes the conclusion of the paper.

2. BACKGROUNDS

Clustering is a widely used methodology for analysing expressive timing. As demonstrated by Li et al. [5], the standardised tempo curves within a phrase can be clustered. Repp [8] used Principle Component Analysis (PCA) to analyse expressive timing and found a certain number of common patterns. Spiro et al. [11] used self-organising maps to cluster expressive timing and expressive dynamics patterns within a bar and asserted that expressive timing and dynamics are affected by music structure. All these works requested a pre-selected unit of analysis with an identical length, such as bars, phrases or the entire piece of music. Such requirements, on the other hand, limit the usability of the methods of analysis because the choice of a unified unit for analysis is hard to find. Clustering regressed polynomial coefficients instead of expressive timing directly relaxes the restriction of a constant phrase length in the testing pieces; thus, more pieces of music can be analysed using different methodologies of clustering.

Using second-order polynomial function, or parabolic function, to regress expressive timing tempo curves representing expressive timing is a traditional way to model expressive timing [14]. This method was widely used in a range of past works [8, 9, 12, 15, 16]. Repp [8, 9] used PCA to analyse expressive dynamics and timing in certain numbers of performances of a Chopin's étude. Repp asserted that the parabolic curves are particularly good at modelling the expressive timing within a longer phrase unit [8] and that parabolic curves are useful for regressing the expressive dynamics [9]. Tobudic and Widmer [13] used multi-level parabolic curves to learn how a concert pianist varied both dynamics and tempo when playing several Mozart pieces. The learned methods were then used to automatically render performances of other pieces with success. Timmers [12] suggested that using parabolic curves to regress expressive parameters in performances is useful in vocal performances. Despite the wide usage of parabolic curves for modelling, it is rare to cluster expressive timing with the regressed parabolic coefficients or regressed polynomial coefficients. This paper intends to use a model selection test to demonstrate that regressed polynomial coefficients are a valid representation of expressive timing for clustering.

GMMs are used to fit the distribution of expressive timing within a phrase and regressed polynomial coefficients. The resulting GMM$_o$ and GMM$_r$ are compared in the proposed model selection test. A model selection test is a common method in machine learning research to test the fitness of data with a mathematical model [1, Ch. 1]. Li et al. [5] used this method to analyse expressive timing. Model selection tests were used to demonstrate expressive timing can be modelled by a clustered model [5] and to determine the factors that affect the selection of clusters of expressive timing [6]. Because GMM$_o$ and GMM$_r$ model two different datasets, the approximation of expressive timing by their corresponding centroids of GMM$_o$ and GMM$_o$ is used for evaluation in this paper.

We adapt the dataset used by Li et al. [5] in which each piece has a constant length phrase to make GMM$_o$ and GMM$_r$ comparable. The three testing pieces of music are Chopin's Mazurkas (Op. 24, No. 2 and Op. 30, No. 2) and *Islamey*, whose lengths of phrases are twelve beats, twenty-four beats and eight beats throughout the entire piece, respectively. For each testing piece, there are sixty-four, thirty-four and twenty-five performances.

In each performance, the timing of each beat is recorded as $\{t_1, t_2, \ldots\}$ and the tempo value on each beat τ_i can be calculated as the reciprocal of inter-beat interval, namely $\tau_i = \frac{1}{t_{i+1} - t_i}$. The tempo value is then smoothed by the method of moving window average with a window size of 3 (i.e. $\bar{\tau}_i = \frac{\tau_{i-1} + \tau_i + \tau_{i+1}}{3}$) to approximate human perception of tempo [2]. The expressive timing within a phrase can then be represented as a vector of tempi or tempo curve: $\mathcal{T} = \{\bar{\tau}_1, \bar{\tau}_2, \ldots, \bar{\tau}_n\}$, where n represents the total number of beats. The resulting standardised tempo curves $\mathbf{T}$ are obtained by setting the mean of each tempo curve to 1, e.g. $\mathbf{T} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_n\}$, where $\hat{\tau}_i = \frac{n\bar{\tau}_i}{\sum_{j=1}^{n} \bar{\tau}_j}$. After the standardisation process, in each data set there are m samples of n-dimensional data to be clustered, where m represents the number of phrases in the testing piece of music. These samples are the raw data for clustering and regression.

3. MODEL EVALUATION

In this paper, a method to cluster expressive timing regardless of the length of phrase is proposed. As there are no musicological ground truth available, the candidate models are evaluated by a traditional way to assess unsupervised machine learning algorithms: how well the original data can be approximated by the centroids of clusters.

Before discussing how GMM$_o$ and GMM$_r$ are compared in details, we will firstly brief how the GMMs are trained to fit data with n dimensions. A traditional way to train a GMM distribution is to use the Expectation Maximisation (EM) algorithm [7, Ch. 11] which attempts to raise the model likelihood of the training data by adjusting the parameters in GMM. Particularly in this paper, all the GMMs are trained for ten times with random initialisation and the best GMM is selected as the resulting GMM.

Next we will show how GMM$_o$ and GMM$_r$ are compared. As these two types of GMMs fit different data, the traditional measurements based on model likelihood such as BIC (Bayesian Information Criterion) are not valid. As a result, we evaluate how well the expressive timing within a phrase is approximated by the centroids of the resulting GMM$_o$ and GMM$_r$. We will discuss how the approximation is measured in this section.

3.1 Evaluation of the Clustered Standardised Tempo Curves

Suppose the expressive timing in the ith phrase can be represented as $\mathbf{T_i} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_n\}$. The distribution of $\mathbf{T_i}$ can be fitted to an A-component GMM (GMM$_o$) as [5]

$$p(\mathbf{T_i}) = \sum_{k=1}^{A} \pi_k \mathcal{N}(\mathbf{T_i}|\vec{\mu}_k, \mathbf{\Sigma}_k). \tag{1}$$

The centroids of the resulting GMM$_o$ can be represented as $\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_A$. If $\vec{\mu_i} = (\mu_1, \mu_2, \ldots, \mu_n)$ is used to represent the centroids of the cluster that tempo curve of the ith phrase ($\mathbf{T_i} = \{\bar{\tau}_1, \bar{\tau}_2, \ldots, \bar{\tau}_n\}$) belongs to (where $\vec{\mu_i} \in \{\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_A\}$), the correlation coefficient (ρ) and Euclidean distance ($\mathcal{D}$) between the corresponding centroids μ_i and the expressive timing within a phrase $\mathbf{T_i}$ are given by

$$\rho(\mathbf{T_i}, \vec{\mu_i}) = \frac{\sum_{j=1}^{n}(\bar{\tau}_j - \overline{\mathbf{T_i}})(\mu_j - \overline{\vec{\mu_i}})}{\sqrt{\sum_{j=1}^{n}(\bar{\tau}_j - \overline{\mathbf{T_i}})^2}\sqrt{\sum_{j=1}^{n}(\mu_j - \overline{\vec{\mu_i}})^2}} \tag{2}$$

$$\mathcal{D}(\mathbf{T_i}, \vec{\mu_i}) = \sqrt{\sum_{j=1}^{n}(\bar{\tau}_j - \mu_j)^2} \tag{3}$$

where $\overline{\mathbf{T_i}} = \dfrac{1}{n}\sum_{k=1}^{n}\hat{\tau}_k$ and $\overline{\vec{\mu_i}} = \dfrac{1}{n}\sum_{k=1}^{n}\mu_k$.

3.2 Evaluation of the Regressed Polynomial Coefficients

With the least square algorithm, the standardised tempo curves can be regressed to a oth order polynomial function $f^o(x) = \sum_{i=0}^{o} b_i x^i$. Thus the standardised tempo curve representing expressive timing in phrase i ($\mathbf{T_i} = \{\bar{\tau}_1, \bar{\tau}_2, \ldots, \bar{\tau}_n\}$) can be represented by a vector of regressed polynomial coefficients $\vec{\mathbf{B}}_i = (b_0, b_1, b_2, \ldots, b_o)$. A GMM (GMM$_g$) fitting the oth order of polynomial coefficients is represented by GMM$_g^o$. For clarity, GMM$_g$ represents the GMMs fitting the regressed parabolic coefficient of any orders.

To prevent overfitting (e.g. the function used for regression is too complex to generalise the distribution of data), the order of polynomial function o should be smaller than the length of phrase n ($o < n$). The GMM$_g^o$ can be trained to fit the distribution of $\vec{\mathbf{B}}_i$ such that

$$p(\vec{\mathbf{B}}_i) = \sum_{k=1}^{A} \pi_k \mathcal{N}(\vec{\mathbf{B}}_i|\vec{m}_k, \mathbf{\Sigma}_k). \tag{4}$$

If the expressive timing of phrase i ($\mathbf{T_i}$) belongs to cluster k whose centroid can be represented as $\vec{m}_k = (b_{m0}, b_{m1}, \ldots, b_{mo})$, the regressed polynomial curve of the expressive timing within phrase i ($\mathbf{T_i}$) can be represented as $f^o(x|\vec{m}_i) = \sum_{j=0}^{o} b_{mj}x^j = (x_1, x_2, \ldots, x_n)$. Thus the correlation coefficient (ρ) and Euclidean distance ($\mathcal{D}$) between the regressed polynomial curve $f^o(x|\vec{m}_i)$ and the expressive timing $\mathbf{T_i}$ are given by

$$\rho(\mathbf{T_i}, f^o(x|\vec{m}_i)) = \frac{\sum_{j=1}^{n}(\bar{\tau}_j - \overline{\mathbf{T_i}})(f_j - \overline{f^o(x|\vec{m}_i)})}{\sqrt{\sum_{j=1}^{n}(\bar{\tau}_j - \overline{\mathbf{T_i}})^2}\sqrt{\sum_{j=1}^{n}(f_j - \overline{f^o(x|\vec{m}_i)})^2}} \tag{5}$$

$$\mathcal{D}(\mathbf{T_i}, f^o(x|\vec{m}_i)) = \sqrt{\sum_{j=1}^{n}(\bar{\tau}_j - x_j)^2} \tag{6}$$

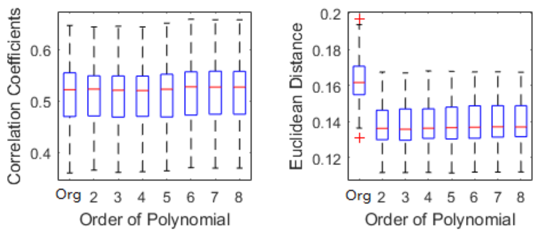
where $\overline{\mathbf{T_i}} = \dfrac{1}{n}\sum_{k=1}^{n}\hat{\tau}_k$ and $\overline{f^o(x|\vec{m}_i)} = \dfrac{1}{n}\sum_{k=1}^{n}x_k$.

4. RESULTS

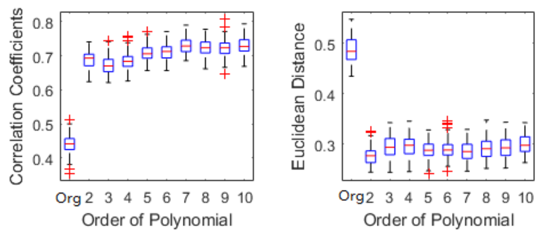
In this section, we will compare how the centroids of GMM$_o$ and GMM$_g$ approximate expressive timing within a phrase by showing the correlation coefficients and Euclidean distance discussed in Section 3. To train a GMM with the dataset selected, an important parameter should be decided: the intended number of clusters. Following the detailed discussion by Li et al. [5], we train GMMs with two Gaussian components for *Islamey*, eight Gaussian components for Chopin Mazurka Op.24/2 and four Gaussian components for Chopin Mazurka Op.30/2. Moreover, the order of polynomial function for regression is chosen between the second order and the tenth order for both Chopin's Mazurkas, whereas for *Islamey* whose phrase length is 8 beats the chosen order of polynomial function is between second order and the eighth order to prevent overfitting.

Compared with the complexity of the proposed GMM$_o$ and GMM$_g$, the data we have is fairly limited. To prevent overfitting, cross validation is used in this experiment. Rather than using the entire dataset to train the GMM$_o$ and GMM$_g$, only four-fifths of the performances form a training dataset, and the remaining performances form a testing dataset. Specifically for the candidate pieces, there are 5, 13, 7 performances used for testing and the numbers of performances for training are 20, 51, 27 for the candidate pieces *Islamey*, Chopin's Mazurka Op.24/2 and Chopin's Mazurka Op.30/2 respectively. To even out the possible bias caused by the randomness of the formation of the testing and training sets, cross validation tests are repeated for 100 times with the performances in the testing and training sets randomly selected. The performance of candidate models are evaluated by the average performance in the 100 cross validation tests.

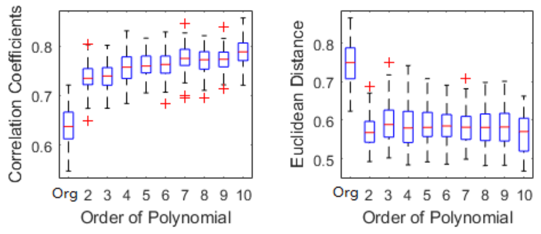
With the EM algorithm, a GMM$_o$ and a GMM$_g$ are trained with each training dataset engaged. The resulting GMM$_o$ and GMM$_g$ are used to cluster the testing dataset. The centroids of the resulting clusters are used to calculate $\rho(\mathbf{T_i}, \vec{\mu_i})$, $\mathcal{D}(\mathbf{T_i}, \vec{\mu_i})$, $\rho(\mathbf{T_i}, f^o(x|\vec{m}_i))$ and $\mathcal{D}(\mathbf{T_i}, f^o(x|\vec{m}_i))$ according to equations (2), (3), (5) and (6) where $\mathbf{T_i}$ is in the testing dataset. To remove the possible bias caused by the randomness of performance selection, the experiment is repeated 100 times. The resulting $\rho(\mathbf{T_i}, \vec{\mu_i})$, $\rho(\mathbf{T_i}, f^o(x|\vec{m}_i))$, $\mathcal{D}(\mathbf{T_i}, \vec{\mu_i})$ and $\mathcal{D}(\mathbf{T_i}, f^o(x|\vec{m}_i))$ are compared pairwisely.



(a) *Islamey*



(b) Mazurka Op.24/2



(c) Mazurka Op.30/2

Figure 1. Box plots of the resulting correlation coefficients and Euclidean distance between the standardised tempo curves and their corresponding cluster centroids. The box shows 25th and 75th percentiles. The line in the box shows the mean. Outliers are shown by a '+' sign. A higher correlation coefficients and a smaller Euclidean distance indicates a better approximation of expressive timing by corresponding centroids. The label 'Org' represents the results of clustering expressive timing directly.

In Figure 1, box plots of the resulting $\rho(\mathbf{T_i}, \vec{\mu}_i)$, $\mathcal{D}(\mathbf{T_i}, \vec{\mu}_i)$, $\rho(\mathbf{T_i}, f^o(x|\vec{m}_i))$ and $\mathcal{D}(\mathbf{T_i}, f^o(x|\vec{m}_i))$ in the 100 cross-validation tests for each testing piece are shown. In the diagram, the label 'Org' represents the performance of the centroids in GMM$_o$ (namely $\rho(\mathbf{T_i}, \vec{\mu}_i)$ and $\mathcal{D}(\mathbf{T_i}, \vec{\mu}_i)$,). The numbered labels represent the value of o in $\rho(\mathbf{T_i}, f^o(x|\vec{m}_i))$ and $\mathcal{D}(\mathbf{T_i}, f^o(x|\vec{m}_i))$. In each boxing plot, the box indicates the 25th and 75th percentiles and the line in the box shows the mean. The '+' signs show the outliers. A higher correlation coefficient and a smaller Euclidean distance means better approximation of the expressive timing by the corresponding centroids in GMM$_o$ and GMM$_g$.

In the resulting diagram, GMM$_g^o$ outperforms GMM$_g$ regardless of the value of o. As seen in Figure 1(b) and Figure 1(c), GMM$_g^o$ outperforms GMM$_r$ according to both the correlation coefficients and Euclidean distance. In Figure 1(a), although the correlation coefficients does not show that GMM$_g^o$ is better than GMM$_r$, the Euclidean distance shows that GMM$_g^o$ outperforms GMM$_r$. This result confirms that using polynomial functions to regress expressive timing within a phrase can help to improve the performance of clustering expressive timing.

Next, we discuss about which value of o makes the best performed GMM$_g^o$. With an one-way ANOVA test [7, Ch.8], we find that a higher value of o does not always introduce a better performance of GMM$_g^o$. To show the significance of the difference between the means and correlation coefficients of GMM$_g^o$ and GMM$_r$, we perform a Tukey's Honest Significant Difference (HSD) test.

With a preference of a simpler model, the results of Tukey's HSD show the following facts. If two GMM$_g^o$ have different values of o but no differences of performance, the GMM$_g^o$ with lower o values will be preferred due to lower complexity of GMM$_g^o$. For *Islamey*, the performance of GMM$_g^2$ to GMM$_g^8$ does not make significant differences thus GMM$_g^2$ is preferred. As a result, the second order of polynomial function is the most suitable method regressing expressive timing in *Islamey*. For Chopin Mazurka Op.24/2, GMM$_g^7$ to GMM$_g^{10}$ make no significant differences according to correlation coefficients whereas according to Euclidean distance, GMM$_g^{10}$ is worse than GMM$_g^7$ to GMM$_g^9$. So in general GMM$_g^7$ is the best model amongst the candidate models and the seventh order of polynomial function is the best function to regress expressive timing within a phrase for Mazurka Op.24/2. For Chopin Mazurka Op.30/2, the best performed models are GMM$_g^7$ to GMM$_g^{10}$ according to Euclidean distance whereas GMM$_g^{10}$ is marginally better than other models according to correlation coefficients. As a result, amongst the candidate models, the tenth order of polynomial function is the best model to regress the expressive timing within a phrase.

Considering the fact the phrase length of *Islamey*, Mazurka Op.24/2 and Mazurka Op.30/2 are 8 beats, 12 beats and 24 beats respectively and the most suitable polynomial function to regress expressive timing within a phrase is the second, the seventh and the tenth order, there may be a potential relationship between the most suitable order of polynomial function for regression and the phrase length. Demonstrating this hypothesis is beyond the scope of this paper but is possibly a future work.

5. DISCUSSION

5.1 Centroid Pairing

From the results of the model selection test, GMM$_r$ outperforms GMM$_o$. However, the resulting GMM$_r$ may not be necessary to make musical sense. As GMM$_o$ makes musical sense [4], the centroids of GMM$_r$ and GMM$_o$ are compared. If the regressed polynomial curves recovered from GMM$_r$ are correlated with the centroids of GMM$_o$, the GMM$_r$ will also make musical sense.

Recall that in Section 3, $\vec{\mu}_i$ represented the centroids of the GMMs for expressive timing within a phrase and $\vec{m}_j$ represented the centroids of the GMMs for regressed polynomial coefficients that can be recovered as a polynomial curve $f^o(x|\vec{m}_j)$. The similarity between centroids can be defined by the correlation coefficients (ρ) between $\vec{\mu}_i$ and $f^o(x|\vec{m}_j)$, namely

$$\rho(\vec{\mu}_i, f^o(x|\vec{m}_j)) = \frac{\sum_{k=1}^{n}(\mu_k - \overline{\vec{\mu}_i})(x_k - \overline{f^o(x|\vec{m}_j)})}{\sqrt{\sum_{k=1}^{n}(\mu_k - \overline{\vec{\mu}_i})^2}\sqrt{\sum_{k=1}^{n}(x_k - \overline{f^o(x|\vec{m}_j)})^2} \quad (7)$$

where $\overline{\vec{\mu}_i} = \frac{1}{n}\sum_{k=1}^{n}\mu_k$ and $\overline{f^o(x|\vec{m}_i)} = \frac{1}{n}\sum_{k=1}^{n}x_k$.

Suppose there are A Gaussian components in the GMM$_r$ and GMM$_o$. The centroids in GMM$_r$ and GMM$_o$ can be paired according to Algorithm 1. In Table 1, the results of pairing the centroids of the GMMs for the oth order polynomial coefficients are shown. In Figure 2, we demonstrate how the centroids of GMM$_o$ compared with the regressed polynomial curves with the centroids of GMM$_r^2$ and GMM$_r^7$ in Chopin's Mazurka Op.24 No.2. From the results we can see that the regressed polynomial curves recovered from the centroids of GMM$_r^2$ are highly correlated with GMM$_o$ while the regressed polynomial curves recovered from the centroids of GMM$_r^7$ are even more similar to GMM$_o$ due to the higher model complexity. Thus the results demonstrate that the GMMs for the regressed polynomial coefficients are musically valid.

Algorithm 1 Pair centroids
```
Require: $f^o(x|\vec{m})_i$,$i \in [1, A]$
Require: $\vec{\mu}_j$,$i \in [1, A]$
  $\mathbf{C}(i,j) = \rho(\vec{\mu}_i, f^o(x|\vec{m}_j))$
  while $\max(\mathbf{C}) \geq -1$ do
    (r,c)=arg max ($\mathbf{C}_{rc}$)
    pairs:=pairs ∪ {r,c}
    Associate $\mathbf{loc}_r$ with $\mathbf{loc}_c$
    $\mathbf{C}_{ri}$ = -2
    $\mathbf{C}_{jc}$ = -2
  end while
```

	Islamey	Op.24/2	Op.30/2
GMM$_r^2$	0.99	0.83	0.75
GMM$_r^3$	0.99	0.82	0.74
GMM$_r^4$	0.99	0.84	0.72
GMM$_r^5$	0.99	0.89	0.80
GMM$_r^6$	1.00	0.90	0.78
GMM$_r^7$	1.00	0.90	0.82
GMM$_r^8$	1.00	0.90	0.76
GMM$_r^9$	N/A	0.90	0.79
GMM$_r^{10}$	N/A	0.90	0.86

Table 1. The correlation coefficients between the polynomial curves recovered from the centroids of GMM$_r^o$ and the centroids of GMM$_o$.

5.2 GMM$_r$ with more clusters

With the results presented, we can conclude that with the same Gaussian components in the model, GMM$_g$ outperforms GMM$_o$ when the intended number of Gaussian components is decided by the GMM$_o$ provided by Li et al. [5]. In this section, we observe whether GMM$_r$, which has more Gaussian components, has a better performance. As an example, we compare the performance of GMM$_r^2$, measured by correlation coefficients with multiple Gaussian components. In Table 2, we show how well the regressed parabolic curves approximate the centroids of GMM$_g^2$ by showing $\rho(\mathbf{T_i}, f^o(x|\vec{m}_i))$ calculated by equation (5).

Clusters	*Islamey*	Op.24/2	Op.30/2
2	*0.5315*	0.5872	0.7339
4	0.5411	0.6181	*0.7399*
8	0.5718	*0.6877*	0.7442
16	0.6261	0.6930	0.7635
32	0.6722	0.7165	**0.7677**
64	0.6857	**0.7324**	0.7585
128	**0.6978**	0.7298	0.7413
256	0.6940	0.7282	N/A
512	0.6467	0.7109	N/A

Table 2. The average value of $\rho(\mathbf{T_i}, f^o(x|\vec{m}_i))$ resulting from GMM$_g^2$ with different numbers of Gaussian components (labelled as clusters in the table). A larger number means a better approximation and a better performance (bold). The number of clusters we set in the previous experiments are in italics. The training set of Mazurka Op.30/2 has less than 256 samples because it is impossible to set 256 and 512 clusters in the experiments.

From the table, we can see that the numbers of Gaussian components we engaged in the experiments in section 4 for GMM$_g^2$ do not have the best performance. Thus the GMM$_g^2$ with more Gaussian components can improve the model performance further.

6. CONCLUSIONS

In this paper, we demonstrate whether regressing standardised tempo curves within a phrase by a polynomial function is a valid method to analyse expressive timing by comparing Gaussian Mixture Models (GMMs) fitting expressive timing (GMM$_o$) and fitting regressed polynomial coefficients (GMM$_g$). As the candidate models fit different sets of data and there are no musicological ground truth for the clustering of expressive timing, the approximation of expressive timing by the centroids of GMM$_o$ and GMM$_r$ is used to evaluate model performance.

Measured by correlation coefficients and Euclidean distance, the experiment shows that GMM$_g$ outperforms GMM$_r$ when the same numbers of Gaussian components are engaged. With more Gaussian components engaged, GMM$_r$ performs even better. The distribution of regressed polynomial coefficients has a lower degree of freedom compared with the tempo curves representing expressive timing within a phrase hence the regression of expressive timing with polynomial function reduces data dimension. The results demonstrate that regressing expressive timing with polynomial functions may help the clustering process.

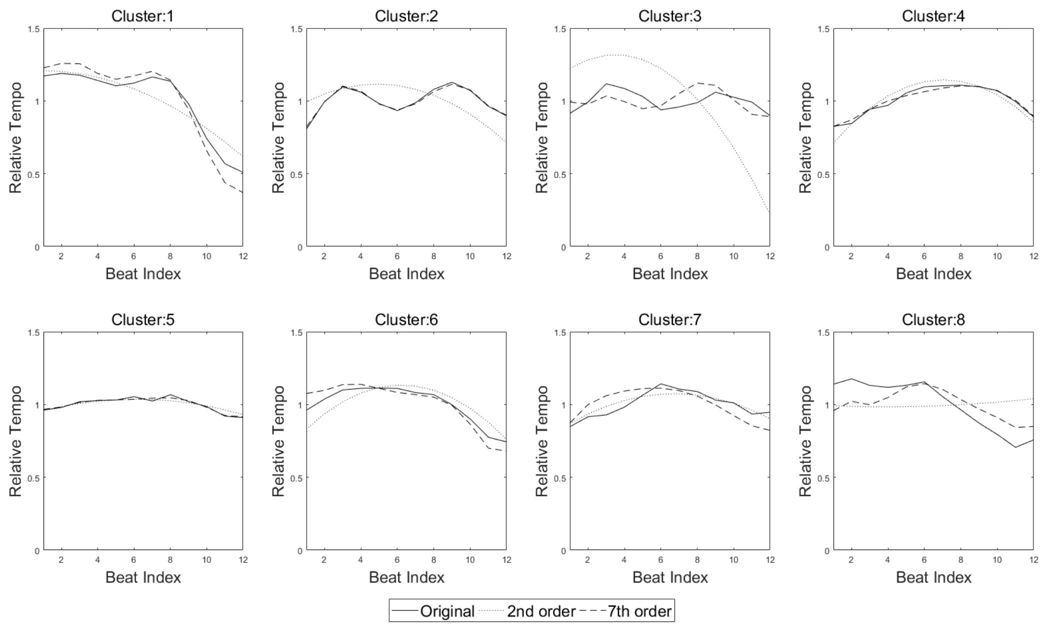


Figure 2. The centroids of GMM$_o$ compared with the regressed polynomial curves with the centroids of GMM$_r^2$ and GMM$_r^7$ in Chopin's Mazurka Op.24 No.2.

When comparing the regressed polynomial curves recovered from the centroids of GMM$_g$ with the centroids of GMM$_o$, the two sets of centroids are highly correlated with each other, which demonstrates that the centroids of GMM$_g$ make similar musical sense with GMM$_o$. As a result, the polynomial functions can be used to help cluster expressive timing, which makes clustering expressive timing across phrases with various lengths possible.

7. ACKNOWLEDGEMENT

As a study based on my PhD research, we would like to thank Dr. Dawn A. A. Black and Prof. Elaine Chew for initial discussions. Chinese Scholarship Council partially founded the initial stage of the research.

8. REFERENCES

[1] Kenneth P. Burnham and David R. Anderson. *Model Selection and Multimodel Inference — A Practical Information-Theoretic Approach*. Springer, 2nd edition, 2002.

[2] Emilios Cambouropoulos, Simon Dixon, Werner Goebl, and Gerhard Widmer. Human preferences for tempo smoothness. In *Proceedings of the VII International Symposium on Systematic and Comparative Musicology and III International Conference on Cognitive Musicology*, pages 18–26, 2001.

[3] Maarten Grachten, Werner Goebl, Sebastian Flossmann, and Gerhard Widmer. Phase-plane representation and visualization of gestural structure in expressive timing. *Journal of New Music Research*, 38(2):183–195, 2009.

[4] Shengchen Li, Dawn A. A. Black, Elaine Chew, and Mark D. Plumbley. Evidence that phrase-level tempo variation may be represented using a limited dictionary. In *Proceedings of International Conference on Music Perception and Cognition (ICMPC'14)*, 2014.

[5] Shengchen Li, Dawn A. A. Black, and Mark D. Plumbley. The clustering of expressive timing within a phrase in classical piano performances by Gaussian mixture models. *Lecture Notes in Computer Science, Post Conference Proceeding of International Symposium on Computer Music Multidisciplinary Research*, 9617:322–345, 2016.

[6] Shengchen Li, Simon Dixon, Dawn A. A. Black, and Mark D. Plumbley. A model selection test for factors affecting the choice of expressive timing clusters for a phrase. In *Proceedings of 13th Sound and Music Computing Conference*, 2016.

[7] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[8] Bruno H. Repp. A microcosm of musical expression. I. Quantitave analysis of pianists' timing in the initial measures of Chopin's Etude in E major. *The Journal of the Acoustical Society of America*, 104:1085–1100, 1998.

[9] Bruno H. Repp. A microcosm of musical expression: II. Quantitative analysis of pianists' dynamics in the initial measures of Chopin's Etude in E major. *The Journal of the Acoustical Society of America*, 105:1972–1988, 1999.

[10] Craig Sapp. Hybrid numeric/rank similarity metrics for musical performance analysis. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 501–506, 2008.

[11] Neta Spiro, Nicolas Gold, and John Rink. The form of performance: Analyzing pattern distribution in select recordings of Chopin's Mazurka op. 24 no. 2. *Musicae Scientiae*, 14(2):23–55, 2010.

[12] Renee Timmers. Vocal expression in recorded performances of Schubert songs. *Musicae Scientiae*, 11(2):237–268, 2007.

[13] Asmir Tobudic and Gerhard Widmer. Playing Mozart phrase by phrase. In *Proceedings of the 5th International Conference on Case-based Reasoning (ICCBR'03)*, pages 552–566. Springer, 2003.

[14] Neil P. Mcangus Todd. The dynamics of dynamics: A model of musical expression. *Journal of the Acoustical Society of America*, 91:3540–3550, 1992.

[15] Gerhard Widmer. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146:129–148, 2003.

[16] Gerhard Widmer and Asmir Tobudic. Playing Mozart by analogy: Learning multi-level timing and dynamics strategies. *Journal of New Music Research*, 32:259–268, 2003.

DISCOURSE ANALYSIS OF LYRIC AND LYRIC-BASED CLASSIFICATION OF MUSIC

Jiakun Fang[1] **David Grunberg**[1] **Diane Litman**[2]
Ye Wang[1]
[1] School of Computing, National University of Singapore, Singapore
[2] Department of Computer Science, University of Pittsburgh, USA
fangjiak@comp.nus.edu.sg, wangye@comp.nus.edu.sg

ABSTRACT

Lyrics play an important role in the semantics and the structure of many pieces of music. However, while many existing lyric analysis systems consider each sentence of a given set of lyrics separately, lyrics are more naturally understood as multi-sentence units, where the relations between sentences is a key factor. Here we describe a series of experiments using discourse-based features, which describe the relations between different sentences within a set of lyrics, for several common Music Information Retrieval tasks. We first investigate genre recognition and present evidence that incorporating discourse features allow for more accurate genre classification than single-sentence lyric features do. Similarly, we examine the problem of release date estimation by passing features to classifiers to determine the release period of a particular song, and again determine that an assistance from discourse-based features allow for superior classification relative to single-sentence lyric features alone. These results suggest that discourse-based features are potentially useful for Music Information Retrieval tasks.

1. INTRODUCTION

Acoustic features have been used as the basis for a wide variety of systems designed to perform various Music Information Retrieval (MIR) tasks, such as classifying music into various categories. However, a piece of music is not entirely defined by its acoustic signal, and so acoustic features alone may not contain sufficient information to allow for a system to accurately classify audio or perform other MIR tasks [24]. This has led to interest in analyzing other aspects of music signals, such as lyrics [16, 22].

Although not all music contains lyrics, for songs that do, lyrics have been proven to be useful for classifying audio based on topic [17], mood [15], genre, release date, and even popularity [7]. This is a natural result since humans also consider lyrics when performing these classifications. But while lyric features have been used in previous MIR studies, such works often use a bag-of-words or bag-of-sentences approach which considers each sentence within a set of lyrics independently. This approach sacrifices the contextual information provided by the lyrical structure, which often contains crucial information. As an example, we consider lyrics from the theme of Andy Williams' "A Summer Place":

- *Your arms reach out to me.*
- ***And** my heart is free from all care.*

The clause 'and' linking these two lines helps to set the mood; the listener can observe a connection between the subject reaching out to the singer, and the singer's heart consequently being at ease. But suppose the word 'and' were changed to the word 'but'. In this case, the meaning of these lyrics would be entirely different; now the singer's heart is at ease *despite* the subject reaching for him, not implicitly *because* of it. A human would no doubt observe this; however, this information would be lost with a bag-of-words or bag-of-sentences approach. We therefore hypothesize that lyrics features which operate on a discourse level, taking into account the relations between textual elements, will better represent the underlying structure of a set of lyrics, and that systems using such features will improve the performances of those using lyric features which consider each sentence independently.

In this paper we consider two classical MIR tasks: genre classification and release date estimation. Prior research has already demonstrated that lyrics-based features can improve accuracy for genre classification [22] as well as release date estimation [7]. This prior work considered individual words without taking into account how those words were linked together with discourse features or other connectors. However, it is already known that the complexity of lyrics often varies between different genres (e.g., rap music tends to have more complex lyrics than other genres [7]) as well as between different eras of music [9]. Lyrics of differing complexity are likely to have differing discourse connectors (e.g., very simple lyrics may only consist of a few unrelated elements and so have almost no discourse connectors, while dense, complicated lyrics may contain many elements which are connected together via discourse connectors), so we hypothesize that discourse connector features may also contribute to the above tasks. As such, we investigate whether discourse features truly improve the accuracy in genre recognition, release-date estimation, and popularity analysis.

2. RELATED WORKS

Discourse analysis is a process analyzing the meaning of a text by examining multiple component sentences together, rather than each sentence on its own [26]. One dimension of it is *discourse relations*, which describes how multiple elements of a text logically relate to each other, and different discourse relation corpora and frameworks have been devised, including Rhetorical Structure Theory [21], Graphbank [27] and the Penn Discourse Treebank (PDTB) [25]. We opted to use PDTB as it is relatively flexible compared to these other frameworks [23] and more able to accommodate a wider variety of lyrics structures.

Another aspect of discourse analysis is text segmentation. In prior MIR studies involving lyrics, acoustic elements were used to help determine lyric segmentation points [3]. However, this approach takes the risk that errors in the audio analysis will propagate through to the lyric segmentation step. In contrast, the algorithm TextTiling takes only text as input and attempts to detect the boundaries of different subtopics within that text in order to perform meaningful segmentation [13]. Because lyrics can change topics during a song, we determined that a topic-based system like TextTiling could provide useful segmentation for MIR systems operating on lyrics.

Coherence and cohesion of a text has been proven to be important for human understanding [12] and writing quality [4]. While text coherence is a subjective property of text based on human understanding, text cohesion is an objective property of explicit text element interpretation patterns [12]. Various studies focused on elements of this specific text analysis, including entity grid [1] and coreference resolution systems [18]. A study by Feng et al. [8] showed the appearance pattern of entities may vary according to different writing style. Therefore, we hypothesize that the cohesion patterns in lyrics may vary according to different categories, and we used entity density, entity grid and coreference chain for lyric cohesion analysis.

Many music classification tasks have been investigated in the field of MIR. However, most systems which incorporate lyrics do not incorporate discourse analysis; they instead rely on approaches such as analyzing bags of words, part-of-speech tags and rhyme [7, 16, 19]. There was still little analysis of the discourse relations, topic shifts or detailed cohesion analysis.

3. FEATURES

3.1 Discourse-based Features

PDTB-styled discourse relations: We used a PDTB-styled parser [1] [20] to generate discourse relation features. In this work, we only focus on explicit discourse relations, since implicit relations are both harder to accurately determine and more subjective. In order to find such explicit relations, the parser first identifies all connectives in a set of lyrics and determines whether each one serves as a discourse connective. The parser then identifies the explicit relation the connective conveys. The system considers four general relations and 16 specific relations which are subcategories of the 4 general relations.

As an example, we consider a lyric from John Lennon's "Just Like Starting Over": "*... I know time flies <u>so</u> quickly/ **<u>But</u>** **<u>when</u>** I see you darling/It's like we both are falling in love again...*" All three of the underlined words are connectives, but the first such word, 'so,' is not a discourse connective because it does not connect multiple arguments. The parser thus does not consider this word in its analysis. The other two connectives, 'but' and 'when', are discourse connectives and so are analyzed to determine what type of relation they are; 'when' is found to convey a Temporal (general) and Synchrony (specific) relation, and 'but' is determined to convey a Comparison and a Contrast relation. In this way, the connections between the different elements of this lyric are understood by the system.

Once all the discourse connectives are found and categorized, we obtain features by counting the number of discourse connectives in each set of lyrics which corresponds to a particular discourse relation. For instance, one song might have 18 discourse connectives indicating a Temporal relation, so its Temporal feature would be set to 18. We also count the number of pairs of adjacent discourse connectives which correspond to particular relations and these adjacent discourse connectives are not necessary consecutive tokens; the same song as before might have 5 instances where one discourse connective indicates a 'Temporal' relation and the next discourse connective indicates a 'Comparison' relation, so its Temporal-Comparison feature would be set to 5. This process is performed independently for the general and the specific relations. Ultimately, we obtain 20 features corresponding to the 4 general relations (4 individual relations and 16 pairs of relations), and 272 features corresponding to the 16 specific relations (16 individual relations, and 256 pairs of relations). After removing features which are zero throughout the entire dataset, 164 features corresponding to specific relations remain. Finally, we calculate the mean and standard deviation of the sentence positions of all discourse connectives in a set of lyrics, as well as all connectives in that set of lyrics in general.

TextTiling segmentation: We ran the TextTiling algorithm to estimate topic shifts within a piece of lyric, using the Natural Language Toolkit Library [2], setting the pseudo-sentence size to the average length of a line and grouping 4 pseudo-sentences per block. Lyrics with fewer than 28 words and 4 pseudo-sentences were set as one segment, since they were too short for segmentation, and lyrics with no line splits were arbitrarily assigned a pseudo-sentence size of 7 words (average length in the dataset). Features were then calculated by computing the mean and

[1] http://wing.comp.nus.edu.sg/ linzihen/parser/

[2] http://www.nltk.org/api/nltk.tokenize.html

standard deviation in the number of words in a lyric's segments and the number of segments.

Entity-density features: General nouns and named entities (i.e., locations and names) usually indicate conceptual information. Previous research have shown that named entities are useful to convey summarized ideas [11] and we hypothesized that entity distribution could vary between song categories. We implemented features including: ratio of the number of named entities to the number of all words, ratio of the number of named entities to the number of all entities, ratio of the number of union of named entities and general nouns to the number of all entities, average number of named entities per sentence, and average number of all entities per sentence. We used OpenNLP [3] to find named entities and Stanford Part-Of-Speech Tagger [4] to extract general nouns.

Coreference inference features: Entities and their pronominal references in a text which represent a same object build a coreference chain [18]. The pattern of how an entity represented by different text elements with same semantic meanings through text may vary in different song styles. We used Stanford Coreference Resolution System [5] to generate coreference chain. The total number of coreference chains, the number of coreference chains which span more than half of lyric length, the average number of coreferences per chain, the average length per chain, the average inference distance per chain and the number of active coreference chains per word were extracted. The inference distance was computed as the minimum line distance between the referent and its pronominal reference. The chain is active on a word if the chain passes its location.

Entity-grid features: Barzilay and Lapata's [1] entity grid model was created to measure discourse coherence and can be used for authorship attribution [8]. We thus hypothesized that subjects and objects may also be related differently in different genres, just as they may be related differently for artists. Brown Coherence Toolkit [6] was used to generate an entity grid for each lyric. Each cell in a grid represent one of the roles of subject (S), object (O), neither of the two (X) and absent in the sentence (-) of a entity in a sentence. We calculated the frequency of 16 adjacent entity transition patterns (i.e., 'SS', 'SO', 'SX' and 'S-') and the number of total adjacent transitions, and computed percentage of each pattern.

3.2 Baseline: Previously Used Textual Features

We selected several lyric-based features from the MIR literature to form comparative baselines against which the discourse-based features could be tested (Table 1) [7]:

Vocabulary: We used the Scikit-learn library [6] to calculate the top 100 n-grams (n = 1, 2, 3) according to their tf-idf values. When performing genre classification, we obtained the top 100 unigrams, bigrams, and trigrams for the lyrics belonging to each genre. When performing year classification, we obtained approximately 300 n-gram features evenly from three year classes. These n-grams were represented by a feature vector indicating the importance of each n-gram in each lyric. We also computed the type/token ratio to represent vocabulary richness and searched for non-standard words by finding the percentage of words in each lyric that could be found in the Urban Dictionary [7], a dictionary of slang, but not in Wiktionary [8].

Part-of-Speech features: We used Part-of-Speech tags (POS tags) obtained from the Stanford POS Tagger [9] to determine the frequencies of each super-tags (Adjective, Adverb, Verb and Noun) in lyrics.

Length: Length features such as lines per song, tokens per song, and tokens per line were calculated.

Orientation: The frequency of first, second and third pronouns as well as the ratio of self-referencing pronouns to non-self-referencing ones and the ratio of first person singular pronouns to second person were used to model the subject of given sets of lyrics. We also calculated the ratio of past tense verbs to all verbs to quantify the overall tense of songs.

Structure: Each set of lyrics was checked against itself for repetition. If the title appeared in the lyrics, the title feature for that song was given a 'True' value, which was otherwise set to false. Similarly, if there were long sequences which exactly matched each other, the 'Chorus' feature was set to 'True' for a given song. Table 1 shows the number of elements in each feature set in the classification tasks.

Dimension	**Abbreviation**	**Length**
discourse-based features	DF	250
PDTB-based discourse relation	DR	204
TextTiling segmentation	TT	3
entity density	ED	5
coreference inference	CI	5
entity grid	EG	33
textual baseline features	TF	318
vocabulary	VOCAB	303
POS tags	POS	4
length	LEN	3
orientation	OR	6
structure	STRUC	2

Table 1: Features used in classification tasks.

3.3 Normalization

Since features used for these tasks are not on the same scale, we then performed normalization on features. Each feature was normalized by its maximum value and minimum value to range from 0 to 1 (Equation 1). Then all normalized features were put into classification tasks. This normalization step was expected to improve the results of

[3] https://opennlp.apache.org
[4] http://nlp.stanford.edu/software/tagger.shtml
[5] http://nlp.stanford.edu/projects/coref.shtml
[6] http://scikitlearn.org/stable/modules/feature_extraction.html
[7] http://www.urbandictionary.com
[8] https://www.wiktionary.org
[9] http://nlp.stanford.edu/software/tagger.shtml

combination of different feature sets, as differences in variable ranges could potentially affect negatively to the performance of classification algorithm.

$$v_n = \frac{v - v_{min}}{v_{max} - v_{min}} \quad (1)$$

4. DATASET AND ANNOTATION

A previously collected corpus of 275,905 sets of full lyrics was used for these experiments and we pre-processed the dataset in 6 different types to clean up lyrics [5], including splitting of compounds or removal of hyphenated prefixes, elimination of contractions, restoration of dropped initial, abbreviation elimination, adjustment to American English spellings, and correction of misspelled words. Unlike other corpora, such as musiXmatch lyrics dataset for the Million Song Dataset [2], lyrics from the selected corpus are not bags-of-words but are stored in full sentences, allowing for the retention of discourse relations. We split song lyrics by punctuations and lines to make sentences and paragraphs to run discourse analysis algorithm in this work. We also downloaded corresponding genre tags and album release years for the songs represented in this dataset from Rovi [10]. The specific number of lyrics for each experiment is shown in Table 2.

Genre classification: We kept all 70,225 songs with a unique genre tag from Rovi for this specific task. The tags indicated that songs in the dataset came from 9 different genres: Pop/Rock (47,715 songs in the dataset), Rap (8,274), Country (6,025), R&B (4,095), Electronic (1,202), Religious (1,467), Folk (350), Jazz (651) and Reggae (446). All of these songs were then used for the genre classification experiments.

Release date estimation: Rovi provided release dates for 52,244 unique lyrics in the dataset. These release dates ranged from 1954-2014. However, some genres were not represented in certain years; no R&B songs, for instance, had release dates after 2010, and no rap songs had release dates before 1980. To prevent this from biasing our results we chose to just use one single genre and settled on Pop/Rock, for which we had 46,957 songs annotated with release dates throughout the 1954-2014 range. We then extracted all the songs labeled as having been released in one of three time ranges: 1969-1971 (536 songs total), 1989-1991 (3,027), and 2009-2011 (4,382). We put gaps of several years between each range on the basis that, as indicated in prior literature, lyrics are unlikely to change much in a single year [7].

5. GENRE CLASSIFICATION

We ran SVM classifiers using 10-fold cross-validation. These classifiers were implemented with Weka [11] using the default settings. We chose SVM classifiers because they have been proven to be of use in multiple MIR tasks [7, 15]. Because each genre had a different number of

[10] http://developer.rovicorp.com

[11] http://cs.waikato.ac.nz/ml/weka

Classification Task	Number of lyric used (after undersampling)
Genre	Pop/Rock: 45,020; Rap: 16,548; Country: 12,050; Jazz: 1,302; R&B: 8,190; Electronic: 2,404; Religious: 2,934; Folk: 700; Reggae: 892
Release Period	1,608 sets of lyrics, split evenly into three time spans

Table 2: Data sizes for experiments.

samples, undersampling [10] was performed for both training and testing to ensure that each genre was represented equally before cross-validation classification. Each song was classified in a 2-class problem: to determine if the song was of the correct genre or not. The undersampling and classification process was repeated 10 times and we present the averages of F-score for each independent classification task. The value of F-score by random should be 0.5.

We first implemented previously-used textual features to generate a baseline for the genre classification task. Models were built based on vocabulary (VOCAB), POS tags (POS), length (LEN), orientation (OR), structure (STRUC) and all combined baseline features (TF) separately. The average F-scores are depicted in Table 3. It is apparent that using vocabulary features can achieve high performance in average, but one thing to be noted is that it heavily depends on which corpus the language model trains on to generate the n-gram vector. Here we used all lyrics from each genre to get top n-grams. Orientation features were useful for R&B recognition since we found more first pronouns in such genre. We then used these features to compare with proposed discourse-based features.

We then evaluated the utility of discourse-based features for this specific task. Table 3 presents the results from using discourse relation (DR), TextTiling topic segmentation (TT), entity density (ED), coreference inference (CI), and entity grid (EG) features to perform genre classification with the SVM classifiers. Because the discourse relation and TextTiling features showed very promising results, we also tested a system which combined those features (DR+TT). Finally, we tested all discourse features together (DF), and then all discourse and all baseline features together. Statistical significance were computed using a standard two-class t-test between the highest F-score and each result from other feature set for each genre, and each column's best result were found to be significant with $p < 0.01$.

First, we note that, for every single genre as well as the overall average, the system's classification accuracy when using DR+TT discourse features is better than its accuracy using any and all baseline features. In fact, DR features alone outperform any and all baseline features for 7 of the 9 genres as well as overall. This serves to demonstrate the utility of these particular discourse features for

Feature Set	R&B	Folk	Country	Rap	Elect.	Reli.	Jazz	Reggae	Pop	Avg.
VOCAB	58.5	51.4	59.4	**90.8**	53.7	53.5	55.3	60.7	65.7	61.0
POS	55.4	47.3	53.6	73.1	49.9	50.3	56.3	47.4	60.0	54.8
LEN	55.2	49.3	55.4	85.8	48.6	50.0	50.3	48.8	59.2	55.4
OR	66.0	54.7	58.1	84.6	54.4	52.6	58.7	54.9	63.4	60.8
STRUC	45.0	46.4	44.5	45.6	46.0	45.7	45.3	47.0	44.6	45.6
TF (All)	62.5	56.5	60.1	81.3	50.7	51.8	58.1	56.5	63.6	60.1
DR	64.9	**61.7**	65.7	89.8	**59.1**	**56.2**	**62.8**	**64.0**	66.7	**65.7**
TT	63.3	51.1	58.2	90.4	53.1	53.0	58.0	55.9	65.9	61.0
ED	55.4	58.3	53.2	76.5	53.8	53.7	46.8	57.1	61.2	57.3
CI	59.1	47.8	62.7	82.4	50.5	52.8	55.7	54.1	63.7	58.8
EG	58.7	48.3	57.1	83.9	50.5	52.6	54.9	51.4	62.9	57.8
DR + TT	**67.4**	59.1	**66.6**	**91.0**	58.3	55.3	62.3	62.3	**67.7**	**65.6**
DF (All)	58.2	53.3	60.9	75.8	49.9	54.0	57.5	49.1	61.5	57.8
All	50.0	34.5	35.7	49.6	45.2	48.3	41.1	49.4	45.8	44.4

Table 3: Accuracy of classifier using different unnormalized feature sets to estimate genre (F-Score*100).

Feature Set	R&B	Folk	Country	Rap	Elect.	Reli.	Jazz	Reggae	Pop	Avg.
VOCAB	59.3	55.6	61.0	91.3	52.7	63.0	61.8	65.1	66.5	64.4
POS	63.5	57.8	55.9	90.9	49.4	48.9	61.8	61.6	65.3	62.4
LEN	61.9	50.5	59.4	86.7	49.2	49.1	61.1	59.4	63.5	60.2
OR	68.2	55.8	55.1	85.4	47.3	46.6	60.0	55.7	64.3	60.4
STRUC	46.9	45.1	45.8	45.8	46.9	44.8	43.8	47.1	44.6	45.6
TF (All)	71.1	59.6	67.4	93.3	55.4	65.0	65.6	68.7	68.3	68.4
DR	60.9	59.0	62.3	88.4	54.9	54.6	61.1	61.0	64.7	63.1
TT	64.1	49.8	54.6	90.9	48.7	51.0	62.7	60.6	66.0	61.7
ED	37.5	45.2	38.3	65.5	45.1	45.5	47.8	47.3	51.6	48.2
CI	63.5	53.2	61.5	84.5	50.5	55.1	62.2	63.7	62.2	61.9
EG	63.7	55.5	64.5	94.1	57.8	49.5	65.5	62.1	64.4	64.1
DF (All)	71.2	**61.3**	67.3	**94.5**	**58.5**	58.5	64.5	66.5	66.3	67.7
All	**73.7**	60.6	**71.5**	**94.8**	**58.9**	**65.6**	**66.9**	**69.6**	**69.4**	**69.9**

Table 4: Accuracy of classifier using different normalized feature sets to estimate genre (F-Score*100).

this task, since they consistently outperform the baseline features. Second, we note that the entity and coreference features did not enable the classifier to achieve maximal results in this task, indicating that these features may not vary as much between genres compared to the DR and TT features. Third, we note that the system's accuracy when all features was used decreased relative to the DR+TT and DR features in every case. We then performed the normalization and each feature was normalized by its maximum value and minimum value to range from 0 to 1.

Table 4 shows the results and the combination of all feature outperformed all baseline features, while the combination of all discourse-based features can achieve higher performance than all baseline feature sets in 3 classes. Best result for each genre were found to be significant with $p < 0.01$. This further emphasized the importance of discourse-based features in this specific task.

One interesting trend in these results is in the 'Rap' column, which shows that not only was the classification accuracy for Rap songs far higher than the other classes, but it was also the one genre where TT features outperformed DR features. Although the discourse-based features did not outperform the baseline features in this genre, it should be noted that the TextTiling segmentation features did obtain virtually identical performance to the best baseline features with only a 3-dimensional feature vector; the VOCAB features, by contrast, encompassed hundreds of dimensions. We investigated this further and found that Rap music tended to have more topic segments (5.9/song on average, while the average for other genres was 4.9), and more varied adjacent discourse relations as well (for instance, each rap song had on average 6.6 different types of adjacent discourse relations; non-rap songs averaged 4.0). This suggests that TextTiling segmentation features may be a more compact way to accurately represent topic-heavy lyrics, such as those commonly found in rap music.

We finally analyzed the portion of each type of discourse connective for the four first-level PDTB-styled discourse relations of all discourse connectives in each genre. We found that Religious songs use more expansion relations than other genres (42% and 37% in average), while less expansion relations are written in Rap songs (34%).

Connectives standing for temporal relations present more in Rap songs (26% and 23% in average). R&B songs contains more contingency connectives (24% and 26% in average).

6. RELEASE DATE ESTIMATION

We investigated whether discourse-based features can help to estimate the release date of a song, on the basis that the lyric structure of song texts is likely to change over time [7, 14]. We first formed a subset of all the Pop/Rock songs in our dataset, since as mentioned before these songs spanned a greater time period than the other genres. We then extracted all the songs labeled as having been released in one of three time ranges: 1969-1971 (536), 1989-1991 (3,027), and 2009-2011 (4,382). Based on the idea from prior study [7], we made gaps since that the lyrics would be unlikely to change very much in a single year. Undersampling was used to balance the dataset building a sub-dataset before each classification with an SVM with 10-fold cross validation for three-class classification. The process was repeated 10 times.

Table 5 shows results. As can be seen from the table, discourse relation features alone outperformed the baseline feature sets in average F-score for each three year class ($p < 0.001$), which indicates that the sentence relations in lyrics likely vary over years, and that discourse relation features are useful at indicating this. Although not as much as the discourse relation features, the topic segments and coreference inference features contribute to this specific classification task as well, showing topic presentation and cohesion structure changed over time. TextTiling features proved to increase accuracy for one year range, 2009-2011, indicating that the number and relations of topics of music released in this era likely varied as compared to previous eras, and also that text segmentation-based features are useful in noting this change. The number of topics and the number of words in each topics in average increases over time. As for the coreference inference features, the number of coreference chains and the number of long coreference chains showed raising values according to release periods. More coreference chains and long coreference appeared more often in the recent years, indicating a fluent and centric content. The other discourse features were again shown to be less useful than these ones. Finally, the early ages and recent ages were more likely to be recognized, while the middle ages generally achieved the lowest F-scores among all feature sets except structure features. This result is intuitive; music will likely be more similar to music that were produced closer together.

We then normalized to 0 to 1 for all features and repeated the task to show whether discourse features can improve the performance of baseline features for this task. Table 6 shows that the combination of all features outperformed the other feature sets in this three-class classification task ($p < 0.001$).

Feature	1969-1971	1989-1991	2009-2011	Avg.
VOCAB	46.8	33.7	34.9	38.5
POS	30.0	24.5	52.8	35.8
LEN	34.6	26.7	50.6	37.3
OR	43.4	32.0	50.6	42.0
STRUC	0.00	29.1	50.7	26.6
TF (All)	42.2	27.6	53.6	41.2
DR	**59.7**	**43.0**	55.0	**52.6**
TT	46.5	34.8	47.6	43.0
ED	40.4	29.5	41.7	37.2
CI	47.7	29.3	53.8	43.6
EG	41.2	32.5	44.3	39.4
DR + TT	58.5	40.7	**56.3**	51.8
DF (All)	43.3	28.3	53.8	41.8
All	36.2	30.6	30.4	32.4

Table 5: Accuracy of classifier using different unnormalized feature sets to estimate release date (F-Score*100).

Feature	1969-1971	1989-1991	2009-2011	Avg.
VOCAB	51.4	41.6	42.3	45.1
POS	58.7	24.5	46.7	43.3
LEN	61.4	27.9	45.8	45.0
OR	58.1	17.4	48.3	41.3
STRUC	0.0	22.0	**87.3**	36.4
TF (All)	**63.4**	42.0	53.1	52.8
DR	57.6	34.5	47.7	46.6
TT	59.9	29.9	37.8	42.5
ED	30.0	16.3	47.4	31.2
CI	62.0	27.2	52.3	47.2
EG	57.4	46.6	42.0	48.7
DF (All)	57.0	44.9	48.8	50.3
All	61.0	**48.8**	54.7	**54.7**

Table 6: Accuracy of classifier using different normalized feature sets to estimate release date (F-Score*100).

7. CONCLUSION AND FUTURE WORK

We investigated the usefulness of discourse-based features and demonstrated that such features can provide useful information for two MIR classification tasks. Genre classification and release date estimation were all enhanced by incorporating discourse features into the classifiers. However, since discourse-based features rely on passages with multiple text elements, it may be noisy when used on music with short lyrics. As this work is an exploration work, further analysis is required. For instance, we split song lyrics by lines and punctuations in this work, which fitted most of the cases in our dataset. The split rules of sentences can influence the results from discourse analysis algorithms.It will be potentially useful to use these features for other MIR tasks such as keyword extraction and topic classification. In the future, we will explore all these discourse-based features on other MIR tasks and find sensible sets of features and fusion strategies for further improving performance for these tasks.

8. REFERENCES

[1] R. Barzilay and M. Lapata. Modeling local coherence: an entity-based approach. *Computational Linguistics*, 34(1):141–148, 2008.

[2] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proceedings of International Society for Music Information Retrieval Conference*, pages 591–596, 2011.

[3] H. T. Cheng, Y. H. Yang, Y. C. Lin, and H. H.Chen. Multimodal structure segmentation and analysis of music using audio and textual information. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 1677–1680, 2009.

[4] S. A. Crossley and D. S. Mcnamara. Cohesion, coherence, and expert evaluations of writing proficiency. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, pages 984–989, 2010.

[5] R. J. Ellis, Z. Xing, J. Fang, and Y. Wang. Quantifying lexical novelty in song lyrics. In *Proceedings of International Society for Music Information Retrieval Conference*, pages 694–700, 2015.

[6] M. Elsner, J. Austerweil, and E. Charniak. A unified local and global model for discourse coherence. In *Proceedings of the Conference on Human Language Technology and North American Chapter of the Association for Computational Linguistics*, pages 436–447, 2007.

[7] M. Fell and C. Sporleder. Lyrics-based analysis and classification of music. In *Proceedings of International Conference on Computational Linguistics*, pages 620–631, 2014.

[8] V. W. Feng and G. Hirst. Patterns of local discourse coherence as a feature for authorship attribution. *Literary and Linguistic Computing*, 29(2):191–198, 2014.

[9] Y. Gao, J. Harden, V. Hrdinka, and C. Linn. Lyric complexity and song popularity: Analysis of lyric composition and relation among billboard top 100 songs. In *SAS Global Forum*, 2016.

[10] J. D. Gibbons and S. Chakraborti. *Nonparametric Statistical Inference*. Chapman & Hall, London, 2011.

[11] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 121–128, 1999.

[12] A. C. Graesser, D. S. Mcnamara, M. M. Louwerse, and Z. Cai. Coh-metrix: Analysis of text on cohesion and language. *Behavior Research Methods Instruments and Computers*, 36(2):193–202, 2004.

[13] M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64, 1997.

[14] H. Hirjee and D. G. Brown. Using automated rhyme detection to characterize rhyming style in rap music. *Empirical Musicology Review*, 5(4):121–145, 2010.

[15] X. Hu and J. S. Downie. When lyrics outperform audio for music mood classification: A feature analysis. In *Proceedings of International Society for Music Information Retrieval Conference*, pages 619–624, 2010.

[16] X. Hu, J. S. Downie, and A. F. Stephen. Lyric text mining in music mood classification. In *Proceedings of International Society for Music Information Retrieval Conference*, pages 411–416, 2009.

[17] F. Kleedorfer, P. Knees, and T. Pohle. Oh oh oh whoah! towards automatic topic detection in song lyrics. In *Proceedings of International Society for Music Information Retrieval Conference*, pages 287–292, 2008.

[18] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, 2011.

[19] T. Li and M. Ogihara. Music artist style identification by semi-supervised learning from both lyrics and content. In *Proceedings of the 12th Annual ACM International Conference on Multimedia*, pages 364–367, 2004.

[20] Z. Lin, H. T. Ng, and M. Y. Kan. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(2):151–184, 2014.

[21] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

[22] R. Neumayer and A. Rauber. Integration of text and audio features for genre classification in music information retrieval. In *Proceedings of the European Conference on Information Retrieval*, pages 724–727, 2007.

[23] J. P. Ng, M. Y. Kan, Z. Lin, V. W. Feng, B. Chen, J. Su, and C. L. Tan. Exploiting discourse analysis for article-wide temporal classificatio. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 12–23, 2013.

[24] F. Pachet and J. J. Aucouturier. Improving timbre similarity: How high is the sky. *Journal of Negative Results in Speech and Audio Sciences*, 1(1):1–13, 2004.

[25] R. Prasad, N. Dinesh, A. Lee, E. Miltsakaki, L. Robaldo, A. K. Joshi, and B. L. Webber. The penn discourse treebank 2.0. In *Proceedings of Language*

Resources and Evaluation Conference, pages 2961–2968, 2008.

[26] B. Webber, M. Egg, and V.Kordoni. Discourse structure and language technology. *Natural Language Engineering*, 18(4):1–54, 2012.

[27] F. Wolf and E. Gibson. Representing discourse coherence: A corpus-based study. *Computational Linguistics*, 31(2):249–287, 2005.

END-TO-END OPTICAL MUSIC RECOGNITION USING NEURAL NETWORKS

Jorge Calvo-Zaragoza
Centre for Interdisciplinary Research in Music Media and Technology, McGill University
Montreal, QC, Canada

Jose J. Valero-Mas, Antonio Pertusa
Software and Computing Systems
University of Alicante
Alicante, Spain

ABSTRACT

This work addresses the Optical Music Recognition (OMR) task in an end-to-end fashion using neural networks. The proposed architecture is based on a Recurrent Convolutional Neural Network topology that takes as input an image of a monophonic score and retrieves a sequence of music symbols as output. In the first stage, a series of convolutional filters are trained to extract meaningful features of the input image, and then a recurrent block models the sequential nature of music. The system is trained using a Connectionist Temporal Classification loss function, which avoids the need for a frame-by-frame alignment between the image and the ground-truth music symbols. Experimentation has been carried on a set of 90,000 synthetic monophonic music scores with more than 50 different possible labels. Results obtained depict classification error rates around 2 % at symbol level, thus proving the potential of the proposed end-to-end architecture for OMR. The source code, dataset, and trained models are publicly released for reproducible research and future comparison purposes.

1. INTRODUCTION

Large-scale analysis of music is of great interest, and so many computational tools have been developed for such purpose. Quite often, the bottleneck for exploiting these ideas is the lack of large corpora of symbolic music.

The transcription of sheet music into some machine-readable format can be carried out manually. However, the complexity of music notation inevitably leads to burdensome software for music score editing, which makes the whole process very time-consuming and prone to errors. As a consequence, the development of automatic transcription systems for musical documents is gaining importance over the last years.

The field devoted to address this task is known as Optical Music Recognition (OMR) [1]. Typically, an OMR tool takes an image of a music score and provides its symbolic content encoded in some structured digital format such as MEI or MusicXML. Unfortunately, OMR is a challenging problem, and results have not been very promising so far [18].

The process of automatically recognizing the content of a music score is complex, and therefore the workflow of an OMR system is very extensive. Previous proposals related to this task focus on specific aspects of the pipeline, such as the binarization of the image [14], the detection of the staves [2], the separation between lyrics and music [3], the staff-line removal [8]—which may be even considered as a task by itself [7]—or the classification of isolated symbols [17]. It therefore comes as no surprise that no work have directly addressed the whole OMR process for modern western notation. We only find full recognition proposals for old music [5,15,16] that, in spite of involving music notation, entails a very different challenge.

One of the practical aspects that constrains end-to-end OMR research is the difficulty of obtaining an aligned dataset containing the labeled music symbols along with their exact position in the image of the score. Note that, from a musical perspective, it is not necessary to retrieve the exact position of each music symbol in the image since the important information is the succession of the music figures. Thus, it seems interesting to tackle the OMR task in an holistic fashion, in which the output is directly the sequence of symbols present in the score image disregarding their exact position in pixels.

Our work aims at setting the basis towards the development of systems that can directly work with a greater part of the OMR workflow. For that, we propose the use of recurrent neural networks, which have been applied with great success to many sequential recognition task such as speech recognition [11], handwriting recognition [12], or automatic music transcription [20]. The premise is that the network works on a single staff section, much in the same way as most Optical Character Recognition systems focuses on recognizing words appearing in a given line image [21,23].

The traditional limitation of such type of networks is that they require a strongly-aligned training set, i.e., the network has to be provided with the desired output of the recurring block for every single input frame of the image. This constraint has typically led to consider other sequential models such as hidden Markov Models, which can be trained with just pairs of input images and transcript sequences. Nonetheless, Graves et al. [10] proposed a method to train recurrent networks with unaligned data known as Connectionist Temporal Classification (CTC). The CTC is actually a loss function that focuses on the desired output sequence, regardless of which frames output each symbol.

For the precise case of this work, we rely on the Convolutional Recurrent Neural Network (CRNN) architecture for scene text recognition proposed by Shi et al. [19]. A CRNN is a deep neural network that comprises a series of convolutional layers, which focus on learning a suitable representation of the input image, followed by recurrent layers, which deal with the sequential nature of the task. In order to jointly train the network in an end-to-end fashion, the CTC loss function is considered.

Besides text recognition, Shi et al. also evaluated CRNN with a small number of music scores, just to assess its capabilities for any sequence-based task. Taking this work as a starting point, we further study the potential of the mentioned end-to-end CRNN model for the case of OMR. More precisely, our contributions are: (i) the redesign and optimization of the original CRNN architecture for this particular task; (ii) a thorough and quantitative assessment of the proposed architecture in terms of a large collection of more than 90,000 monophonic music scores.

The rest of the paper is structured as follows: Section 2 describes the details of the corpus created for this work; Section 3 describes the end-to-end model proposed; the evaluation procedure as well as the results obtained are shown and discussed in Section 4; finally, Section 5 concludes the work and proposes future lines to address.

2. CORPUS GENERATION

For assessing the proposed scheme we generated a set of monophonic score images together with their ground-truth annotations disregarding any frame-level alignment for the case of end-to-end training. This set contains 94,984 random sequences from a vocabulary of 52 Common Western Music Notation symbols: music notes from C4 to E5 (10 pitches), four possible note durations (half, quarter, eighth, and sixteenth) and their four respective silences, three time signatures (3/4, 4/4, and 6/8), accidentals (sharp, flat, and natural), the treble clef, and the bar line.

All the scores follow this structure: an initial clef; a set of alterations for the key of the piece; the time signature; the music content, being always the bar line annotated as it constitutes a symbol to be recognized. Note that bar lines are not randomly placed in the score but in their corresponding positions at the end of each complete bar.

The length of the generated sequences is random, with a minimum length of 4 symbols and a maximum of 37. Figure 1 shows a histogram of the length of the produced sequences.

The generation of the music content is random, i.e., no restriction is imposed about the pitch interval between two consecutive notes or their respective duration. Similarly, accidentals are randomly applied to further increase the variability in the scores. Given a sequence of music sym-

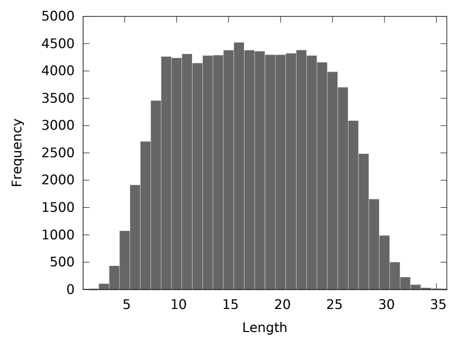


Figure 1. Histogram of the length of the sequences of the corpus.

bols we generated the image scores with the music engraving software Lilypond [1] . Figure 2 shows two examples of music scores along with their ground-truth annotations.

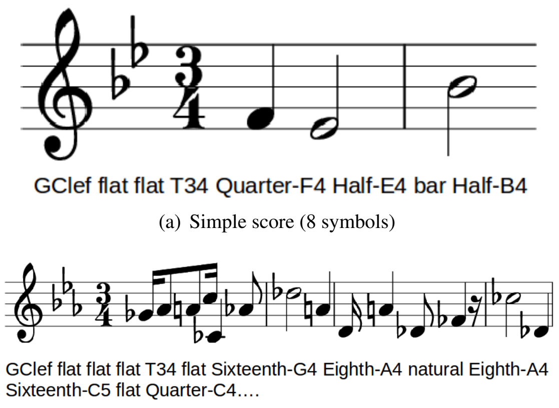

GClef flat flat T34 Quarter-F4 Half-E4 bar Half-B4
(a) Simple score (8 symbols)
GClef flat flat flat T34 flat Sixteenth-G4 Eighth-A4 natural Eighth-A4 Sixteenth-C5 flat Quarter-C4....
(b) Challenging score (34 symbols)

Figure 2. Example of scores depicting different levels of difficulty from our collection, along with its associated ground-truth.

3. FRAMEWORK

Our OMR approach is based on a Convolutional Recurrent Neural Network (CRNN) which takes as input an image of a monophonic staff section and directly outputs the sequence of music symbols, with no previous symbol segmentation or staff-line removal process. A conceptual scheme is illustrated in Figure 3.

Before the actual CRNN, we assume that a preprocessing step identifies and segments the different monophonic staff sections from the initial image for processing them independently. While this may be seen as a strong assumption, there exist algorithms in the literature that successfully address this task [6]. Once this monophonic staff section is segmented, the resulting image is normalized (pixel values between 0 and 1), rescaled to an aspect ratio of 1:4

[1] `http://lilypond.org/`

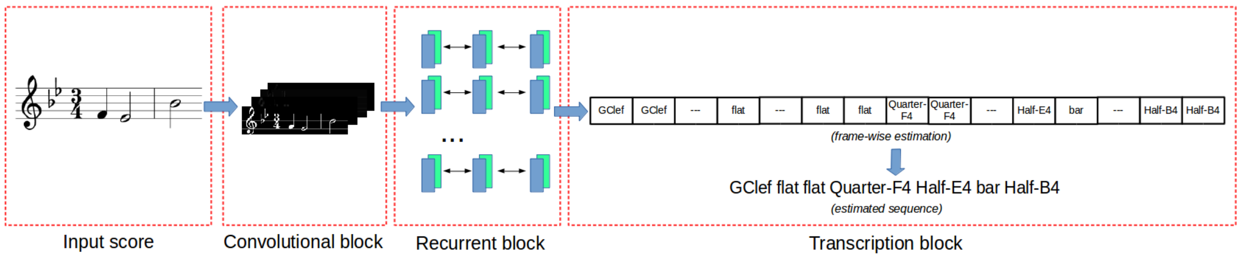


Figure 3. Conceptual scheme of the proposed approach. The input score is processed with a series of convolutional filters; the resulting features are then processed by the recurrent layers to model the temporal context of the piece; a frame-wise transcription using CTC is performed to obtain the estimation in an end-to-end fashion.

(i.e., the width is four times the height), and used as input to the CRNN. We established that this ratio is adequate for the task at issue by means of informal testing.

Table 1 shows the specific details of the proposed CRNN architecture, whose configuration and parameterization were determined experimentally. First, the image is processed with a series of convolutional layers which use Rectifier Linear Unit (ReLU) activation functions, followed by max pooling layers. Then, the output of the convolutional block is reshaped to serve as input to a recurrent neural network block, which is composed of three Bidirectional Long-Short Term Memory (BLSTM) networks [9, 13] with 256 hidden units. Finally, a fully-connected layer with a SoftMax activation function is added to retrieve the most likely class of each frame.

The CRNN model is trained using a batch size of 32 samples (i.e., 32 monophonic staff sections), RMSprop as the gradient descent method, and the aforementioned CTC as the loss function. We set 20 epochs for the training of the model and selecting the configuration that minimizes the validation error.

Note that the output of the CRNN is a framewise prediction that must be processed to obtain the actual output symbol sequence. However, this process is very straightforward because the CTC loss function forces the network to predict a *blank* symbol to indicate the separation between consecutive symbols [10].

4. EVALUATION

4.1 Partitions

We split the generated corpus in three fixed partitions: training and validation, which are meant to train the model and select the most appropriate hyper-parameters of the network, and a test partition to eventually assess the performance of the system. These sets represent the 60 $\%$, 20 $\%$, and 20 $\%$ out of the total set of available scores, respectively.

Table 2 describes these partitions in terms of the number of scores, measures, and running symbols in each of them. It must be noted that at least one element of the vocabulary appears in all the partitions, and so there are no out-of-vocabulary elements.

4.2 Metrics

In order to assess the performance of the proposed method we consider three metrics which allow the evaluation at different levels:

- *Score-level error rate* (S_e): ratio of scores that are not correctly recognized in their entirely (i.e., contain at least one error amongst the estimated ones).

- *Edit distance* (E_d): average number of edit operations to convert the predicted sequence into the ground-truth one.

- *Normalized edit distance* (E_{d_N}): same as the *Edit distance* metric but normalizing each sequence by its length.

Note that the relevance of each metric depends on the final scenario. If a totally autonomous system is pursued, it is important to pay attention to the score-level error. However, quite often it is assumed that an expert user will supervise the output of the system because guaranteeing a error-free model is not feasible [4]. In this case, therefore, it is more interesting to measure the errors at the symbol level, which is more related to the number of corrections to be made.

4.3 Results

Input images must be resized to fixed dimensions for the input of the network. As mentioned earlier, an aspect ratio of 1:4 was chosen. Thus, we have experimented with values involving 40×160, 50×200, and 60×240.

For each case, network parameters are optimized by means of the training set, while the validation set is used to find the most appropriate epoch to stop. The metric chosen to determine the performance after each epoch during training is the normalized edit distance (E_{d_N}).

Once a model is trained, predictions are made on the samples of test set. Table 3 shows the results of our series of experiments in terms of the three figures of merit previously described.

An initial remark to begin is that all input sizes behave similarly. In all the cases, a remarkable performance at symbol level is attained, with figures lower than 0.6 and $4\,\%$ for E_d and E_{d_N}, respectively. It is true, however, that

Block	Configuration			
Convolutional	Conv(64,3,3) MaxPool (2,2)	Conv(128,3,3) MaxPool (2,2)	Conv(256,3,3) Conv(256,3,3) MaxPool(2,1)	Conv(512,3,3) Conv(512,3,3) MaxPool(2,1)
Recurrent	BLSTM (256)	BLSTM (256)	BLSTM (256)	FC (52)

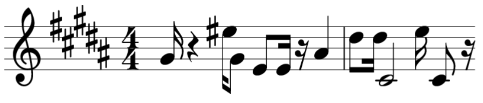
Table 1. Description of the CRNN architecture considered. Notation Conv(f,w,h) stands for a layer with f convolution operators of size $w \times h$ pixels followed by a ReLU activation function. MaxPool(w,h) stands for the max-pooling operator of dimensions $w \times h$ pixels, BLSTM(n) represents a Bidirectional Long-Short Term Memory unit with n hidden layers, and FC(n) is a fully-connected layer of n neurons followed by a SoftMax activation function.

	Training	Validation	Test
Scores	56 991	18 996	18 997
Measures	125 971	41 883	41 986
Symbols	989 744	329 802	330 092

Table 2. Statistics of the partitions used in this work, reporting the number of scores, the number of measures, and the number of running symbols.

	Metric		
Input image	S_e (%)	E_d	E_{d_N} (%)
40 × 160	27.30	0.52	3.01
50 × 200	29.79	0.54	3.12
60 × 240	22.37	0.37	2.16

Table 3. Performance achieved on the test partition with respect to the shape of the input image.


(a) Input score

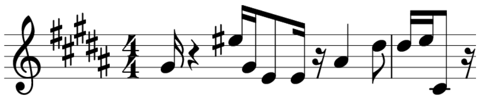

(b) Prediction of the CRNN

Figure 4. An example of prediction with errors ($E_d = 3$, $E_{d_N} = 11.53$) obtained in our experiments.

the score-level error rates are much higher. That is, quite often there is at least one incorrectly recognized symbol in each score sequence.

Best results are obtained using images of 60 × 240. In that case, a symbol-level error rate of 22.37 % is attained, with an average of 0.37 symbol-level errors per score (2.16 % of the symbols if lengths are taken into account). This means that less than one symbol has to be corrected to obtain the actual score, on average. An example of prediction results depicting representative transcription errors is illustrated in Figure 4. Note how some of these error change the arrangement of the beamed groups, as the predicted sequence does not fulfill time signature constraints.

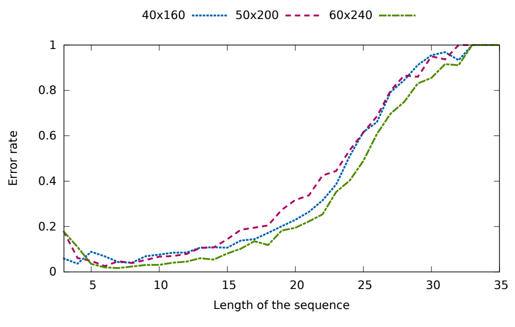
Clearly, these results reflect that the proposed framework allows recognizing accurately the symbols of monophonic scores in an end-to-end manner. In turn, the approach is not so reliable to optimize the number of perfectly recognized images, regardless of the number of errors. However, it has to be considered that some music symbols of the generated scores have vertical overlapping, as can be seen in the first note C from Figure 2. When this happens, the order of the symbolic sequence might not perfectly align with the order of the symbols in the image, thereby introducing noise in the samples.

As discussed in Sect. 1, there are no previous approaches dealing with the OMR task in an end-to-end way and, therefore, there is no feasible comparison in this work. Nevertheless, it is our hope that these results will establish a new way of approaching OMR.

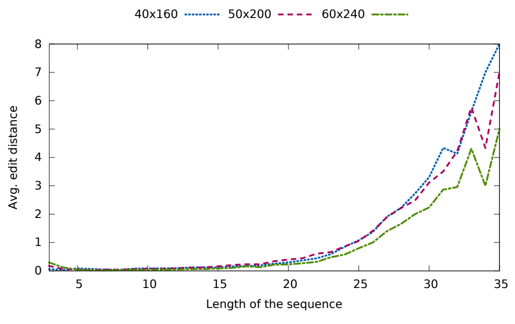
4.4 Further Analysis

In this section we further analyze some details of the experiments carried out.

First we intend to measure the performance of the models with respect to the size of the input sequence. Clearly, the size of the sequence has a direct impact on the ability of the models to recognize all their symbols. It is expected that the greater the number of symbols in the score, the worse performance the models attain. Figure 5 shows the performance curves as a function of the size of the sequences. On the one hand, Figure 5(a) reports the error rate curve, which depicts that the performance gets dramatically worsen from sequences of 10-15 symbols, depending on the model. On the other hand, Figure 5(b) shows the curve of the edit distance, for which it is observed that the average number of editing operations to correct a sequence predicted by the model is less than 1 up to 25 symbols. The interesting remark about these curves is that they allow us to conclude that in relatively short sequences, the models can obtain an almost optimal performance. Fortunately, the scores can be further subdivided into bars, which have a limited number of symbols. Therefore, it might be interesting to address the problem by first performing a segmentation of measures, for which there already exists successful algorithms [22] as previously mentioned.



(a) Score-level error rate (S_e)



(b) Edit distance (E_d)

Figure 5. Performance attained by the models with respect to the length of the input sequences.

Finally, it is interesting to analyze the convergence for each considered model, since that is an indicator of the representation capacity and the difficulty with which they learn the task. Figure 6 shows the normalized edit distance on the validation set as a function of the number of training epochs. It is observed that all models follow a similar trend, in which there is a drastic decrease in the first 6 epochs. After that, models begin to need more epochs to improve their results, reaching convergence (except for minor fluctuations) around 12 epochs. We can therefore say that all models have a similar representation capabilities, although it has been demonstrated in the previous section that the model that accepts 60×240 images has a greater generalization ability. In addition, the low number of required epochs indicate that the models are able to learn the task quickly.

5. CONCLUSIONS

This work addresses the Optical Music Recognition task in an end-to-end fashion with the use of a Convolutional Recurrent Neural Network (CRNN). We have redesigned the architecture from Shi et al. [19] for OMR using a large collection of over 90,000 synthetic scores generated through

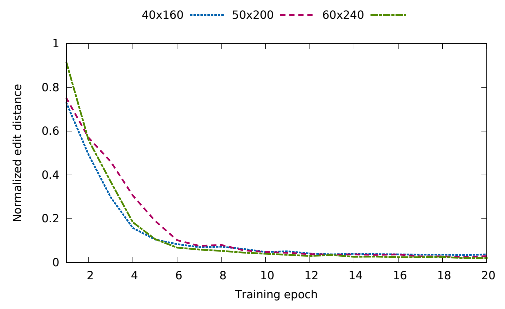


Figure 6. Validation performance (normalized edit distance) with respect to the number of training epochs.

Lilypond, a music engraving system. As the network is trained using a Connectionist Temporal Classification loss function, the music symbols do not need to be aligned with the pixels of the original images.

The CRNN topology and hyper-parameters were experimentally adjusted for the task at hand, obtaining remarkably low error rates with the evaluated corpus. The network converges quickly and an average edit distance of 0.37 is obtained using as input 60×240 images.

In order to increase the accuracy of the proposed method, those scores containing temporal overlappings could be removed from the corpus. However, the ultimate goal of OMR is to detect music symbols in polyphonic scores. This is a challenging task using CRNN as it implies to extend CTC for multi-label classification, which stands as future work to explore and study.

Another evident future work line is to train the network with real scores. Synthetic data could be used as a basis by adding noise and transformations such as rotation or scaling for a preliminary experimentation as in [19], but ideally a large real corpus should be used instead. Currently there are no large datasets containing labeled images of real scores, but an end-to-end annotation of the data is straightforward as it does not requires the symbols to be aligned with the image pixels.

Finally, note that one of the main advantages of the proposed neural-based approach is that alternative notations could be recognized by just changing the corpus and retraining the model. This opens a path for research in research of ancient music recognition written in, for instance, mensural or neume notation, among others.

6. REPRODUCIBILITY

For reproducibility purposes, the source code, trained models, and considered data have been publicly released at `http://grfia.dlsi.ua.es/gen.php?id=software`.

7. ACKNOWLEDGEMENT

This work was partially supported by the Social Sciences and Humanities Research Council of Canada, and Universidad de Alicante through FPU program (UAFPU2014–5883) and grant GRE-16-04.

8. REFERENCES

[1] D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

[2] V. Bosch, J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal. Sheet music statistical layout analysis. *International Conference on Frontiers in Handwriting Recognition 2016*, 2016.

[3] J. A. Burgoyne, Y. Ouyang, T. Himmelman, J. Devaney, L. Pugin, 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*, pages 723–727, 2009.

[4] J. Calvo-Zaragoza and J. Oncina. An efficient approach for interactive sequential pattern recognition. *Pattern Recognition*, 64:295–304, 2017.

[5] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal. Early handwritten music recognition with hidden markov models. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 319–324, 2016.

[6] V. B. Campos, J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal-Ruiz. Sheet music statistical layout analysis. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 313–318, 2016.

[7] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–766, 2008.

[8] T. Géraud. A morphological method for music score staff removal. In *Proceedings of the 21st International Conference on Image Processing (ICIP)*, pages 2599–2603, Paris, France, 2014.

[9] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with lstm recurrent networks. *J. Mach. Learn. Res.*, 3:115–143, March 2003.

[10] 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 '06, pages 369–376, New York, NY, USA, 2006. ACM.

[11] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

[12] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[14] T. Pinto, A. Rebelo, G. A. Giraldi, and J. S. Cardoso. Music score binarization based on domain knowledge. In *Pattern Recognition and Image Analysis - 5th Iberian Conference, IbPRIA 2011, Las Palmas de Gran Canaria, Spain, June 8-10, 2011. Proceedings*, pages 700–708, 2011.

[15] L. Pugin. Optical music recognitoin of early typographic prints using hidden markov models. In *ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006, Proceedings*, pages 53–56, 2006.

[16] C. Ramirez and J. Ohya. Automatic recognition of square notation symbols in western plainchant manuscripts. *Journal of New Music Research*, 43(4):390–399, 2014.

[17] A. Rebelo, G. Capela, and J. S. Cardoso. Optical recognition of music symbols - A comparative study. *IJDAR*, 13(1):19–31, 2010.

[18] 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*, 1(3):173–190, 2012.

[19] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *Computing Research Repository*, abs/1507.05717, 2015.

[20] S. Sigtia, E. Benetos, and S. Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.

[21] A. H. Toselli, V. Romero, M. Pastor, and E. Vidal. Multimodal interactive transcription of text images. *Pattern Recognition*, 43(5):1814–1825, 2010.

[22] G. Vigliensoni, G. Burlet, and I. Fujinaga. Optical measure recognition in common music notation. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013*, pages 125–130, 2013.

[23] P. Voigtlaender, P. Doetsch, and H. Ney. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, pages 228–233, 2016.

EXPLOITING PLAYLISTS FOR REPRESENTATION OF SONGS AND WORDS FOR TEXT-BASED MUSIC RETRIEVAL

Chia-Hao Chung
National Taiwan University
b99505003@ntu.edu.tw

Yian Chen
KKBOX Inc.
annchen@kkbox.com

Homer Chen
National Taiwan University
homer@ntu.edu.tw

ABSTRACT

As a result of the growth of online music streaming services, a large number of playlists have been created by users and service providers. The title of each playlist provides useful information, such as the theme and listening context, of the songs in the playlist. In this paper, we investigate how to exploit the words extracted from playlist titles for text-based music retrieval. The main idea is to represent songs and words in a common latent space so that the music retrieval is converted to the problem of selecting songs that are the nearest neighbors of the query word in the latent space. Specifically, an unsupervised learning method is proposed to generate a latent representation of songs and words, where the learning objects are the co-occurring songs and words in playlist titles. Five metrics (precision, recall, coherence, diversity, and popularity) are considered for performance evaluation of the proposed method. Qualitative results demonstrate that our method is able to capture the semantic meaning of songs and words, owning to the proximity property of related songs and words in the latent space.

1. INTRODUCTION

Online music streaming services, such as Spotify, Apple Music, and KKBOX, create various playlists for the convenience of music listening for users. Meanwhile, users may create their own playlists for replay or music sharing with friends [1–5]. The use of playlist makes music retrieval and organization simple and easy, largely because the title of a playlist carries significant thematic information of the songs contained in the playlist [1–3]. The theme can be about an artist, genre, mood, or context of the playlist. Therefore, the thematic information is useful for music retrieval. The goal of this paper is to exploit playlists for text-based music retrieval.

One critical issue of text-based music retrieval is how to identify and quantify the relationship between words and songs (i.e. which songs and words are relevant to each other and how much is the relevance). Most previous approaches to text-based music retrieval rely on human-labeled datasets [6, 7] or social tags [8, 9] which normally have a limited size of vocabulary (word set). The web-based approach [10, 11] has been considered a good alternative because web documents have rich text information. However, its performance may degrade in the presence of noisy text [12]. In contrast, the playlist-based approach has the following appealing features: 1) The rich text information conveyed by the succinct playlist title is highly relevant to the songs in the playlist and 2) Songs wrapped in one playlist must be related to each other in a certain way. If the relationship can be determined from the playlist, additional efforts on audio signal analysis [6, 7, 11] can be saved.

Our main idea is to represent songs and words in a common latent space so that music retrieval can be converted to the problem of selecting songs sufficiently near the query word in the latent space. Specifically, we propose an unsupervised learning method to generate a representation of songs and words extracted from playlist titles, in which the learning function is optimized based on the co-occurrence of songs and words in playlists. As each song or word (an object) is represented as a vector in a latent space, the semantic similarity between two objects can be easily determined by the distance between the two corresponding vectors. By exploiting this property, we can improve the performance of text-based music retrieval.

Our contributions can be summarized as follows:

- We propose an unsupervised learning method to model the relevance between songs and words of playlists and to represent these two kinds of objects in a common latent space.
- We make text-based music retrieval easier to solve by formulating it as a nearest neighbor search problem in the latent space.
- Both qualitative and quantitative evaluations are conducted to demonstrate the effectiveness of the proposed method.

2. RELATED WORK

In this section, we review previous work related to playlist understanding, text-based music retrieval, and representation learning.

2.1 Playlist Understanding

To understand the use of playlist, Hagen [1] and Cunningham et al. [2] conducted user interviews to analyze various themes and contexts of playlists. The results motivated Pichl et al. [3] to mine common listening contexts using playlist titles for context-aware

Playlist Title	Words	Songs (Artists)
Summer's Over	summer	The Boys of Summer (The Ataris) So Long, So Long (Dashboard Confessional) Last Days of Summer (Silverstein) Close To Home (The Get Up Kids) Always Summer (Yellowcard) …
Happy Morning Chill	happy morning chill	Snap Out Of It (Arctic Monkeys) Unbelievers (Vampire Weekend) Demons (Imagine Dragons) The Mother We Share (Chvrches) Everybody Wants To Rule The World (Lorde) …
George Michael - For the Heart	george_michael heart	Don't Let the Sun Go Down on Me (George Michael) Careless Whisper (George Michael) Heal The Pain (George Michael) A Different Corner (George Michael) I Can't Make You Love Me (George Michael) …

Table 1. Illustration of words extracted from playlists. Only the first five songs of a playlist are shown.

music recommendation. In our work, we take a step further and investigate how to exploit the words extracted from playlist titles for text-based music retrieval.

A related issue is playlist quality measurement. Motivated by the observation that the songs in a playlist, although diverse, are related to each other in a certain way, Fields [4] introduced coherence and diversity as metrics of playlist quality. It was found that popularity and freshness of songs in a playlist are also important metrics [5]. Considering that the response to a text query is in the form of playlist, we apply coherence, diversity, and popularity as metrics for performance evaluation.

2.2 Text-Based Music Retrieval

To allow the retrieval of music pieces by text query, the relevance between words and songs has to be identified. Turnbull et al. [6] and Chechik et al. [7] developed a multi-class classification approach to predict the relevance of a music piece to a query. To address the issue that the perception of relevance is subjective, Hariri et al. [8] and Cheng et al. [9] used a probabilistic model and listening records to personalize text-based music retrieval. To extend the coverage of text queries, Knees et al. [10–12] crawled web documents relevant to a music piece and represented the music piece by the text extracted from the web documents. However, most of the body of words contained in web documents can be irrelevant to the theme of the music pieces. To solve the problem, we develop an alternative approach that seeks relevant words from the playlist titles.

2.3 Representation Learning

Representation learning has been widely applied to music recommendation [13, 16, 29], playlist recommendation [17], music annotation and retrieval [18], playlist generation [19, 20], and listening behavior analysis [21, 22]. The popularity of representation learning is due to its two appealing features. First, it can efficiently handle large scale dataset [23, 24] because of low model complexity. Second, it makes information retrieval or recommendation an easy task that can be efficiently accomplished. However, little attention has been paid to exploit representation learning for text-based music retrieval. In this paper, we extend the idea of embedding learning [16–24], which is a typical representation learning approach, to model the relevance between songs and words of playlists.

3. PROPOSED METHOD

We first introduce the notations used in this paper. Then, we describe the proposed method for learning a representation of songs and words and the detail of the training processing, including optimization and data sampling. Finally, we describe how the learned representation is applied to text-based music retrieval.

3.1 Notations

Let $L = \{l_1, l_2, \dots, l_\mathrm{I}\}$ be a set of playlists and $T = \{t_1, t_2, \dots, t_\mathrm{I}\}$ be the set of corresponding playlist titles. Each playlist l_i, $1 \le i \le \mathrm{I}$, as illustrated in Table 1, is associated with a set of songs $S^i = \{s_1^i, s_2^i, \dots, s_{|l_i|}^i\}$ and a set of words $W^i = \{w_1^i, w_2^i, \dots, w_{|t_i|}^i\}$ extracted from t_i. Let $S = \{s_1, s_2, \dots, s_\mathrm{N}\}$ be the union of all S^i, and $W = \{w_1, w_2, \dots, w_\mathrm{M}\}$ be the union of all W^i. The goal is to learn a representation $\theta(\cdot)$ to map each $s_n \in S$ or $w_m \in W$ to a vector.

3.2 Representation Learning for Songs and Words

We extend the idea of embedding learning to songs and words. In its basic form, the embedding learning generates a representation for a set of objects based on the co-occurrence of the objects [23]. It consists of two stages. In the first (or initialization) stage, the representation $\upsilon(\cdot)$ assigns a vector of random values to each object. In second (or update) stage, the vector is progressively updated in two steps. In the first step, a conditional probability $P\big(o_c \big| \upsilon(o)\big)$ for each pair of objects o and o_c is created, where o_c is the co-occurring

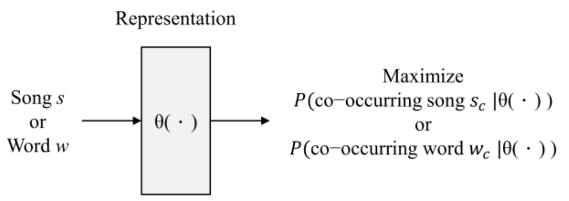

Representation
Song s or Word w
θ(·)
Maximize
P(co−occurring song s_c |θ(·))
or
P(co−occurring word w_c |θ(·))

Figure 1. Representation learning for songs and words. Given a song or a word, the representation is optimized based on its co-occurring song or word.

object of o. In the second step, $\upsilon(o)$ is optimized by maximizing the conditional probability. The two steps are repeated until the maximization for every pair of o and o_c is completed.

To extend the basic idea of embedding learning to songs and words, we need to define the co-occurring relationship between songs and words. If two songs (words) belong to the same playlist (playlist title), we say that they are co-occurring in the playlist (playlist title). Likewise, any word of a playlist title and any song in the playlist have a co-occurring relationship. In our method, co-occurring pairs of songs, words, or song and word are considered positive pairs.

Once the songs, words, and positive pairs of all playlists are in place, a learning process that consists of two stages is applied. In the first stage, the representation $\theta(\cdot)$ for each song or word is randomly initialized. In the second stage, $\theta(\cdot)$ is optimized. Specifically, we go through every positive pair and randomly select a word (or song), denoted as s (or w) from it. Then, we optimize the representation $\theta(s)$ (or $\theta(w)$) in two steps. In the first step, we construct a conditional probability which can be expressed in one of the following four formats:

$$P\big(s_c\big|\theta(s)\big), P\big(w_c\big|\theta(s)\big), P\big(w_c\big|\theta(w)\big), \text{ or } P\big(s_c\big|\theta(w)\big),$$

where s_c or w_c is the remainder song or word in the positive pair. In the second step, we optimize the representation $\theta(s)$ (or $\theta(w)$) by maximizing the conditional probability. The two steps, as illustrated in Figure 1, are repeated until the maximization for every positive pair is completed (e.g. an epoch is completed).

We formulate the entire learning process by the following object function:

$$\mathcal{L} = \sum_{l_i \in L}\Big(\sum_{s \in S^i}\Big(\sum_{s_c \in S^i} \log P\big(s_c\big|\theta(s)\big) + \sum_{w_c \in W^i} \log P\big(w_c\big|\theta(s)\big)\Big) + \sum_{w \in W^i}\Big(\sum_{s_c \in S^i} \log P\big(s_c\big|\theta(w)\big) + \sum_{w_c \in W^i} \log P\big(w_c\big|\theta(w)\big)\Big)\Big). \quad (1)$$

Note that the natural logarithm converts a conditional probability to a log likelihood for the convenience of update stage [24]. The conditional probability $P\big(s_c\big|\theta(w)\big)$ is modeled by a softmax function [23] and can be rewritten as:

$$P\big(s_c\big|\theta(w)\big) = \frac{\exp(\varphi(s_c)\cdot\theta(w))}{\sum_{s'_c \in S} \exp(\varphi(s'_c)\cdot\theta(w))}, \quad (2)$$

where $\varphi(\cdot)$ maps s_c into a vector space. Likewise, $P\big(w_c\big|\theta(w)\big)$, $P\big(s_c\big|\theta(s)\big)$, and $P\big(w_c\big|\theta(s)\big)$ are modeled in the same way. Finally, $\theta(\cdot)$ and $\varphi(\cdot)$ is optimized by maximizing Equation (1).

3.3 Training

There are $2 \times (\mathrm{N} + \mathrm{M}) \times \mathrm{D}$ parameters, including $\theta(\cdot)$ and $\varphi(\cdot)$, to be optimized, where N is the number of songs, M is the number of words, and D is the dimension of the representation. The parameters are optimized by maximizing Equation (1) using the Adam algorithm [25]. However, the computation cost of the optimization is proportional to N and M because of the normalization term in the softmax function. As an alternative, we adopt the negative sampling approach [24] to reduce the computational cost, where 30 negative pairs are randomly sampled for each positive pair.

In our experiments, the dimension of the representation was set to 32, and the hyper-parameters of the Adam algorithm were $\alpha = 0.025$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e^{-08}$. The training was repeated for five epochs.

3.4 Text-based Music Retrieval

The response to a text query $q \in W$ is the songs that are the nearest neighbors of q in the latent space. Specifically, the cosine similarity between each $s_n \in S$ and q is calculated:

$$cosine\ simiarity = \frac{\theta(s_n)\cdot\theta(q)}{\|\theta(s_n)\|_2\|\theta(q)\|_2}, \quad (3)$$

where $\|\cdot\|_2$ denotes the Euclidean norm of a vector. The songs having high cosine similarity are the response to q.

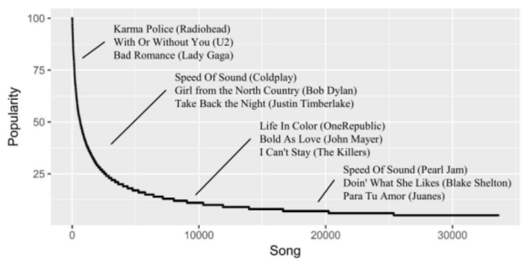
4. EXPERIMENTS

In this section, we describe the experiments conducted to evaluate the performance of the proposed method against matrix factorization, which is another typical approach to representation learning. We first describe the dataset used in the experiments and the pre-processing step applied to the dataset. Then, we describe the implementation details of matrix factorization. Finally, we describe the results of performance evaluation.

4.1 Dataset and Pre-processing

The dataset was collected by Pichl et al. [3] using Spotify API[1]. It contains 21,485 playlists created by 1,500 users, and each playlist contains a title and a list of songs. Standard natural language processing techniques were applied to process the playlist titles. First, all characters in playlist titles were converted to lowercase, and punctuations and stop words, such as “the”, “of”, and “a”, were removed. Then, each playlist title was segmented into a set of words using the NLTK toolkit[2], and single

[1] https://developer.spotify.com/web-api/
[2] http://www.nltk.org/



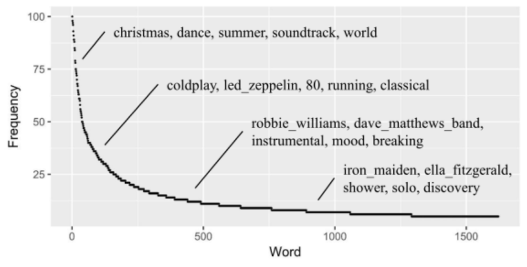


Figure 2. Song popularity and word frequency.

characters or digits were removed from the resulting word set. As many playlists titles contain artist names, the name entity recognition implemented in the NLTK toolkit was applied to identify artist names. Such artist names were considered one entity. The space in an artist name was replaced with the symbol "_" for the convenience of data processing. As shown in Table 1, the words extracted from the playlist title "George Michael - For the Heart" are "george_michael" and "heart".

However, like in other popularity studies [26], we found a serious long tail phenomenon: Few songs and words appear frequently while many others appear rarely. Both kinds of songs and words may affect representation learning. Therefore, we removed songs and words that appear less than 4 times or more than 100 times in the dataset. Interestingly, similar to stop words, words like "radio", "liked", and "music" are not useful for music retrieval but they were automatically removed because they appear many times in playlist titles. At the end of this filtering processing, 33,625 songs and 1,623 words were left (a playlist was removed if its songs and words were all removed). The statistics of the final dataset is listed in Table 2, and the song popularity (the number of times a song appears in playlists) and word frequency (the number of times a word appears in playlist titles) are shown in Figure 2. Note that the entire dataset was used for representation learning, and the performance of the representation for music retrieval was evaluated.

4.2 Matrix Factorization

Matrix factorization (MF) [14, 15] is compared with the proposed method. In MF, the vector $\boldsymbol{x}_w$ for word w and the vector $\boldsymbol{y}_s$ for song s are learned by solving the optimization problem

$$\min_{q_*,p_*} \sum_{w,s}(c_{ws} - \boldsymbol{x}_w^T\boldsymbol{y}_s)^2 + \lambda(\|\boldsymbol{x}_w\|^2 + \|\boldsymbol{y}_s\|^2), \quad (4)$$

where c_{ws} is the number of times w and s co-occur in the playlists, and λ is a regularization parameter to avoid overfitting. The inner product of a query vector and each song vector is calculated to determine which music piece to retrieve. A song with a higher inner product value is considered a better response to the query.

Number of playlists	18,417
Number of songs	33,625
Number of words	1,623
Average number of songs per playlist	20.37
Average number of words per playlist	1.10

Table 2. Data statistics.

We adopted the implementation by MyMediaLite[3]. The dimension of the vectors learned by MF was set to 32, and λ was set to 0.015.

4.3 Performance Evaluation

We measure the quality of the response to a text query by the following five metrics:

Precision and recall: We use these two standard performance evaluation metrics to measure the relevance of a response to a query as follows:

$$precision = \frac{|S_r \cap S_t|}{|S_r|}, \quad (5)$$

$$recall = \frac{|S_r \cap S_t|}{|S_t|}, \quad (6)$$

where S_r is the set of retrieved songs (the songs in the response) and S_t is the set of relevant songs (the songs in the playlists that have the query in the titles). A high precision means that most of the retrieved songs are relevant, and a high recall means that most relevant songs are retrieved.

Coherence: This metric measures the coherence of the songs in the response to a query. Specifically, we obtain social tags of songs from Allmusic[4] and calculate pointwise mutual information (PMI) for every pair of the songs in a response. The coherence is defined as the average of the PMIs,

$$coherence = \frac{1}{\mathrm{L}} \sum_{i<j} \log \frac{P(s_i,s_j)}{P(s_i)P(s_j)}, \quad (7)$$

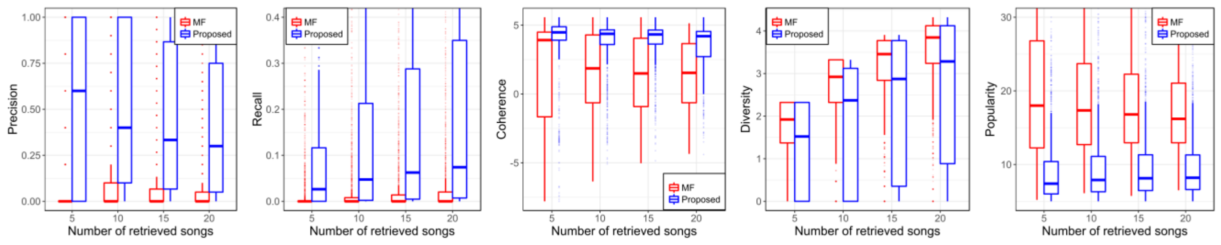
where L is the number of the song pairs, $P(s)$ denotes the probability of s having tags and $P(s_i,s_j)$ denotes the probability of s_i and s_j having the same tags. The coherence would be high if the songs in the response have the same social tags.

Diversity: This metric measures how diverse the songs in a response are [4, 27]. The diversity is defined as the cross entropy of artists appearing in the response:

$$diversity = \sum_{a \in A} P(a) \log(P(a)), \quad (8)$$

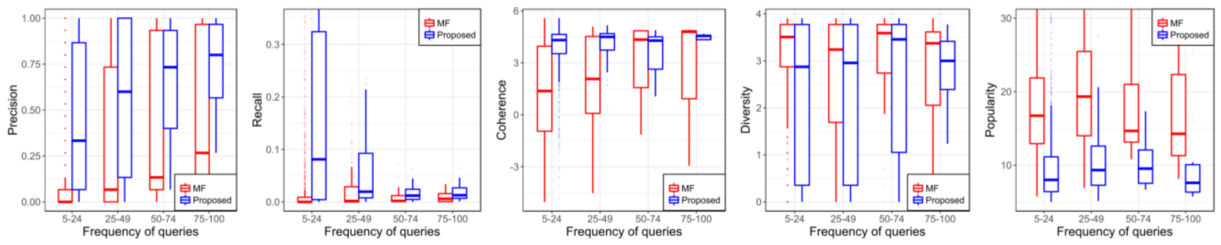
where A represents the set of artists in the response, and $P(a)$ denotes the probability of artist a appearing in the response. The diversity would be high if various artists appear in the response.

[3] http://www.mymedialite.net/
[4] http://www.allmusic.com/discover



(a) (b) (c) (d) (e)

Figure 3. Performance comparison of the proposed method and MF. The results are shown as box plots [28], where the bottom and top of a box are the first and third quartiles, and the band inside the box is the second quartile (the median). Please refer to [28] for the details of box plot.



(a) (b) (c) (d) (e)

Figure 4. Performance comparison of the proposed method and MF for queries with different word frequency. The number of retrieved songs is set to 15.

Popularity: We calculate the average popularity of the songs in a response [27],

$$popularity = \frac{1}{\mathrm{K}}\sum_{k\leq \mathrm{K}} PS_k, \quad (9)$$

where K is the number of songs in the response, and PS_k represents how many times a song appears in the dataset. A low-popularity response is desired for a music retrieval and recommendation system because users may discover songs they have never heard before.

The response quality of the proposed method is compared with that of MF by the five metrics. Every word in the vocabulary (W) is considered a query to retrieve k relevant songs using a method. The average results are shown in Figure 3. We can see from Figures 3(a) and 3(b) that the precision and recall of the proposed method are higher than those of MF. It shows the effectiveness of the proposed method for preserving the relevance between songs and words. In Figure 3(c), we can see that the proposed method outperforms MF in terms of coherence. As more songs are retrieved, our method provides a stable coherence while MF has a descending coherence. In Figure 3(d), we can see that our method has a large variation in terms of diversity. It is because many responses of our method contain only the songs of one artist (zero diversity) if the query is an artist name. In Figure 3(e), we can see that the responses of the proposed method tend to have lower popularity than the responses of MF. It implies that MF favors popular songs.

We compare the responses of queries with different word frequencies. As shown in Figure 4, queries are divided into four groups according to the word frequency. We can see from Figures 4(a) and 4(b) that the proposed method has higher precision and recall than MF for each group. We can also see from Figure 4(c) that our method provides a stable coherence regardless of the frequency of query words while MF favors queries with high word frequency. In Figure 4(d), it is interesting to see that our method yields a low diversity for some queries with low word frequency. It is because part of the words with low frequency are artist names. Figure 4(e) shows that the proposed method provides responses with low popularity regardless of the word frequency of queries.

4.4 Qualitative Study

We show the responses of the two methods under comparison to five queries (“christmas”, “punk”, “60s”, “coldplay”, and “miles_davis”) in Table 3. The five queries are selected manually to cover various semantic meanings and word frequencies. Additional results and visualization of the learned latent space are provided on our website[5].

The first query “christmas” has a high word frequency, which means this word is frequently used in playlist titles. We can see that both the proposed method and MF can find songs relevant to Christmas. However, we note that the response of MF contains only two artists (actually, four of the five songs in Table 3 belong to the same artist) and has a high popularity. In contract, our method can find songs with high diversity and low popularity. The second query “punk” has a lower word frequency than the first query. We can see that the proposed method still provides a good response, while the response of MF is not very relevant to “punk”. It implies that MF may fail when the query has a low word frequency.

[5] http://mpac.ee.ntu.edu.tw/chiahaochung/textMR.php

Query	Matrix factorization	Proposed method
christmas (98[a])	It's Beginning To Look A Lot Like Christmas (Michael Bublé[b], 38[c]) All I Want For Christmas Is You (Mariah Carey, 40) White Christmas (Michael Bublé, 23) Santa Claus Is Coming To Town (Michael Bublé, 15) All I Want For Christmas Is You (Michael Bublé, 25)	Queen Of The Winter Night (Trans-Siberian Orchestra, 5) O Come All Ye Faithful/ O Holy Night (Trans-Siberian Orchestra, 6) Rudolph The Red Nosed Reindeer (Burl Ives, 6) Rockin' Around The Christmas Tree (She & Him, 5) Christmas Is Going To The Dogs (Eels, 6)
punk (23)	Sing (Ed Sheeran, 68) Shirtsleeves (Ed Sheeran, 17) Don't Let It Go (Beck, 30) Somewhereinamerica (JAY Z, 24) Bloodstream (Ed Sheeran, 29)	I Want To Conquer The World (Bad Religion, 6) Story of My Life (Social Distortion, 19) Monosyllabic Girl (NOFX, 6) Generator (Bad Religion, 10) Leave It Alone (NOFX, 8)
60s (13)	Together (Calvin Harris, 8) The Card Cheat (The Clash, 6) Bowery (Local Natives, 14) You Make Loving Fun (Fleetwood Mac, 34) Second Hand News - Early Take (Fleetwood Mac, 6)	Take Good Care Of My Baby (Bobby Vee, 5) Silence Is Golden (The Tremeloes, 5) Wooly Bully (Sam The Sham & The Pharaohs, 8) Daydream (The Lovin' Spoonful, 11) Blue Velvet (Bobby Vinton, 10)
coldplay (40)	Charlie Brown (Coldplay, 51) Major Minus (Coldplay, 17) Mylo Xyloto (Coldplay, 19) Hurts Like Heaven (Coldplay, 36) Every Teardrop Is a Waterfall (Coldplay, 42)	U.F.O. (Coldplay, 19) Prospekt's March/Poppyfields (Coldplay, 12) White Shadows (Coldplay, 15) Mylo Xyloto - Live (Coldplay, 7) Twisted Logic (Coldplay, 9)
miles_davis (7)	Scarborough Fair / Canticle (Simon & Garfunkel, 14) Is She Weird (Pixies, 6) Shoes Upon the Table (Blood Brothers - 1995 London Cast, 5) I Would For You (Nine Inch Nails, 13) Love Is The Answer (Aloe Blacc, 13)	Fran-Dance (Miles Davis, 7) On Green Dolphin Street (Miles Davis, 7) Spanish Key (Miles Davis, 5) Flamenco Sketches (Miles Davis, 13) Love For Sale (Miles Davis, 8)

Table 3. Qualitative Study. Only the top five songs to a query are shown. ([a] word frequency, [b] artist, [c] song popularity)

The query “60s” is interesting, as it is related to the songs or artists in 1960s. We can see that our method can find the songs of artists who were popular in 1960s, including Bobby Vee, The Tremeloes, Sam The Sham & The Pharaohs, The Lovin' Spoonful, and Bobby Vinton. MF fails in this case because “60s” has a low frequency.

The last two queries “coldplay” and “miles_davis” are both artists, where the former has higher word frequency than the latter. We can see that our method provides good responses to the two queries, while MF fails in the case of “miles_davis”. It can be expected that the response to this kind of query should contain only the songs of the artist specified in the query. Note that there are many artist names in our vocabulary, and most of them have low word frequency. Because the proposed method works for these artist queries as well as other queries, the diversity of the proposed method has a large variation, as shown Figures 3(d) and 4(d).

5. DISCUSSION

We first discuss the difference between the proposed method and MF in terms of the learning function. As described in Equation (4), MF considers only the co-occurrence of song-word pairs. In contrast, our method exploits three types of co-occurrence between songs and words of playlists. Although an improved MF [29] can be applied to factorize multiple co-occurrence matrices, we believe that the property of MF (i.e. the favor of popular songs and words) would make MF unsuitable for text-based music retrieval.

Our method is related to the embedding method proposed by Moore et al. [20] for representation learning of songs and tags for playlist prediction. The difference between our method and their method lies in the function used to model the conditional probability: a softmax function vs a logistic function that uses the Euclidean distance between two vectors as input. Besides, we applied two modern approaches, the Adam algorithm [25] and the negative sampling [24], to improve the efficiency of representation learning.

Finally, we discuss two possible directions to extend the proposed method. One direction is to enlarge song set and word set. As song titles and lyrics also contain rich text information, they can be incorporated to expand word set. Besides, the approach proposed by Oord et al. [30] can be applied to map new songs into the latent space learned by our method. This approach also solves the cold start problem [27]. The other direction is to develop a music retrieval system which allows multiple words as a query, because people may use multiple words or even a sentence to retrieve music. There are simple solutions, for example, combining the responses to multiple single-word queries [6]. However, such combination may not truly capture the semantic meaning of a multiple-words query. To deal with such query, a better solution, such as the approach proposed by Mikolov et al. [24], can be incorporated into the proposed method. We can see the potential and high extendability of our method.

6. CONCLUSION

In this paper, we have proposed an unsupervised learning method to generate the latent representation of songs and words of playlists for text-based music retrieval. Such representation captures the relevance between songs and words, owning to the proximity property of the latent space. Both qualitative and quantitative evaluations show the effectiveness of the proposed method compared against the matrix factorization method for text-based music retrieval.

7. REFERENCES

[1] A. N. Hagen, "The playlist experience: Personal playlists in music streaming services," *Popular Music and Society*, vol. 38, no. 5, pp. 625–645, 2015.

[2] S. J. Cunningham, D. Bainbridge, and A. Falconer, "More of an art than a science: Supporting the creation of playlists and mixes," in *Proc. 7th Int. Conf. Music Inf. Retrieval (ISMIR)*, pp. 240–245, 2006.

[3] M. Pichl, E. Zangerle, and G. Specht, "Towards a context-aware music recommendation approach: What is hidden in the playlist name?" in *Proc. 15th IEEE Int. Conf. Data Mining Workshop (ICDMW)*, pp. 1360–1365, 2015.

[4] B. Fields, "Contextualize your listening: The playlist as recommendation engine," PhD dissertation, Dept. Comput., Goldsmiths, Univ. London, 2011.

[5] D. Jannach, I. Kamehkhosh, and G. Bonnin, "Analyzing the characteristics of shared playlists for music recommendation," in *Proc. 6th Workshop Recommender Syst. Social Web (RSWeb)*, 2014.

[6] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Towards musical query-by-semantic-description using the CAL500 data set," in *Proc. 30th Int. ACM Conf. Res. Develop. Inf. Retrieval (SIGIR)*, pp. 23–27, 2007.

[7] G. Chechik, E. Ie, M. Rehn, S. Bengio, and D. Lyon, "Large-scale content-based audio retrieval from text queries," in *Proc. 1st ACM Int. Conf. Multimedia Inf. Retrieval*, pp. 105–112, 2008.

[8] N. Hariri, B. Mobasher, and R. Burke, "Personalized text-based music retrieval," in *Workshops 27th AAAI Conf. Artificial Intell.*, 2013.

[9] Z. Cheng, J. Shen, and S. C.H. Hoi, "On effective personalized music retrieval by exploring online user behaviors," in *Proc. 39th Int. ACM Conf. Res. Develop. Inf. Retrieval (SIGIR)*, pp. 125–134, 2016.

[10] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner, "A document-centered approach to a natural language music search engine," in *European Conf. Inf. Retrieval (ECIR)*, Springer Berlin Heidelberg, pp. 627–631, 2008.

[11] P. Knees, T. Pohle, M. Schedl, D. Schnitzer, K. Seyerlehner, and G. Widmer, "Augmenting text-based music retrieval with audio similarity," in *Proc. 10th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, pp. 579–584, 2009.

[12] P. Knees, M. Schedl, T. Pohle, K. Seyerlehner, and G. Widmer, "Supervised and unsupervised web document filtering techniques to improve text-based music retrieval," in *Proc. 11th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, pp. 543–548, 2010.

[13] G. Dror, N. Koenigstein, and Y. Koren, "Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy," in *Proc. 5th ACM Int. Conf. Recommender Syst. (RecSys)*, pp. 165–172, 2011.

[14] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[15] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, pp. 263–272, 2008.

[16] C.-M. Chen, M.-F. Tsai, Y.-C. Lin, and Y.-H. Yang, "Query-based music recommendations via preference embedding," in *Proc. 10th ACM Conf. Recommender Syst. (RecSys)*, pp. 79–82, 2016.

[17] C.-M. Chen, C.-Y. Yang, C.-C. Hsia, Y. Chen, and M.-F. Tsai, "Music playlist recommendation via preference embedding," in *Poster Proc. 10th ACM Conf. Recommender Syst. (RecSys)*, 2016.

[18] J. Weston, S. Bengio, and P. Hamel, "Large-scale music annotation and retrieval: Learning to rank in joint semantic spaces," *arXiv preprint arXiv: 1105.5196*, 2011.

[19] S. Chen, J. L. Moore, D. Turnbull, and T. Joachims, "Playlist prediction via metric embedding," in *Proc. 18th ACM Int. Conf. Knowledge Discovery Data Mining (SIGKDD)*, pp. 714–722, 2012.

[20] J. L. Moore, S. Chen, T. Joachims, and D. Turnbull, "Learning to embed songs and tags for playlist prediction," in *Proc. 13th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, pp. 349–354, 2012.

[21] J. L. Moore, S. Chen, T. Joachims, and D. Turnbull, "Taste over time: The temporal dynamics of user preferences," in *Proc. 14th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, pp. 401–406, 2013.

[22] C.-H. Chung, J.-K. Lou, and H. Chen, "A latent representation of users, sessions, and songs for listening behavior analysis," in *Proc. 17th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, 2016.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv: 1301.3781*, 2013.

[24] T. Mikolov, I. Sutskever, K. Chen,G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Advances in Neural Inf. Process. Syst. (NIPS)*, pp. 3111–3119, 2013.

[25] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv: 1412.6980*, 2014.

[26] O. Celma, *Music Recommendation and Discovery*, Springer Berlin Heidelberg, 2010.

[27] S.-Y. Chou, Y.-H. Yang, and Y.C. Lin, "Evaluating music recommendation in a real-world setting: On data splitting and evaluation metrics," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, pp. 1–6, 2015.

[28] R. McGill, J. W. Tukey and W. A Larsen, "Variations of box plots," *American Statistician*, vol. 32, no. 1, pp.12–16, 1978.

[29] A. Vall, M. Skowron, P. Knees, and M. Schedl, "Improving music recommendations with a weighted factorization of the tagging activity," in *Proc. 16th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, pp. 65–71, 2015.

[30] A. Van de Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Advances Neural Inform. Process. Syst. (NIPS)*, pp. 2643–2651, 2013.

FREESOUND DATASETS: A PLATFORM FOR THE CREATION OF OPEN AUDIO DATASETS

Eduardo Fonseca*, **Jordi Pons***, **Xavier Favory***, **Frederic Font, Dmitry Bogdanov, Andres Ferraro, Sergio Oramas, Alastair Porter, and Xavier Serra**
Music Technology Group, Universitat Pompeu Fabra, Barcelona
name.surname@upf.edu

ABSTRACT

Openly available datasets are a key factor in the advancement of data-driven research approaches, including many of the ones used in sound and music computing. In the last few years, quite a number of new audio datasets have been made available but there are still major shortcomings in many of them to have a significant research impact. Among the common shortcomings are the lack of transparency in their creation and the difficulty of making them completely open and sharable. They often do not include clear mechanisms to amend errors and many times they are not large enough for current machine learning needs. This paper introduces Freesound Datasets, an online platform for the collaborative creation of open audio datasets based on principles of transparency, openness, dynamic character, and sustainability. As a proof-of-concept, we present an early snapshot of a large-scale audio dataset built using this platform. It consists of audio samples from Freesound organised in a hierarchy based on the AudioSet Ontology. We believe that building and maintaining datasets following the outlined principles and using open tools and collaborative approaches like the ones presented here will have a significant impact in our research community.

1. INTRODUCTION

Machine learning has a prominent role in data-driven approaches nowadays for many research fields, including sound and music computing. Because of this, having well curated datasets is essential for allowing solid research outcomes. The ImageNet dataset powered most recent advances in computer vision research [5, 27]. This was possible because ImageNet is a large-scale, openly available dataset with a solid ground truth. Despite quite a few datasets being available in the sound and music computing field, there are still major shortcomings in many of them. What follows is a list of some of the most relevant datasets from the sound and music computing community along with some observations. We put a special focus on music and environmental sounds datasets. Table 1 shows some general statistics about them.

• **GTZAN** [39]. Its online availability [1] has enabled easy benchmarking of music genre recognition tasks in the music information retrieval (MIR) field [35]. Despite its popularity, this dataset has been criticized for its small size, its faults and its commonly used data partitions [36]. However, its faults (repetitions, mislabelings, and distortions) were not identified until 2012, ten years after its release. In addition, commonly used data partitions were found to provide an over-optimistic view of the state-of-the-art. Recent work shows that performance is much worse when using a "fault-filtered" GTZAN partition [13].

• **Ballroom** [10]. This music dataset contains beat, tempo and genre annotations [10, 14] and can therefore be used for more than one task. It has also been highly criticized for its small size, its repeated songs (thirteen replicas were found [2]), and the strong relationship between tempo and genre labels (even though the dataset was designed to assess rhythmic descriptors) [10]. Recently, an extension was proposed [20] and 4180 audio clips are now available for 13 unbalanced classes.

• The **Million Song Dataset** [2] was released to provide a large-scale dataset for MIR benchmarking. It contains audio features and metadata for a million contemporary popular music tracks, with a bias towards pop/rock songs. [3] Audio features can be linked to resources useful for several MIR disciplines: lyrics, CD artwork, tags, similarity measures, user data, cover songs or genre labels. This makes it perfect for exploring multimodal approaches. However, the audio files are not available, and the provided audio features were extracted with proprietary software which is neither debuggable nor inspectable [31].

• The **MagnaTagATune** [17] dataset includes music data released under Creative Commons (CC) licenses, which simplifies data sharing, and annotations (tags and similarity) were made by engaging users in playing a game. Since gamification was a research goal in itself, the

*Equally contributing authors.

[1] http://marsyasweb.appspot.com/download/data_sets/
[2] http://media.aau.dk/null_space_pursuits/2014/01/ballroom-dataset.html
[3] http://www.ifs.tuwien.ac.at/mir/msd/MAGD.html

annotation tool is well documented [16]. This makes this dataset more transparent than others in terms of its creation process. As main drawbacks, it is highly unbalanced and its annotations are noisy and inconsistent.[4] To alleviate these issues, researchers typically use top-50 tags [25, 6], or a cleaner and pre-processed version (Magtag5k [21]).

• **AudioSet** [9] is, to date, the largest audio dataset available. It is structured with an ontology containing 632 classes, including music and environmental sound concepts. AudioSet provides a web-interface to navigate and listen to examples from the dataset, which helps giving researchers an overview of its contents. However, even though the dataset annotations have been manually validated, ≈15% of the categories present a quality estimate with a score below 50%.[5]

• **TUT Acoustic Scenes** [23] is a publicly available dataset released for the DCASE 2016 challenge.[6] It is composed of high quality real-world binaural recordings for environmental sound research. A subset of this dataset is also annotated with timestamps and labels for individual sound events. A cross-validation setup is provided together with a baseline. Data collection procedures are explained for possible extension by other parties, and identified errors are listed. The dataset was recorded and annotated by two people, each annotating half of it.

• **UrbanSound8K** [30] includes a taxonomy and two audio collections for urban sounds research. *UrbanSound* includes variable length recordings with timestamps for sound events and salience annotations. *UrbanSound8K* contains labeled slices from these events provided in folds for benchmarking with a baseline. The authors compared several classification models on this dataset and observed that deep models only outperformed shallow models when applying data augmentation techniques [29], suggesting that current machine learning approaches require large and varied datasets.

• **ESC** [24] is an open dataset for environmental sound classification that includes an estimation of human-level performance, a baseline, and code for reproducing author's original classification results. This dataset is composed of two main parts. *ESC-50* contains 2000 annotated clips manually annotated by a single person, while *ESC-US* is a compilation of 250k unlabeled clips. The substantial scale difference between them exemplifies how unscalable annotation procedures can limit the size of datasets.

Based on the observations made in this review, we draw a number of conclusions which could be understood as requirements to consider when creating a dataset: *(i)* small datasets may limit the application of certain machine learning techniques, thus larger datasets are desirable; *(ii)* dataset creation processes must be scalable and sustainable to be able to create large datasets; *(iii)* datasets are sometimes re-annotated and complemented with new data which makes them suitable for new tasks; *(iv)* ways to amend existing datasets and turn them into something dynamic should be established; *(v)* it is important to document the workflows of the dataset creation process and make them transparent; *(vi)* intuitive interfaces for navigating the contents of a dataset are useful for gaining insight; *(vii)* providing data splits facilitates reproducibility and benchmarking; and *(viii)* open licenses allowing for the free distribution of the audio content are desirable for higher research impact.

In this paper we introduce Freesound Datasets, an online platform for the collaborative creation of audio datasets which, based on the requirements above, follows principles of transparency, openness, dynamic character, and sustainability. This paper describes our vision of this platform as a long term project and the first steps that we have carried out as a proof-of-concept. The remainder of this paper is organized as follows. In section 2 we outline the core ideas of our vision and the creation of open audio datasets. Section 3 describes the current state of the platform, which at the time of this writing already allows community contributions through validation of existing annotations. In section 4 we present an early snapshot of a large-scale audio dataset built using this platform and which includes audio samples from Freesound[7] organised in a hierarchy based on the AudioSet Ontology. We end this paper with a summary and future work in section 5.

2. FREESOUND DATASETS VISION

We envision a collaborative process for creating audio datasets[8] built by a community of users that can contribute in different aspects of the dataset creation process. After the observations reported in section 1 and by embracing the ideas described in [22] and [33] for sustainable MIR evaluation and reproducibility of computational methods, we define the following principles that apply to our vision of Freesound Datasets and the creation of datasets with the online platform:

• **Transparency**. It is important that workflows in the dataset creation process are transparent so that dataset users are aware of them. This will allow a better understanding of the dataset itself, its potential and limitations. In this respect, facilitating the exploration of the content through intuitive interfaces is a useful functionality that is often overlooked. Moreover, splits of datasets (e.g., train and test) should be proposed and made publicly available for system benchmarking and reproducibility, so that researchers can carry out experiments whose results are directly comparable.

• **Openness**. It is necessary that datasets are completely open, including audio data and ground truth. Both should be available under open licenses that allow the free distribution and reuse of their content. Further, other relevant data generated during the dataset creation process could be

[4] For example, the following tags are equivalent: *beat/beats* or *female singer/female singing/female vocals/woman singing*.

[5] See `https://research.google.com/audioset/dataset/index.html` for further details on how quality is estimated, accessed 26th April 2017.

[6] `http://www.cs.tut.fi/sgn/arg/dcase2016/`

[7] `https://freesound.org/`

[8] By audio datasets we mean datasets that can include not only audio waveforms but also other audio-related data, e.g., tags or descriptions corresponding to the audio samples.

Dataset (release year)	***#clips***	***clip length***	***dataset duration***	***#classes***	***do authors provide audio? / license type***
GTZAN (2002)	1000	30s	8.33h	10 – balanced	yes / questionable © permission
Ballroom (2006)	698	≈30s	≈5.81h	8 – unbalanced	yes / questionable © permission
Million Song Dataset (2011)	1M	-	-	*external resources*	no
MagnaTagATune (2009)	25,856	≈30s	215.46h	188 – unbalanced	yes / open license
AudioSet (2017)	≈2.1M	10s	≈5833h	527 – (un)balanced*	no
TUT Acoustic Scenes (2016)	1560	30s	13h	15 – balanced	yes / open license
UrbanSound8k (2014)	8732	≤4s	8.75h	10 – balanced	yes / open license
ESC-50 (2015)	2000	5s	2.78h	50 – balanced	yes / open license
FSD early snapshot (2017)	23,519	≤90s	119h	398 – unbalanced	yes / open license

Table 1: Characteristics of the reviewed datasets and presented early snapshot of FSD. *Different partitions are provided.

made available (e.g., annotation procedures and the actual raw annotations). Keeping this information as open as possible aids in the detection of potential issues or biasses in the collection process.

• **Dynamism**. It is desirable that the dataset and the procedures carried out in its collection can be the subject of discussion. We have seen that in some previous works criticism of specific datasets is made [36] and alternative versions or subsets of a dataset are proposed [13]. We envision such criticism and proposals happening in a collaborative online platform where detected faults and issues can be discussed and adequately addressed. This would imprint a dynamic character to datasets which could be versioned and updated with contributions from the community.

• **Sustainability**. For such a vision to stand in the long term, a sustainable approach is required not only in terms of gathering audio content and annotations, but also in terms of maintenance. In the envisioned scenario the community acts as a continuous source of information at different levels. Ideally, the community would be self-sufficient as a source of audio-related content by uploading and sharing open content at large scale. Indeed, some previous works have adopted similar approaches for gathering huge amounts of data based on user-provided content, (e.g., AudioSet, based on YouTube videos [9], or ImageNet, based on Flickr and other search engines [5]). In order to construct corresponding ground truth at a large scale level, it is likely that a substantial part of the annotations needs to be gathered through crowdsourcing. Finally, technical maintenance requirements should be kept as low as possible.

2.1 Objectives

Based on the aforementioned principles, the main goals of the Freesound Datasets platform are: *(i)* to **allow the creation and sharing of open audio datasets** containing audio and/or metadata that the community can leverage, be them of general purpose or tailored to specific research problems. And *(ii)* to **allow room for discussion** around the datasets with the purpose of gaining insight and identifying potential improvements. The discussion will ideally be focused not only on the dataset content but also on the workflow of the data collection process. With respect to the content, datasets are intended to be dynamic in the sense that they can evolve over time at multiple levels. Firstly, detected errors, e.g., mislabelings or distorted sounds, should be amended. Secondly, we also consider the possibility of expanding the datasets when more annotated content is generated. Finally, major modifications of a dataset could also be addressed if firm agreement by the research community exists, e.g., modifications of the taxonomy when applicable. As a result, datasets created in such framework may improve over time in terms of quality (better ground truth annotations), and quantity (larger amounts of data). This concept of time-evolving datasets triggers the need for appropriate dataset versioning, leading to consecutive releases of every dataset. Assigning permanent identifiers, e.g., Digital Object Identifiers (DOIs), to releases can help to enhance their unique identification and proper citation, e.g., using a tool like Zenodo[9] [26].

2.2 Two Key Factors

The success of a platform like the one we envision relies on two key factors: *(i)* the need of substantial and regular sources of open and diverse audio-related information, and *(ii)* the need of a community of users able to enrich datasets by providing annotations. In regards to the first factor, we plan to leverage Freesound, an online collaborative audio sample sharing site that has been supporting diverse research and artistic purposes since 2005 [7]. Freesound has 6.5 million registered users and over 340,000 sounds. More than 3000 new sounds are added every month.[10] The most obvious type of content is audio samples, covering a wide range from music samples to environmental sounds, including human sounds, audio effects, etc. Also, users complement the sounds with metadata, e.g., tags, descriptions and comments. A remarkable characteristic is that quality is prioritized over quantity in terms of sound quality and metadata associated to the sound files. All of the content is released under CC licenses. A number of openly available datasets containing Freesound clips have already been used for research [24, 30, 34], showing that it is already a useful source for the creation of datasets.

Regarding the second factor, we need a community around the datasets to enrich their data. Freesound already has a highly engaged community who contributes to the ecosystem by uploading, rating and discussing sounds. We

[9] `https://zenodo.org/`
[10] Data from 26th April 2017.

believe that part of this community will be interested in contributing to a platform like Freesound Datasets. We also hope that a broader community will form around this initiative, consisting of members of the research community and other sound enthusiasts who share our principles.

3. FREESOUND DATASETS PLATFORM

A working prototype of the Freesound Datasets platform has been already deployed and is available at `https://datasets.freesound.org`. For each hosted dataset, [11] the platform provides a number of tools which are described in the following subsections.

3.1 Annotation Tools

This is the set of tools which allows us to provide annotations about the contents of a dataset (i.e., about its audio resources). These annotations could be labels or any sort of information to be used as ground truth. The current version of the platform only implements one way for users to provide annotations. This is a validation task in which users are presented with a number of audio samples and, for each sample, have to assess the presence of a particular sound category. In future iterations of the platform we intend to support other annotation tasks, e.g., adding labels to audio samples or annotating timestamps of specific acoustic events happening within an audio file.

Several options exist for collecting annotations, such as relying on experts or leveraging annotation effort from volunteers in a controlled environment, e.g., a university campus. However, these options seem neither scalable nor sustainable when collecting large-scale datasets. In this case, regardless of the nature of the annotation task, we expect the bulk of annotations to be provided by the community of platform users in a *crowdsourced* fashion. Many existing datasets have been built via crowdsourcing, ImageNet being an iconic example due to its impact in the image processing field [27]. Different kinds of crowdsourcing approaches have been explored, which mainly vary in the way users are rewarded by their contributions, including volunteering-based approaches, games with a purpose, and *paid-for* crowdsourcing [28]. In the sound and music computing field, a number of initiatives have already explored the use of volunteering-based crowdsourcing approaches [11, 37, 22]. The audio community has also extensively explored the *gamification* approach, where the annotation task is presented as an engaging and entertaining experience. Examples of this include games with the purpose of collecting tag annotations, e.g., TagATune [18], the Listen Game [38] and MajorMiner [19]; or games to collect similarity measurements like Spot the Odd Song Out [41]. Finally, a few *paid-for* crowdsourcing experiences exist in the audio field, e.g., ESC dataset [24] and the VU Sound Corpus [40]—both of which contain annotations for Freesound content—or MoodSwings Turk [32] and SocialFX [42] datasets.

Effective quality control plays an important role in determining the success of any data collection venture, especially for crowdsourcing annotations. A common solution to ensure good quality in the gathered annotations is to rely on redundancy. For instance, correct answers can be identified by applying majority voting, or a quality score for each user or *worker* that contributed annotations can be estimated [12]. To ensure that workers are qualified enough to successfully contribute to an annotation task, a proper training phase is typically designed along with a simple task design with clear guidelines [28, 27]. These aspects are considered in the implementation of the annotation tools of the platform. In particular, in our implemented validation task, the training phase shows descriptions and representative audio samples of the sound category to be assessed and its related categories, in order to help the worker form a judgment before proceeding with the task. The mechanisms of quality control used are inspired by those of CrowdFlower [12] and good-sounds.org [1], and include, among other measures, the periodic usage of verification clips to ensure that submitted responses are reliable.

3.2 Other Tools

As described in the principles presented in section 2, it is important to provide an environment for intuitively exploring the content of a dataset, reporting mistakes, making available alternative versions of the dataset, and discussing any of the elements involved in the creation workflow. To this end, we envision a number of tools for the Freesound Datasets platform which provide such functionalities:

- **Audio exploration**. Dataset content can be explored by browsing the audio samples organized by sound categories and samples can be played while visualizing their waveforms. During this process, it is possible to report faulty audio samples or wrong annotations. Systematically flagged examples can be reallocated in a post-processing stage and, for example, marked for further validation.

- **Data downloading**. A single dataset can be made available for download in different releases which include updated ground truth and contents. Audio samples in their original format are provided, thereby allowing researchers to compute any kind of audio features and to adopt any type of machine learning approach. In addition, audio features [13] pre-computed with the Essentia library [3] are available. Along with the audio content, existing Freesound metadata for audio samples (e.g., user-provided title, tags, textual description, etc.), and collected ground truth data can be retrieved. We plan to link specific releases of datasets with DOIs to facilitate referencing.

- **Discussion tools**. The platform encourages discussion [14] about several aspects of the datasets, including but not limited to: faulty audio samples, wrong annotations, annotation tasks protocol (including aspects such as

[11] The current version of the platform is hosting an early snapshot of the dataset described in section 4, and is, at present, the only dataset available.

[12] `https://www.crowdflower.com/`

[13] A list of pre-computed audio features can be found in `https://freesound.org/docs/api/analysis_docs.html`.

[14] Discussion can be joined at `https://github.com/MTG/freesound-datasets/issues`.

a ground truth target taxonomy), and the platform itself. We decided to host discussions in GitHub, since issues can be created and labeled to organize the discussion in topics, and functionalities to track their evolution are available.

4. EARLY SNAPSHOT OF THE FIRST DATASET

As a proof-of-concept of the use of the Freesound Datasets prototype, we present an early snapshot of a dataset that is being created using the platform. We call this dataset the *FreesoundDataset* (FSD). Similarly to the recently introduced AudioSet [9], FSD aims to be a general-purpose and large-scale audio dataset. Audio samples in FSD are therefore labeled using the same hierarchical ontology of AudioSet, which includes 632 audio classes. In the following subsections we describe how this dataset is being created and discuss its status at the time of this writing.

4.1 Data Gathering and Preprocessing

We started building the FSD by automatically populating it with a number of candidate audio samples from Freesound for each category in the AudioSet Ontology. The selection of candidate audio samples was done based on a process of tag matching in which we manually assigned a number of Freesound tags to each category in AudioSet. Then, each category was automatically populated with all sounds from Freesound that contained the selected tags. Suitable tags were found by considering category descriptions provided in AudioSet and obtaining most frequent Freesound tags that co-occur with the target label. After this first selection of tags, a refinement process was performed in some categories by defining tags to be rejected when needed.

After this initial process, which was carried out by three of the authors, we were able to map more than 300,000 Freesound clips to the AudioSet classes. Because sounds in Freesound can widely vary in length, we decided to filter out all samples longer than 90s, which left us with a total of 268,261 candidate samples in the FSD. Each sample was annotated with an average of 2.62 AudioSet categories.

In order to assess the quality of the selected candidates, we conducted an Internal Quality Assessment (IQA). It consisted of a validation task which was carried out using the Freesound Datasets interface. For each sound category, 12 randomly chosen audio samples were presented and a single rater validated the presence of that category in each sample, with possible responses being "Present", "Not Present", and "Unsure". A quality value for each category could be estimated as the percentage of "Present" responses. The IQA, performed by 11 subjects, was useful to *(i)* determine categories with very low quality, likely due to mapping errors to be improved, and *(ii)* to collect feedback about the Freesound Datasets validation task interface and incorporate improvements for next phases.

Finally, we discarded sound categories for which there were less than 40 assigned audio samples and for which the rate of "Not present" responses from the IQA was larger than 75%. Since this process removed half of the musical genre categories, we decided to omit the rest of them too. [15] This left a total of 398 sound categories, with an average of 1553 candidate audio samples per category.

4.2 Validating Annotations

Having the automatic annotations provided by the tag matching algorithm for each sound category, the goal was then to manually validate these annotations at a significantly larger scale than in the IQA. To this end, we recruited 31 participants (mostly masters and PhD students from our department) and asked them to carry out a validation task very similar to that of the IQA for the selected 398 audio categories. To facilitate the task of validating annotations, participants were asked to validate groups of related categories (e.g., sibling categories). In this way they could get familiarized with specific sections of the ontology and provide more consistent validations [28].

Raters were first instructed to access the online platform and choose one of the available groups of categories. Then, for every category, they had an initial training phase where they acclimatized themselves with the category by looking at its location in the hierarchy and a provided textual description, together with representative sound examples. After that, they were presented with 12 randomly chosen audio samples from that category and asked to rate its presence as: "Present and predominant" (PP), "Present but not predominant" (PNP), "Unsure" (U) or "Not Present" (NP). They were instructed that PP means that the type of sound is either isolated from other types of sounds or with low background noise, whereas PNP implies that the audio clip also contains other salient types of sound and/or strong background noise. We added these two levels of "presentness" as during IQA we observed that, in some audio samples, several sound sources and/or acoustic events co-existed with different salience levels and this made the *Present* option rather ambiguous. A similar approach was used in [30]. Hereafter, "Present" = PP + PNP.

After 12 clips were validated, participants could continue validating annotations of the same sound category and 12 new (non-validated) samples were presented. Audio samples were presented using headphones and in a quiet classroom environment. Along with the playable audio and its waveform, participants were also given links to the specific Freesound page for each audio sample. In case of doubt, they could open the page to take their decision based on the sound metadata provided there. Likewise, they could leave general feedback for every audio category through a text box. After two annotation sessions, we gathered more than 42k validations from our 31 participants.

4.3 Characteristics of the FSD

The early snapshot of FSD consists of a list of audio samples together with labels that determine the sound category/ies they belong to, (out of the 398 previously selected). The main statistics of the current snapshot can be seen in Table 2. The table shows, from left to right, *(i)* the number of candidates/annotations that were generated

[15] This was expected as Freesound does not host music content in the traditional sense of "songs".

	Mapped	Validated*	Present*	PP*	PNP*
Annot.	703,359	42,575	25,365	21,526	3839
Clips	268,261	37,398	23,519	20,206	3679
Hours	986	176	119	92.5	30

Table 2: Main statistics of the current snapshot of FSD: number of annotations, audio clips and hours of audio for several sets of data. Columns with * refer to the 398 selected categories. "Present" is the union of PP and PNP. Despite PP and PNP responses being complementary, an *audio clip* annotated in several categories could receive different subjective ratings for each annotation. This is why "Present" $\leq$ PP + PNP.

using the tag-based mapping, *(ii)* the amount of them that have been validated through the conducted experiments, and *(iii)* the amount that has been validated as "Present", (split into *(iv)* PP and *(v)* PNP). Since the sound categories are non-exclusive, the number of annotations is larger than that of audio samples (as mentioned above, the average number of annotations per sample is 2.62). We consider annotations rated as "Present" (PP + PNP) as the most relevant for building the ground truth of a dataset. Our early snapshot contains 23,519 audio clips (119h of audio) with 25,365 annotations. The lengths of the audio clips in the snapshot are irregularly distributed up to a maximum of 90s (40% of the samples are shorter than 6s and around 78% last less than 30s). The number of validated annotations varies among the 398 sound categories, but all of them have at least 72 validated annotations, as designed in the conducted experiments. 75% of the sound categories contain at least 40 valid audio samples (i.e., with annotations rated as "Present"), whereas 20% of them contain more than 80 valid audio samples.

While audio samples come with labels expressing the presence of a sound category, exact start and end times of event occurrences are not given, (i.e., *weakly* labeled data [15]). However, 7015 clips with annotations rated as PP are also shorter than 4s. [16] Based on these two conditions, we can assume that most of those audio clips are just examples for those acoustic events, and can be considered *strongly* labeled data [15]. Thus, we can estimate that the snapshot is composed mainly of weakly labeled data and a small amount of strongly labeled data, in a rough proportion of 70%/30%. Finally, Figure 1 depicts the number of validated annotations gathered for each of the seven *families* of sounds, according to the first layer of the AudioSet Ontology [9].

4.4 Discussion

The FSD snapshot comprises a wide range of audio samples in terms of content, recording scenarios and sources, which presumably makes it representative of real world situations. The differentiation between PP and PNP allows us to have two different subsets of audio presenting different conditions (see section 4.2). Among the shortcomings, the mapping used to generate candidates for AudioSet categories is not optimal, but it is a starting point that will enable future improvements, e.g., a content-based mapping. Another issue is that, for this early snapshot, annotations were only validated by a single rater. While we believe that annotator agreement is required for defining ground truth based on human-sourced annotations [8], it is also true that a number of datasets do not meet this condition [24,23,39].

[16] 4s is taken as a reference length [30] since it was found to be enough for humans to recognize environmental sounds with 82% accuracy [4].

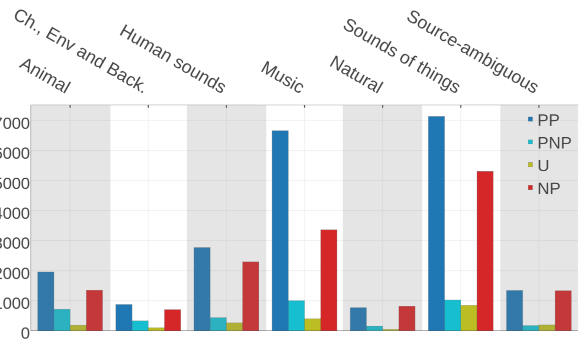


Figure 1: Number of validated annotations (PP, PNP, U, NP) for the different *families* of sounds according to the first layer of the AudioSet Ontology.

Compared to AudioSet, from which FSD takes its ontology, the presented early snapshot is much smaller (Table 1). Currently, AudioSet offers more categories (527) with available content. However, FSD is accompanied by audio waveforms and metadata. Furthermore, FSD includes a mixture of strongly and weakly labeled data whereas in AudioSet only weakly labeled data is provided.

We believe that there exist several applications for FSD within the field of machine listening such as audio event recognition, which enable a variety of specific tasks, e.g., multimedia description, semantically assisted annotation or wildlife monitoring. It also allows a number of approaches like the usage of strongly and weakly labeled data or multimodality, e.g., using audio and metadata for classification. Moreover, future snapshots of the FSD will include improved ground truth data which will further increase its value for research.

5. SUMMARY AND FUTURE WORK

In this paper we have introduced Freesound Datasets, an online platform for the collaborative creation of open audio datasets. We have outlined the core ideas of our vision on the creation of open audio datasets. The current state of the online platform has been described and we have also presented an early snapshot of a large-scale audio dataset built using this platform. This being a long term project, only first steps have been carried out. Next milestones include enhancing the platform functionalities and adding new ones that allow us to crowdsource annotations reliably for new annotation tasks, while promoting discussion around the datasets. After gathering more validated annotations for FSD, we will make the first release including data splits and a baseline for reproducibility and benchmarking. We hope that our platform can serve as an inspiration for creating datasets of completely different nature.

6. ACKNOWLEDGMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688382 "AudioCommons", and from the Maria de Maeztu Units of Excellence Programme (MDM-2015-0502). The authors would like to thank Nat Friedman for his support through the AI Grant, Sertan Senturk for his valuable feedback, and the volunteers who participated in the conducted experiments.

7. REFERENCES

[1] Giuseppe Bandiera, Oriol Romani Picas, Hiroshi Tokuda, Wataru Hariya, Koji Oishi, and Xavier Serra. Good-sounds.org: A framework to explore goodness in instrumental sounds. In *International Society for Music Information Retrieval Conference*, pages 414–419, 2016.

[2] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[3] Dmitry Bogdanov, Nicolas Wack, Emilia Gómez, Sankalp Gulati, Perfecto Herrera, Oscar Mayor, Gerard Roma, Justin Salamon, José R Zapata, Xavier Serra, et al. Essentia: An audio analysis library for music information retrieval. In *ISMIR*, pages 493–498, 2013.

[4] Selina Chu, Shrikanth Narayanan, and C.-C. Jay Kuo. Environmental sound recognition with time-frequency audio features. *Transactions on Audio, Speech, and Language Processing*, 17(6):1142–1158, August 2009.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[6] Sander Dieleman and Benjamin Schrauwen. End-to-end learning for music audio. In *Proceedings of the Acoustics, Speech and Signal Processing International Conference*, pages 6964–6968. IEEE, 2014.

[7] Frederic Font, Gerard Roma, and Xavier Serra. Freesound technical demo. In *Proceedings of the ACM International Conference on Multimedia*, pages 411–412. ACM, 2013.

[8] Peter Foster, Siddharth Sigtia, Sacha Krstulovic, Jon Barker, and Mark D Plumbley. Chime-home: A dataset for sound source recognition in a domestic environment. In *Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–5. IEEE, 2015.

[9] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dartaset for audio events. In *Proceedings of the Acoustics, Speech and Signal Processing International Conference*, 2017.

[10] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[11] Perfecto Herrera, Òscar Celma, Jordi Massaguer, Pedro Cano, Emilia Gómez, Fabien Gouyon, Markus Koppenberger, David García, J Mahedero, and Nicolas Wack. Mucosa: A music content semantic annotator. In *Proceedings of the International Conference on Music Information Retrieval*, pages 77–83, 2005.

[12] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on Human Computation*, pages 64–67. ACM, 2010.

[13] Corey Kereliuk, Bob L Sturm, and Jan Larsen. Deep learning and music adversaries. *Transactions on Multimedia*, 17(11):2059–2071, 2015.

[14] Florian Krebs, Sebastian Böck, and Gerhard Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 227–232, 2013.

[15] Anurag Kumar and Bhiksha Raj. Audio event and scene recognition: A unified approach using strongly and weakly labeled data. *arXiv preprint arXiv:1611.04871*, 2016.

[16] Edith Law and Luis Von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1197–1206. ACM, 2009.

[17] Edith Law, Kris West, Michael I Mandel, Mert Bay, and J Stephen Downie. Evaluation of algorithms using games: The case of music tagging. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 387–392, 2009.

[18] Edith LM Law, Luis Von Ahn, Roger B Dannenberg, and Mike Crawford. Tagatune: A game for music and sound annotation. In *Proceedings of the International Symposium on Music Information Retrieval*, volume 3, page 2, 2007.

[19] Michael I Mandel and Daniel PW Ellis. A web-based game for collecting music metadata. *Journal of New Music Research*, 37(2):151–165, 2008.

[20] Ugo Marchand and Geoffroy Peeters. The Extended Ballroom Dataset. Late-Breaking Demo Session of the 17th International Society for Music Information Retrieval Conference. 2016.

[21] Gonçalo Marques, Marcos Aurélio Domingues, Thibault Langlois, and Fabien Gouyon. Three current issues in music autotagging. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 795–800, 2011.

[22] Brian McFee, Eric Humphrey, and Julián Urbano. A plan for sustainable mir evaluation. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 285–291, 2016.

[23] Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. In *Proceedings of the European Signal Processing Conference*, pages 1128–1132, 2016.

[24] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the ACM International Conference on Multimedia*, pages 1015–1018. ACM, 2015.

[25] Jordi Pons, Olga Slizovskaia, Rong Gong, Emilia Gómez, and Xavier Serra. Timbre analysis of music audio signals with convolutional neural networks. *arXiv preprint arXiv:1703.06697*, 2017.

[26] Stefan Pröll and Andreas Rauber. Enabling reproducibility for small and large scale research data sets. *D-Lib Magazine*, 23(1/2), 2017.

[27] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[28] Marta Sabou, Kalina Bontcheva, Leon Derczynski, and Arno Scharl. Corpus annotation through crowdsourcing: Towards best practice guidelines. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 859–866, 2014.

[29] Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[30] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the ACM International Conference on Multimedia*, pages 1041–1044. ACM, 2014.

[31] Alexander Schindler and Andreas Rauber. Capturing the temporal domain in EchoNest features for improved classification effectiveness. In *International Workshop on Adaptive Multimedia Retrieval*, pages 214–227. Springer, 2012.

[32] Jacquelin A Speck, Erik M Schmidt, Brandon G Morton, and Youngmoo E Kim. A comparative study of collaborative vs. traditional musical mood annotation. In *International Society for Music Information Retrieval Conference*, pages 549–554, 2011.

[33] Victoria Stodden, Marcia McNutt, David H Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A Heroux, John PA Ioannidis, and Michela Taufer. Enhancing reproducibility for computational methods. *Science*, 354(6317):1240–1241, 2016.

[34] Dan Stowell and Mark D Plumbley. An open dataset for research on audio field recording archives: freefield1010. *arXiv preprint: 1309.5275*, 2013.

[35] Bob L Sturm. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, pages 29–66. Springer International Publishing, 2012.

[36] Bob L Sturm. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. *arXiv preprint: 1306.1461*, 2013.

[37] Nikolaos Tsipas, Charalampos A Dimoulas, George M Kalliris, and George Papanikolaou. Collaborative annotation platform for audio semantics. In *Audio Engineering Society Convention 134*. Audio Engineering Society, 2013.

[38] Douglas Turnbull, Ruoran Liu, Luke Barrington, and Gert RG Lanckriet. A game-based approach for collecting semantic annotations of music. In *Proceedings of the International Conference on Music Information Retrieval*, volume 7, pages 535–538, 2007.

[39] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[40] Emiel van Miltenburg, Benjamin Timmermans, and Lora Aroyo. The VU sound corpus: adding more fine-grained annotations to the freesound database. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2016.

[41] Daniel Wolff. *Spot the Odd Song Out: Similarity Model Adaptation and Analysis using Relative Human Ratings*. PhD thesis, City University London, 2014.

[42] Taylor Zheng, Prem Seetharaman, and Bryan Pardo. Socialfx: Studying a crowdsourced folksonomy of audio effects terms. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 182–186. ACM, 2016.

FROM BACH TO THE BEATLES: THE SIMULATION OF HUMAN TONAL EXPECTATION USING ECOLOGICALLY-TRAINED PREDICTIVE MODELS

Carlos Cancino-Chacón[1,2] **Maarten Grachten**[2] **Kat Agres**[3]

[1] Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria
[2] Department of Computational Perception, Johannes Kepler University, Linz, Austria
[3] Institute of High Performance Computing, A*STAR, Singapore

carlos.cancino@ofai.at, maarten.gracthen@ofai.at, kat_agres@ihpc.a-star.edu.sg

ABSTRACT

Tonal structure is in part conveyed by statistical regularities between musical events, and research has shown that computational models reflect tonal structure in music by capturing these regularities in schematic constructs like pitch histograms. Of the few studies that model the acquisition of perceptual learning from musical data, most have employed self-organizing models that learn a topology of static descriptions of musical contexts. Also, the stimuli used to train these models are often symbolic rather than acoustically faithful representations of musical material. In this work we investigate whether sequential predictive models of musical memory (specifically, recurrent neural networks), trained on audio from commercial CD recordings, induce tonal knowledge in a similar manner to listeners (as shown in behavioral studies in music perception). Our experiments indicate that various types of recurrent neural networks produce musical expectations that clearly convey tonal structure. Furthermore, the results imply that although implicit knowledge of tonal structure is a necessary condition for accurate musical expectation, the most accurate predictive models also use other cues beyond the tonal structure of the musical context.

1. INTRODUCTION AND RELATED WORK

Computers are increasingly being used to perform music-related tasks (automated music analysis, music recommendation, composition, etc). To perform such tasks reliably, there is a need for computers to grasp concepts that are relevant to our perception and understanding of music [37]. Empirical findings from music psychology are valuable in this respect, since they shed light on the process of human music perception and cognition.

We know from extensive research in music psychology that listeners implicitly extract statistical properties governing tonal structure through exposure to music [3,19,29]. The tonal *stability*, or relative importance, of notes in a key may be largely due to the frequency of occurrence of pitches in a piece of music. The more foundational pitches (e.g., C, E, and G in the key of C major) will tend to be anchor points in the music, and will often occur on metrically-important positions [21,26].

Through exposure to these kinds of melodic (and harmonic) statistical properties, listeners form an implicit mental model of tonality. Evidence for this has been provided, for example, through the seminal work of Krumhansl and colleagues employing a ‘probe-tone paradigm’, in which listeners rate how well the last pitch, or probe-tone, of a musical sequence fits in with the previous context. When provided with a tonal context, such as an ascending or descending musical scale, listeners perceive certain pitches as sounding more appropriate than others [19, 21, 22]. The profile of listeners’ ratings of probe-tones reflects a tonal hierarchy, and it is this hierarchy of pitch stabilities that plays a large role in governing tonal perception. The extent to which different music listening behaviors and one’s musical ‘culture’ influence tonal perception is an open question, although evidence exists that Western classical music training results in differentiated, and often more nuanced, pitch expectations and probe-tone profiles [4,10,20,34].

To model these types of findings, computational models of tonal perception typically aim to provide methods that, given a musical context, compute a response that can be judged to be more or less appropriate for the implicit tonality of that context. Given the predominance of the probe-tone paradigm for studies of human tonal perception, a common practice is to elicit a *quasi*-goodness-of-fit response from the model for a probe-tone given a musical stimulus, such that the responses can be compared to human probe-tone ratings (e.g. [6,23,25,35]). Another way to judge the responses is to define a metric over the responses and compare the resulting topology to geometric constructs from music theory, such as the Tonnetz [36], a toroidal representation of key distance [18], or the circle of fifths [6].

The computational models proposed in the literature tend to emphasize one of various different factors that are thought to play a role in tonal perception. Whereas some works seek to explain empirical results mainly by a computational account of the lower levels of the auditory system [23,25], others focus more strongly on the role of long-term memory in tonal perception [6, 24, 35].

Of the models that involve some representation of long-term memory, most do not account for that representation in an *ecologically plausible* manner, meaning that there is no plausible simulation of how the long-term memory representations come about as a result of long-term exposure to music. First, long-term memory is usually modeled by some form of self-organization of static representations of musical contexts or events, producing a low-dimensional map of musical stimuli, in which the neighborhood relationship captures semantic information (such as tonal affinity) [6, 24, 35, 36]. Although the principle of self-organization has been used to account for the structure of cortical maps such as those in the visual cortex [12], there is no evidence that this principle also underpins long-term memory. Moreover, the fact that musical contexts are being mapped as static entities is at odds with the fundamentally temporal nature of the music listening process. As formalized in the *predictive coding* framework [13], an increasingly prominent idea in cognitive science is that of anticipation as a universal driving force for cognition [8, 11].

Music researchers have also focused on the temporal dynamics of tonal and harmonic expectations (e.g., [32] and [30]), and some models based on self-organizing maps (SOMs) [17] do account for effects of temporal order in musical listening [25, 35]. A limitation, however, is that these effects are not taken into account in the training of the maps, representing the learning process that forms long term memory of music. Toiviainen and Krumhansl [36] also employ a SOM, but, as they state, use it for visualization purposes, and not "to simulate any kind of perceptual learning that would occur in listeners through, for instance, the extraction of regularities present in Western music." Although the model offered by [9] does learn from musical sequences to predict tonal expectation in listeners, the model itself does not use sequential tonal information to learn and drive its predictions.

Another limitation of most long-term memory models for tonal learning is that they work with stimuli that are reduced in one or more ways. For example, the input may consist of discrete representations of tones such as MIDI note numbers [6], pitch classes [35], artificial harmonic representations [2], or of artificial harmonic sounds such as Shepard tones [25]. Furthermore, the musical material that a model is exposed to may be limited to monophonic melodic lines [6], sets of chords or harmonic cadences [25], or even a set of probe-tone profiles [36]. A notable exception to this is [24], which uses an audio recording of Bach's Well Tempered Clavier (WTC), performed on a harpsichord, to train a SOM by converting the acoustic signal to *auditory images*. The work of [9] also takes an ecological approach by using real audio and plausible psychological representations, with multiple representations along the sensory-cognitive spectrum, to better account for human tonal expectation.

The central question of this work is whether sequential predictive models of musical memory induce memory representations that convey tonal structure, similar to the static self-organizing models that are predominant in computational modeling of tonal learning. To answer this question, we employ Recurrent Neural Networks (RNNs) and variants such as Long Short Term Memory (LSTM) [16], which provide a common and effective modeling approach to the task of predicting future input from a history of past inputs. A further objective is to see whether tonal expectations can also be elicited in the models by training on ecologically valid musical data rather than artificial data. The present work approaches ecological validity in four ways: 1) using commercial audio recordings rather than symbolic or reduced music, 2) employing a psychoacoustically plausible input representation (the Constant-Q representation), 3) training corpora that span more than one genre (Bach and the Beatles) to better reflect a lister's musical experience, and 4) using more than one key to train the model (much related research transposes the training dataset to one key, e.g. [1, 6]). We test the effect of the training data on the strength and character of the tonal expectations of the model. Furthermore we measure the impact of shuffling the training data to gauge the importance of the sequential order of the music. Finally, we investigate the relationship between the training objective of the models (to predict the immediate future based on the present and past), and the strength of tonal hierarchy in the model expectations.

The paper is structured as follows. In Section 2, we provide a brief description of both the audio representation and of the predictive RNN models used in our experiments. Section 3 briefly reviews the datasets used to train the RNN models, and presents and discusses a comparison of model predictions to the results of probe-tone experiments. Finally, conclusions and future work are presented in Section 4.

2. METHOD

In this Section we describe the predictive models we use for our experiment (Section 2.2), and the audio representation used to present the data to the models (Section 2.1).

2.1 Constant-Q Transform

The Constant-Q Transform (CQT) [5] is a discrete frequency domain representation of audio. Although the CQT was not conceived explicitly as a model of the human auditory periphery, it shares an important characteristic with such models in that it samples the frequency axis logarithmically—a psychoacoustically plausible feature, since human listeners tend to perceive pairs of tones as equidistant when their respective frequency *ratios* are equal. The CQT is widely used in applications involving musical audio, since its frequency bins can be configured to match the 12 tone octave division of Western music. To obtain a CQT spectrogram, conveying the change in frequency content of audio over time, the CQT can be computed over series of consecutive short, windowed segments of the audio, analogous to the Short-Time Fourier Transform.

2.2 Recurrent Neural Networks

An RNN is a neural architecture that allows for modeling dynamical systems [15]. Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ be a sequence of N-dimensional (normalized) input vectors and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ be its corresponding sequence of outputs. An RNN provides a natural way to model $\boldsymbol{x}_{t+1}$, the next event in the sequence, by using the outputs of the network to parametrize a predictive distribution given by

$$p(x_{t+1,i} \mid \boldsymbol{x}_t, \ldots, \boldsymbol{x}_1) = y_{t,i} \tag{1}$$

where $x_{t+1,i}$ and $y_{t,i}$ are the i-th component of $\boldsymbol{x}_{t+1}$ and $\boldsymbol{y}_t$ respectively.

The basic component of an RNN is the *recurrent layer*, whose activation at time t depends on both the input at time t and its activation at time $t-1$. Although theoretically very powerful, in practice RNNs with *vanilla* recurrent layers are known to have problems learning long term dependencies due to a number of problems, including vanishing and exploding gradients [27]. Other recurrent layers such as LSTM layers [16] and gated recurrent units (GRUs) [7] try to address some of these problems by introducing special structures within the layer, such as purpose-built memory cells and gates to better store information. More recently, recurrent layers with multiplicative integration (MI-RNNs) [38] have been shown to extend the expressivity of traditional additive RNNs by changing the way the information from different sources is aggregated within the layer while introducing just a small number of extra parameters.

Given a training set consisting of inputs and targets, the parameters of an RNN can be learned in a supervised fashion by minimizing the cross entropy (CE) between its predictions and the targets.

A more thorough description of RNNs lies outside of the scope of this paper. For a more mathematical formulation of LSTMs and GRUs, we refer the reader to [7,15]. A more detailed description of MI-RNNs can be found in the Appendix of [38].

3. EXPERIMENTS

In this Section we describe the two datasets used for the experiments in this paper (Section 3.2) and briefly review the theoretical framework of probe-tone experiments (Section 3.1), as well as a description of the training procedure (Section 3.3). In Section 3.4 the results of the probe-tone experiments are presented and discussed.

3.1 Probe-Tone Experiments

A probe-tone test is an experimental framework to quantitatively assess the hierarchy of tonal stability [19]. This experimental framework consists of a set of musical stimuli like scales or cadences that unambiguously instantiate a specific musical context, such as a key. After presenting the stimulus, a participant hears a set of probe-tones, usually the set of 12 pitch classes, and the participant, either a human participant or a computer model, is asked to rate on quantitatively how well the probe-tones fit the musical stimulus.

Let $\mathbf{X} = \{\boldsymbol{x}_1, \cdots \boldsymbol{x}_T\}$ be an input musical stimulus, and $\mathbf{T} = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_{12}\}$ the set of probe-tones each corresponding to one of the 12 pitch classes. In order to quantitatively assess how well a probe-tone $\boldsymbol{\tau}$ fits the musical stimulus, we compare $\boldsymbol{y}^*$, the predictions of the RNN given the input stimulus, and the probe-tone using the Kullback-Leibler (KL) divergence.

In this paper, we use the above described model to reproduce the classic Krumhansl and Kessler (KK) probe-tone experiment [18]. This study is interesting for us mainly because 1) the probe tone contexts are polyphonic, featuring scales, chords, and cadences, thus highlighting capability of the proposed model to process polyphonic data, and 2) only expert listeners were tested (the participants of this experiment had an average of 11 years of formal music education), allowing us to directly compare the expectations of the model to those of an expert listener. The setup for this experiment requires a set of 14 tonal contexts [1]: ascending major and (harmonic) minor scales, three chord cadences (II-V-I, IV-V-I, VI-V-I) in both major and minor and individual chords (major triad, minor triad, dominant seventh chord and diminished chord). In our experiments, we transpose each context to every key, yielding 12 variants of each context. In order to aggregate the results over all keys, we average the KL divergence for each context.

Following the original experimental setup, both stimuli and probe-tones are generated using Shepard-tones, which consists of five sine wave components in a five-octave range from 77.8 Hz to 2349 Hz, with an amplitude envelope such that the low and high ends of the range approached hearing threshold [19].

We use Pearson's correlation coefficient to compare the goodness-of-fit of the probe-tones learned by the models with the KK probe-tone ratings.

3.2 Datasets

The WTC is a collection of 96 pieces for solo keyboard, consisting of two sets of 24 Preludes and Fugues in each key. Composed by Johann Sebastian Bach, the WTC is widely recognized as one of the most important works in Western music. We use a performance of the WTC by renowned Canadian pianist Angela Hewitt [2]. The total duration of this recording is 4.5 hours. We perform data

[1] See Table 1 in [18].

[2] Hyperion CDS44291/4 1998

augmentation on the WTC dataset by pitch shifting each recording between -6 and $+5$ semitones using *pyrubberband*[3]. We thus obtain 1152 pieces for the WTC, equivalent to nearly 53 hours of music.

Additionally, we use a second dataset consisting of 12 Albums by The Beatles, with a total of 179 songs with an approximate duration of 7.5 hours. We do not perform data augmentation on the Beatles data[4].

To facilitate the exposure of the models to regularities in the change of pitch content over time, we do not compute the CQT spectrograms by taking equidistant frames in absolute time, but instead link the spectrogram frame rate to the musical time, such that the instantaneous frame rate is always an integer multiple or submultiple of the beat rate. For the Beatles data, we do so by using publicly available beat annotations[5]. For the WTC recording by Hewitt no such annotations were available, but versions in Humdrum format of the pieces were obtained from KernScores[6]. The Humdrum files were converted into MIDI files, which were manually edited using MuseScore to match the repetitions as performed by Hewitt. By aligning piano-synthesized audio renderings of the MIDI files to the Hewitt recordings using the method described in [14], beat times were automatically inferred for the recordings.

Based on the typical temporal densities of musical events in the two datasets, we chose a temporal resolution of a quarter beat for the CQT spectrogram in the case of the Beatles, and a sixteenth beat in the case of WTC. We will return to this issue in Section 3.3.1.

Each slice of the CQT spectrogram is a 334-dimensional vector that represents frequencies between 27.5 and 16744.04 Hz with a resolution of 36 frequency bins per octave. This configuration was chosen to avoid spectral leakage between adjacent frequency bins, and is similar to the one used by Purwins et. al. [28]. Additionally, this configuration is also able to accommodate at least the fundamental frequency plus at least three harmonics to the highest note of a piano. We normalize each slice of the CQT to lie between 0 and 1.

3.3 Training

For the experiments in this paper we use RNNs as described in Section 2.2 as a sequential alternative to the static models typically used for tonal learning, such as SOMs and RBMs. To get an impression of the performance of sequential models in general for this task, we test five different variants of the recurrent layer, namely a vanilla RNN (vRNN), an LSTM, a GRU, and two models with multiplicative integration: a vanilla recurrent layer (vRNN/MI) and an LSTM/MI[7]. In all variants, the model has a single hidden recurrent layer with 75 $\tanh$ units and an output layer with sigmoid units. The use of different model variants also allows us to investigate the relationship between the prediction error and the similarity of model expectations to human goodness-of-fit ratings of probe-tones.

In order to investigate the kind of statistical regularities in music that produce human-like probe-tone results, we train each model on two different versions of each dataset, namely training the model using the original data, and training the model shuffling the spectrograms in a piece-wise fashion. Randomizing inputs per piece preserves the global pitch distribution of the piece but disrupts temporal cues to musical expectations, like harmonic progressions and voice-leading.

We split each dataset into 5 equally sized non-overlapping folds, resulting in 4 RNN architectures $\times$ 2 orderings of the CQT spectrograms (original vs. randomized spectrograms) $\times$ 5 folds $\times$ 2 datasets $=$ 80 trained models. For each fold, 80% of the pieces (ca. 184 pieces for the WTC and 29 for the Beatles) are randomly selected to be used for training and 20% for testing (ca. 46 pieces for the WTC and 7 for the Beatles). The predictive accuracy of each model is measured by the mean cross-entropy (MCE) on the test set. The models are trained using RMSProp [33], a variant of stochastic gradient descent that adaptively updates the step-size using a moving average of the of the magnitude of the gradients. The initial learning rate is set to 10^{-3}. The gradients are computed using truncated back propagation through time, where computation of the gradients is truncated after 100 steps and are clipped at 1. Each training batch consists of 20 sequences of 100 CQT slices. Each sequence is selected randomly out of the training data. Thus, an epoch of training corresponds to the model seeing roughly the same number of time steps as in the whole fold. Early stopping is used after 100 epochs without any improvement in the test set. All RNNs are implemented using *Lasagne*[8]. We provide online supplementary materials describing all of the technical details for performing the probe-tone experiments in this paper[9].

3.3.1 *Biasing Learning Towards Predicting Change*

A crucial question when applying discrete time recurrent models to a continuous stream of data such as audio is how to choose the rate of discrete time steps with respect to the absolute time of the data. This choice depends on the approximate rate or temporal density of relevant events in the data—in our case the notes that make up the musical material. Ideally, we would like the discrete time steps to be small enough to capture the occurrence of even the shortest notes individually, but if the discrete time step is chosen much smaller than the median event rate, this leads to strong correlations between data at consecutive time steps. A result of this is that training models to predict the data at time step $t+1$ teaches them to strongly expect the data

[3] `https://github.com/bmcfee/pyrubberband` Accessed April 2017.

[4] Exploratory experiments showed that using pitch shifting on the Beatles songs worsened the predictions of the RNNs. This worsening might be due to the fact that most of these recordings include several instruments and voices, including unpitched percussion instruments.

[5] `http://isophonics.net/content/reference-annotations-beatles`. Accessed April 2017.

[6] `http://kern.ccarh.org`. Accessed April 2017.

[7] In the current experiments the GRU/MI yielded pathological results, possibly due to an implementation problem.

[8] `https://github.com/Lasagne/Lasagne`. Accessed April 2017.

[9] `http://carloscancinochacon.com/documents/online_extras/ismir2017/sup_materials.html`.

at $t+1$ to be approximately equal to the data at t. Choosing a larger discrete step size for the model alleviates this problem, but has the disadvantage that the data the model sees at a particular time may actually be an average over consecutive events that happened within that larger step.

We slightly revise the training objective of the models as a remedy to this unfortunate trade-off. This revised objective biases the models to care more about correctly predicting the data at $t+1$ when the change from t to $t+1$ is large (e.g. the start of a new note) than when it is small (e.g a transition without any starting or ending note events). This allows us to use a relatively small step size without causing the models to trivially learn to expect the data to stay constant between consecutive time steps.

More specifically, we modify the original cross-entropy objective CE_t by multiplying it with a time-varying weight w_t as follows:

$$\tilde{CE}_t \leftarrow w_t CE_t, \tag{2}$$

where w_t is given by

$$w_t = \begin{cases} 1 & \text{if } \sum_i^N |x_{t+1,i} - x_{t,i}| > \varepsilon \\ \beta & \text{otherwise} \end{cases} \tag{3}$$

where $\varepsilon \in \mathbb{R}$ acts as a threshold distinguishing small and large change transitions, and $\beta \in \mathbb{R}$ controls the relative influence of prediction errors on the training in the case of small change transitions [10] .Based on an informal inspection of the model predictions in a grid search on β and ε, we choose $\beta = 10^{-3}$, and ε such that

$$P_{training}\left(\sum_i^N |x_{t+1,i} - x_{t,i}| \leq \varepsilon\right) = 0.505 \tag{4}$$

where $P_{training}(X)$ denotes the empirical probability of event X under the training data.

3.4 Results and Discussion

Figure 1 compares the aggregation of the probe-tone ratings (see Section 3.1) for both major and minor contexts with the expectations of the best predictive models (as in lowest MCE in the test set) for each dataset, which in both cases is the GRU trained without shuffling the data. Table 1 shows the correlation between the KK profiles and the model expectations. All of the correlations are statistically significant ($p < 0.0002$). Although the values obtained for the models trained on the Beatles data are slightly lower, the strength of the correlations between the empirical data and the model simulation is on a par with those reported in the literature [24, 35]. Pairwise two-sample Kolmogorov–Smirnov tests (KK vs. Hewitt/WTC, KK vs. Beatles and WTC/Hewitt vs. Beatles) reveal that the three profiles are not significantly different from one another ($p \geq 0.19$).

The above result shows that the expectations of the proposed models reflect the tonal characteristics of the musical context that evoked those expectations. This is expected but not trivial, since the training objective of the models is solely to predict how a given sequence of musical information (in the form of CQT spectrograms) will continue. An interesting question is therefore whether there is any relation between the predictive accuracy of a model (that is, how successfully it predicts future musical events based on the music up to now), and the correlation of its probe-tone response to that of human subjects. In the plots of Figure 2, the vertical axis measures the Pearson correlation coefficient of the probe-tone responses of different models with the KK profiles, and the horizontal axis measures predictive accuracy of the models, in terms of their MCE over the test data. For each model type in the legend, there are five different scatter points, representing models trained on each of five non-overlapping folds of

[10] We empirically found a binary distinction between small and large change transitions to be more effective than a gradual weighting scheme

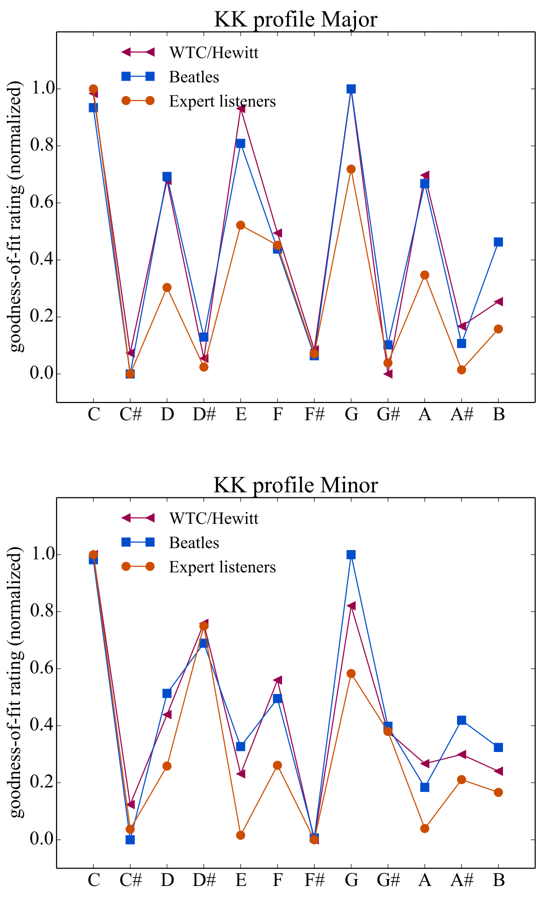


Figure 1. Expectations of the models trained on WTC and Beatles datasets compared to average probe-tone ratings by expert listeners for major and minor contexts [18]

	KK major	KK minor
WTC/Hewitt	0.915	0.940
Beatles	0.900	0.885

Table 1. Pearson's correlation between normalized predictions of the model with the lowest mean cross-entropy for each dataset and KK major and minor profiles

the data (see Section 3.3). The vertical coordinate of each scatter point is the result of averaging the correlation coefficients of responses to all transpositions of the probe-tone stimuli (see Section 3.1).

The scatterplots for the WTC and Beatles show that on average, MCE is higher for models trained the Beatles data than for those trained on WTC. This is likely due to the fact that the WTC data are single instrument recordings (piano) with relatively homogeneous CQT spectrograms, whereas the Beatles recordings are multi-instrumental, leading to more dense and complex CQT spectrograms.

For the WTC data, training models on shuffled CQT data has a noticeable negative impact on both predictive accuracy and tonal expectations. For the Beatles data this effect is less pronounced. There may be multiple explanations for this. First, even if the WTC data, being solo piano recordings, are spectrally simpler, they are probably more complex both harmonically and melodically than the Beatles data. As such, shuffling the data temporally is more of a disruption to the WTC data than to the Beatles data. Secondly, the WTC pieces tend to include brief modulations away from the main key of the piece. This means that shuffling the data within a piece may mix data from different keys, making prediction more dificult.

Despite these differences, both WTC- and Beatles-trained models roughly show the same overall pattern: models with low predictive error have high KK correlations, whereas models with high predictive error may or may not have high KK correlations. This suggests that in order to form accurate musical expectations, it is indispensable to have a notion of tonal structure. But conversely, having a notion of tonal structure by itself is not a sufficient condition for accurate musical expectations. This implies that there are other factors beyond tonality, such as voice leading, rhythm, and cadential structure, that help predict how a given musical context will continue (see [31] and [32] for behavioral evidence to this effect).

4. CONCLUSION

In this paper we showed that the expectations of ecologically trained predictive models of music exhibit tonal structure very similar to that observed in humans through probe-tone experiments. We believe this finding is relevant, since most computational modeling approaches to tonal perception that involve a representation of statistical regularities in musical data do not account for the perceptual learning of such regularities in a plausible way. The musical expectations of the models used here are formed by training the model to reduce the prediction error for future musical events based on the musical context up to the present—a cognitively plausible task according to the predictive coding theory of the brain [8]. Furthermore, we demonstrate that tonal learning within such models is not only possible based on training data known to exhibit rich tonal qualities (Bach's WTC, artificial cadences), but also occurs as an effect of exposure to audio representations of "real-world" popular and harmonically simpler music (The Beatles). This more accurately mirrors the kind of musical

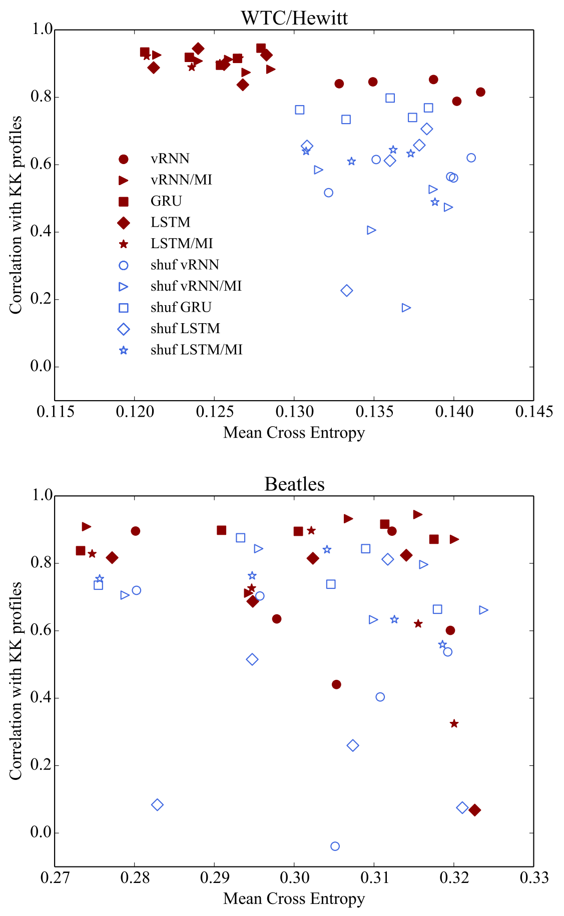


Figure 2. Similarity of model expectations to human probe-tone ratings (Pearson correlation coefficient) plotted versus the mean cross-entropy of the models over a test set; *shuf* denotes models trained on shuffled data

exposure people have, even if real-world musical enculturation would typically involve a wider range of music.

An analysis of the relation between the predictive accuracy of the model and the degree of tonal structure exhibited by model expectations shows that tonal expectations are a necessary but not a sufficient condition for accurate musical expectations. This suggests that there are other—presumably temporal—cues to musical expectation beyond tonal structure. Evidence for this is the fact that models trained on temporally shuffled WTC data form less accurate expectations than models trained on the ordered data. This effect is not observed for the Beatles data, possibly because of its simpler melodic and harmonic structure.

The empirical validation of the models we presented here offers various further avenues of research that we have not yet pursued. For example, a qualitative analysis of the learned representations of the models may provide further insights into the cues that influence musical expectations. In models with multiple hidden layers, an interesting question is where the different learned representations lie along the sensory-cognitive spectrum of tonal representations, as hypothesized by [9].

5. ACKNOWLEDGMENTS

This work has been partly funded by the European Research Council (ERC) under the EUs Horizon 2020 Framework Programme (ERC Grant Agreement No. 670035, project CON ESPRESSIONE). We thank Carol L. Krumhansl for providing the probe-tone data.

6. REFERENCES

[1] K. Agres, C. Cancino, M. Grachten, and S. Lattner. Harmonics co-occurrences bootstrap pitch and tonality perception in music: Evidence from a statistical unsupervised learning model. In *CogSci 2015: The annual meeting of the Cognitive Science Society*, 2015.

[2] K. Agres, C. E. Cancino Chacón, M. Grachten, and S. Lattner. Harmonics co-occurrences bootstrap pitch and tonality perception in music: Evidence from a statistical unsupervised learning model. In *CogSci 2015: The annual meeting of the Cognitive Science Society*, Pasadena, CA, USA, 2015.

[3] K. Agres, S. Abdallah, and M. Pearce. Information-theoretic properties of auditory sequences dynamically influence expectation and memory. *Cognitive Science*, 2017.

[4] G. M. Bidelman, S. Hutka, and S. Moreno. Tone language speakers and musicians share enhanced perceptual and cognitive abilities for musical pitch: evidence for bidirectionality between the domains of language and music. *PloS one*, 8(4):e60676, 2013.

[5] J. C. Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, January 1991.

[6] C. E. Cancino Chacón, S. Lattner, and M. Grachten. Developing tonal perception through unsupervised learning. In *Proceedings of the 15th International Conference on Music Information Retrieval*, Taipei, Taiwan, October 2014.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[8] A. Clark. Whatever next? predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[9] T. Collins, B. Tillmann, F. S. Barrett, C. Delbe, and P. Janata. A combined model of sensory and cognitive representations underlying tonal expectations in music: from audio signals to behavior. *Psychological review*, 121(1):33, 2014.

[10] L. L. Cuddy and B. Badertscher. Recovery of the tonal hierarchy: Some comparisons across age and levels of musical experience. *Perception & Psychophysics*, 41(6):609–620, 1987.

[11] D. C. Dennett. *Consciousness Explained*. Penguin Books, 1991.

[12] R. Durbin and G. Mitchison. A dimension reduction framework for understanding cortical maps. *Nature*, 343:644–647, 1990.

[13] K. Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1456):815–836, 2005.

[14] M. Grachten, M. Gasser, A. Arzt, and G. Widmer. Automatic alignment of music performances with structural differences. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil, November 2013.

[15] A. Graves. Generating Sequences With Recurrent Neural Networks. *arXiv*, 1308:850, 2013.

[16] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[17] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernetics*, 43:59–69, 1982.

[18] C. L. Krumhansl and E. J. Kessler. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological review*, 89(4):334–368, July 1982.

[19] C. L. Krumhansl. *Cognitive foundations of musical pitch*. Cognitive foundations of musical pitch. Oxford University Press, New York, 1990.

[20] C. L. Krumhansl. Music psychology: Tonal structures in perception and memory. *Annual review of psychology*, 42(1):277–303, 1991.

[21] C. L. Krumhansl and L. L. Cuddy. A Theory of Tonal Hierarchies in Music. In *Music Perception*, pages 51–87. Springer New York, New York, NY, June 2010.

[22] C. L. Krumhansl and F. C. Keil. Acquisition of the hierarchy of tonal functions in music. *Memory & Cognition*, 10(3):243–251, 1982.

[23] E. W. Large, J. C. Kim, N. K. Flaig, J. J. Bharucha, and C. L. Krumhansl. A neurodynamic account of musical tonality. *Music Perception: An Interdisciplinary Journal*, 33(3):319–331, 2016.

[24] M. Leman and F. Carreras. The self-organization of stable perceptual maps in a realistic musical environment. In G. Assayah, editor, *Proceedings of the Journées d'Informatique Musicale 1996*, pages 156–169, Caen, 1996. Univ. de Caen – IRCAM, Les Cahiers du GREYC No. 4.

[25] M. Leman. A model of retroactive tone-center perception. *Music Perception*, 12(4):439–471, 1995.

[26] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, 1983.

[27] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1–9, Atlanta, Georgia, USA, 2013.

[28] H. Purwins, B. Blankertz, and K. Obermayer. A new method for tracking modulations in tonal music in audio data format. In *Proceedings of the International Joint Conference on Neural Networks*, volume 6, pages 270–275. IEEE, 2000.

[29] J. R. Saffran, E. K. Johnson, R. N. Aslin, and E. L. Newport. Statistical learning of tone sequences by human infants and adults. *Cognition*, 70(1):27–52, 1999.

[30] M. A. Schmuckler and M. G. Boltz. Harmonic and rhythmic influences on musical expectancy. *Attention, Perception, & Psychophysics*, 56(3):313–325, 1994.

[31] M. A. Schmuckler and M. G. Boltz. Harmonic and rhythmic influences on musical expectancy. *Attention, Perception, & Psychophysics*, 56(3):313–325, 1994.

[32] D. Sears, W. E. Caplin, and S. McAdams. Perceiving the classical cadence. *Music Perception: An Interdisciplinary Journal*, 31(5):397–417, 2014.

[33] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA Neural Networks for Machine Learning*, 2012.

[34] B. Tillmann. Music cognition: Learning, perception, expectations. *Computer music modeling and retrieval. Sense of sounds*, pages 11–33, 2008.

[35] B. Tillmann, J. J. Bharucha, and E. Bigand. Implicit learning of tonality: A self-organizing approach. *Psychological Review*, 107(4):885–913, 2000.

[36] P. Toiviainen and C. L. Krumhansl. Measuring and modeling real-time responses to music: The dynamics of tonality induction. *Perception*, 32(6):741–766, 2003.

[37] G. Widmer. Getting closer to the essence of music: The *Con Espressione* manifesto. *ACM TIST*, 8(2):19:1–19:13, 2017.

[38] Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, and R. Salakhutdinov. On Multiplicative Integration with Recurrent Neural Networks. In *30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016.

FUNCTION- AND RHYTHM-AWARE MELODY HARMONIZATION BASED ON TREE-STRUCTURED PARSING AND SPLIT-MERGE SAMPLING OF CHORD SEQUENCES

Hiroaki Tsushima[1] **Eita Nakamura**[1] **Katsutoshi Itoyama**[1] **Kazuyoshi Yoshii**[1,2]
[1]Graduate School of Informatics, Kyoto University, Japan [2]RIKEN AIP, Japan
{tsushima, enakamura}@sap.ist.i.kyoto-u.ac.jp, {itoyama, yoshii}@kuis.kyoto-u.ac.jp

ABSTRACT

This paper presents an automatic harmonization method that, for a given melody (sequence of musical notes), generates a sequence of chord symbols in the style of existing data. A typical way is to use hidden Markov models (HMMs) that represent chord transitions on a regular grid (e.g., bar or beat grid). This approach, however, cannot explicitly describe the rhythms, harmonic functions (e.g., tonic, dominant, and subdominant), and the hierarchical structure of chords, which are supposedly important in traditional harmony theories. To solve this, we formulate a hierarchical generative model consisting of (1) a probabilistic context-free grammar (PCFG) for chords incorporating their syntactic functions, (2) a metrical Markov model describing chord rhythms, and (3) a Markov model generating melodies conditionally on a chord sequence. To estimate a variable-length chord sequence for a given melody, we iteratively refine the latent tree structure and the chord symbols and rhythms using a Metropolis-Hastings sampler with split-merge operations. Experimental results show that the proposed method outperformed the HMM-based method in terms of predictive abilities.

1. INTRODUCTION

Creation of chord sequences plays a key role in music composition and arrangement since harmony affects the mood of music and characterizes the impression of a certain musical style. Our aim is automatic melody harmonization, or automatic generation of a sequence of chord symbols for a given melody (a sequence of musical notes). In this paper, we restricted our focus to the automatic harmonization in the style of popular music. Instead of manually describing music theories for the style such as jazz and classical music, we take a statistical approach to automatically learn model architectures and parameters from a music corpus and harmonize in the style of that data. We formulate a probabilistic model that represents how likely a chord sequence is to be generated and another model that represents how likely a melody is to be generated conditionally on a chord sequence.

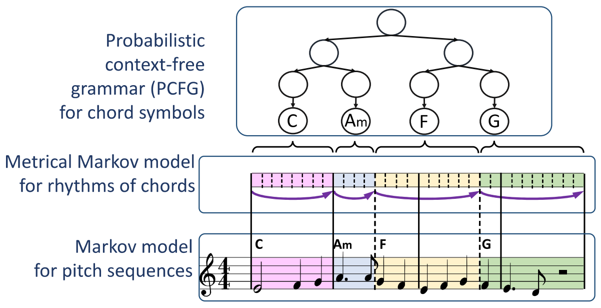

Probabilistic context-free grammar (PCFG) for chord symbols
C
Am
F
G
Metrical Markov model for rhythms of chords
Markov model for pitch sequences

Figure 1: The hierarchical generative model for chord symbol, chord rhythms, and melodies.

Since chord sequences are usually described by Markov models [21, 25], a standard way to statistical harmonization is to use a hidden Markov model (HMM) that has a latent Markov chain of chord symbols and assumes a musical note sequence to be generated conditionally on the chords. This approach, however, does not consider the syntactic roles and hierarchical structure of chords. In traditional harmony theories (e.g., [14, 22]), such syntactic roles are often referred to as *harmonic functions* (e.g., tonic, dominant, and subdominant), which are similar to parts-of-speech in language theories. Another problem of the conventional HMMs lies in the description of the chord rhythms (onset score times or durations of chords). Since chord durations are described by self-transition probabilities on a regular time grid (e.g., beat or bar grid), the chord rhythms are not explicitly described.

To solve these problems, we propose a tree-structured hierarchical generative model that consists of (1) a probabilistic context-free grammar (PCFG) that generates chord symbols, (2) a metrical Markov model that generates chord rhythms, and (3) a Markov model that generates a melody from a chord sequence (Fig. 1). The use of the PCFG was inspired by Steedman's pioneering work [27] that uses a context-free grammar (CFG) for representing the hierarchical structure of chords. A key advantage of our study is that the rule probabilities and tree structure of the PCFG can jointly be estimated in an unsupervised manner from a corpus of chord sequences, expecting that the syntactic roles of chords are captured by the non-terminal symbols.

The metrical Markov model is used for explicitly describing transition probabilities between the onset beat positions of succeeding chords.

Using the tree-structured hierarchical generative model, we propose a statistical harmonization method based on a sophisticated Metropolis-Hasting (MH) sampler with split-merge operations. To estimate a variable-length chord sequence with appropriate chord rhythms for a given melody, we stochastically search for the most likely latent tree structure, symbols, and onset score times of chords from their posterior distributions. In this search, our sampler has four types of proposals: the whole latent tree structure is updated using a variant of the Viterbi algorithm, one of the chords is split or two adjacent chords are merged according to the latent tree structure, and one of the chord onset score time is moved back or forth. Such stochastic global or local updates can be interpreted as a repeated trial-and-error process of finding an optimal chord sequence.

2. RELATED WORK

This section introduces related studies on the automatic harmonization and on the music language model for chords and notes.

2.1 Automatic Harmonization

Some studies on harmonization aim to generate sequences of chord symbols (as in this paper) and other studies aim to generate several (typically four) voices of musical notes. In the former direction, Chuan and Chew [3] proposed a hybrid method that consists of three processes: selection of chord tones (constituent tones of chords) from given melodies with a support vector machine (SVM), construction of triad chords from chord tones, and generation of chord progressions by a rule-base method. Simon et al. [25] developed a method based on HMMs in which chord transitions are described by Markov models. This method has been implemented in a commercial system *MySong*. Raczyński et al. [21] proposed similar Markov models in which chords are conditioned by melodies and time-varying keys. To our knowledge, PCFG has not been used for melody harmonization.

In the latter direction, Ebcioğlu [5] proposed a rule-based method for generating four-part chorales in Bach's style. Methods by using variants of genetic algorithms (GAs) based on music theories have also been studied [18, 19, 28]. Allan and Williams et al. [1] proposed a method based on HMMs which represent chords as hidden states and musical notes as observed outputs. A hidden semi-Markov model (HSMM) [9] has been used for explicitly representing the durations of chords. Paiement et al. [17] proposed a hierarchical tree-structured model that describes chord movements from the viewpoint of hierarchical time scales by dividing the notations of chords.

2.2 Music Language Modeling

Several language models for musical notes have been studied for music structure analysis [10, 11, 13, 15]. According to the *Generative Theory of Tonal Music* (GTTM) [13], a note sequence is assumed to have a hierarchical structure that describes which notes are important. This theory consists of rules for recursively reducing a note sequence into a single note. Computational implementation of GTTM and the analysis of musical pieces using it have been studied [10, 11]. A probabilistic formulation of GTTM based on PCFG has been proposed and enabled unsupervised learning of production rules directly from note sequences [15].

Various language models for chord sequences have been proposed in the context of automatic chord recognition for music audio signals [16, 24, 29], music analysis [23, 27], and music arrangement [6, 20]. The conventional language model for chord sequences is n-gram models [6,24]. To avoid the sparseness problem with a large value of n, smoothing methods have been studied for improving the predictive ability of the language model [2]. Yoshii et al. [29] proposed a vocabulary-free infinity-gram model in which each chord depends on a variable-length history of chords. Paiement et al. [16] introduced several hidden layers of state transitions that represent the hierarchical structure of chords. Some studies attempted to explicitly describe the generative grammar to represent the hierarchical structure of chords [20, 23, 27]. Steedman [27] and Rohrmeier [23] proposed a description of the production rules for chord sequences. A probabilistic extension is later studied in the context of music arrangement and unsupervised learning of the probabilities has been performed [20]. In these studies, the lists of non-terminals and production rules were manually given based on music theories or musical intuition.

3. PROBABILISTIC MODELING

This section explains how to formulate and train our hierarchical generative model of chords and melodies. In our model, the PCFG for chord symbols is trained by unsupervised learning from a corpus of chord sequences while estimating the latent tree structures behind these sequences. The metrical Markov model for chord rhythms is trained by supervised learning from a corpus including chord rhythms. The Markov model for pitch sequences is also trained by supervised learning from paired data of melodies and chord sequences.

3.1 Model Formulation

The PCFG stochastically generates a sequence of chord symbols (or simply *chords* in the following) $\boldsymbol{z} = \{z_n\}_{n=1}^{N}$ and the metrical Markov model generates the corresponding onset score times $\boldsymbol{\phi} = \{\phi_n\}_{n=1}^{N}$ described in units of 16th notes, where N is the number of chords. A subsequence of pitches in the melody $\boldsymbol{x}_n = \{x_{n,i}\}_{i=1}^{I_n}$ in the time span of each chord z_n is then generated, where I_n is the number of pitches in that time span. Concatenating all such subsequences, the whole sequence of pitches $\boldsymbol{x} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ is obtained. $I = \sum_{n=1}^{N} I_n$ denotes the number of melody notes. Let $\psi_{n,i}$ be the onset score time of the melody note corresponding to $x_{n,i}$ and let $\boldsymbol{\psi} = \{\{\psi_{n,i}\}_{i=1}^{I_n}\}_{n=1}^{N}$. ϕ_n and $\psi_{n,i}$ can take integer values from 0 to $16L - 1$, where L is the total number of measures in the whole melody. Although in the training phase, we have multiple sequences of chords sequences and melodies, we formulate here the case of a single sequence for notational simplicity. Extension for multiple sequences is straightforward.

The PCFG G is defined by

$$G = (V, \Sigma, R, S), \tag{1}$$

where V is a set of non-terminal symbols, which are expected to represent hierarchical structure and syntactic roles of chords, Σ is a set of terminal symbols (chord symbols), R is a set of rule probabilities, and S is a start symbol (a non-terminal symbol located on the root of a syntax tree). Rule probabilities consist of the following three types. $\theta_{A\to BC}$ is the probability that a non-terminal symbol $A \in V$ branches to non-terminal symbols $B \in V$ and $C \in V$. $\eta_{A\to\alpha}$ is the probability that $A \in V$ emits terminal symbol $\alpha \in \Sigma$. Each non-terminal symbol $A \in V$ has a coin-toss probability λ_A that stochastically determines whether A emits (otherwise A branches). These probabilities should be normalized properly as follows:

$$\sum_{B,C\in V} \theta_{A\to BC} = 1, \quad \sum_{\alpha\in\Sigma} \eta_{A\to\alpha} = 1. \tag{2}$$

We define $\boldsymbol{\theta}_A = \{\theta_{A\to BC}\}_{B,C\in V}$, $\boldsymbol{\eta}_A = \{\eta_{A\to\alpha}\}_{\alpha\in\Sigma}$, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_A\}_{A\in V}$, $\boldsymbol{\eta} = \{\boldsymbol{\eta}_A\}_{A\in V}$, etc. Similar notations are used throughout this paper.

The metrical Markov model describes the transition probabilities for chord onset beat positions (16th-note level relative score time in a measure) as

$$p(\phi_n|\phi_{n-1}) = \pi_{\bar{\phi}_{n-1},\bar{\phi}_n}, \tag{3}$$

where $\bar{\phi}_n = \phi_n$ mod 16 and π_{ab} $(0 \le a, b < 16)$ indicates the transition probability from beat position k to beat position l. When $\bar{\phi}_n \le \bar{\phi}_{n-1}$, we interpret that the onset of chord n is in the next measure.

The Markov model is described with the following transition probability:

$$p(x_{n,m}|x_{n,m-1}, z_n) = \tau^{z_n}_{x_{n,m-1},x_{n,m}}, \tag{4}$$

where $\tau^{z_n}_{x_{n,m-1},x_{n,m}}$ is the transition probability from pitch $x_{n,m-1}$ to pitch $x_{n,m}$ under chord z_n. In addition, the probability of the first pitch in $\boldsymbol{x}_n$ is given by $p(x_{n,1}|z_n)$.

We put conjugate priors on the parameters of the PCFG as

$$\boldsymbol{\theta}_A \sim \text{Dirichlet}(\boldsymbol{\xi}_A), \tag{5}$$
$$\boldsymbol{\eta}_A \sim \text{Dirichlet}(\boldsymbol{\zeta}_A), \tag{6}$$
$$\lambda_A \sim \text{Beta}(\boldsymbol{\iota}_A), \tag{7}$$

where $\boldsymbol{\xi}_A$, $\boldsymbol{\zeta}_A$ and $\boldsymbol{\iota}_A$ are hyperparameters. Similarly, we put conjugate priors on the parameters of the Markov models as follows:

$$\boldsymbol{\pi}_a \sim \text{Dirichlet}(\boldsymbol{\beta}), \tag{8}$$
$$\boldsymbol{\tau}^z_x \sim \text{Dirichlet}(\boldsymbol{\gamma}), \tag{9}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are hyperparameters.

To complete the generative model of chords and melodies, we need to specify a model generating $\boldsymbol{\psi}$. This model can be formulated as for the model of $\boldsymbol{\phi}$ and we omit the details here since melodies are given as inputs for our harmonization problem.

3.2 Bayesian Learning

We obtain the model parameters $\Theta = \{\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{\pi}, \boldsymbol{\tau}\}$ by the maximum posterior (MAP) estimation. To estimate the parameters $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\lambda}$ of the PCFG, we use a variant of Gibbs sampling called the inside-filtering-outside-sampling algorithm [12]. We assume that a chord sequence $\boldsymbol{z}$ was derived from a latent syntactic tree $\boldsymbol{t}$. $\boldsymbol{t}$ can be represented by a set of non-terminal nodes $\{t_{n:m}\}_{1\le n\le m\le N}$, where $t_{n:m}$ is the root node of a subtree that derives a subsequence of chords $z_{n:m} = \{z_n, z_{n+1}, \cdots, z_m\}$. The latent tree $\boldsymbol{t}$ and the parameters $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\lambda}$ are alternately sampled from the *conditional* posterior distributions $p(\boldsymbol{t}|\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{z})$ and $p(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}|\boldsymbol{t}, \boldsymbol{z})$. This algorithm is proven to yield samples of $\boldsymbol{t}$, $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, $\boldsymbol{\lambda}$ following the true posterior distribution $p(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{t}|\boldsymbol{z})$.

In the inside filtering step, we focus on the conditional probability (*inside probability*) that a subsequence $z_{n:m}$ is derived from a subtree whose root node is A

$$p^A_{n,m} = p(z_{n:m}|t_{n:m} = A). \tag{10}$$

This probability can be calculated recursively from the leaf nodes to the root node as follows:

$$p^A_{n,n} = \lambda_A \eta_{A\to z_n}, \tag{11}$$
$$p^A_{n,n+k} = \sum_{B,C\in V} \left[(1-\lambda_A)\theta_{A\to BC} \sum_{1\le l\le k} p^B_{n,n+l-1} p^C_{n+l,n+k} \right].$$

In the outside sampling step, we recursively sample a latent tree $\boldsymbol{t}$ from a start symbol S to the leaf nodes according to $p(\boldsymbol{t}|\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \boldsymbol{z})$ by using the inside probabilities. When a node $t_{n:n+k} = A$ is already sampled, the two non-terminal symbols B and C into which $t_{n:n+k}$ branches are sampled as follows:

$$\begin{aligned} &p(l, B, C) \\ &= p(t_{n:n+l-1} = B, t_{n+l:n+k} = C \mid t_{n:n+k} = A, z_{n:n+k}) \\ &= (1-\lambda_A)\theta_{A\to BC}\, p^B_{n,n+l-1}\, p^C_{n+l,n+k} / p^A_{n,n+k}, \end{aligned} \tag{12}$$

where $1 \le l \le k$ indicates a split position.

Finally, we sample parameters $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\lambda}$ according to $p(\boldsymbol{\theta}, \boldsymbol{\eta}, \boldsymbol{\lambda}|\boldsymbol{t}, \boldsymbol{z}) = p(\boldsymbol{\theta}|\boldsymbol{t}, \boldsymbol{z})p(\boldsymbol{\eta}|\boldsymbol{t}, \boldsymbol{z})p(\boldsymbol{\lambda}|\boldsymbol{t}, \boldsymbol{z})$ given by

$$\boldsymbol{\theta}_A \sim \text{Dirichlet}(\boldsymbol{\xi}_A + \boldsymbol{u}_A), \tag{13}$$
$$\boldsymbol{\eta}_A \sim \text{Dirichlet}(\boldsymbol{\zeta}_A + \boldsymbol{v}_A), \tag{14}$$
$$\lambda_A \sim \text{Beta}(\boldsymbol{\iota}_A + \boldsymbol{w}_A), \tag{15}$$

where $u_{A\to BC}$ $(v_{A\to\alpha})$ is the number of times the binary production rule $\theta_{A\to BC}$ (the emission rule $\eta_{A\to\alpha}$) is used in $\boldsymbol{t}$, and $w_{A,0}$ $(w_{A,1})$ is the number of times a non-terminal symbol A branches (emits) and $\boldsymbol{t}$.

The parameters $\boldsymbol{\pi}$ and $\boldsymbol{\tau}$ of the Markov models are obtained by supervised learning. Given $\boldsymbol{\phi}$, the posterior distribution of $\boldsymbol{\pi}$ can be calculated easily because of the conjugacy between the Dirichlet and categorical distributions. Similarly, given paired data of $\boldsymbol{z}$, $\boldsymbol{\phi}$, $\boldsymbol{x}$, and $\boldsymbol{\psi}$, the posterior distribution of $\boldsymbol{\tau}$ can be calculated.

4. AUTOMATIC HARMONIZATION

This section explains how to generate sequences of chords for a given melody by using the model in Section 3.

4.1 Problem Specification

Given a melody with pitches $\boldsymbol{x}$ and onset score times $\boldsymbol{\psi}$ and trained model parameters $\boldsymbol{\Theta}$, we aim to estimate a variable-length sequence of chords $\boldsymbol{z}$ and their onset score times $\boldsymbol{\phi}$ that are not restricted to bar lines. Note that the number of chords N is not fixed and should be estimated and that a latent tree $\boldsymbol{t}$ for chords is considered and estimated unlike conventional harmonization methods.

4.2 Metropolis-Hastings Sampling

We propose a Metropolis-Hastings (MH) sampler with split-merge operations for generating samples of $\boldsymbol{t}$, $\boldsymbol{z}$, and $\boldsymbol{\phi}$ from the posterior distribution $p(\boldsymbol{t}, \boldsymbol{z}, \boldsymbol{\phi}|\boldsymbol{x}, \boldsymbol{\psi}, \boldsymbol{\Theta})$ based on the following four types of proposals:

- **Global update:** Update chords $\boldsymbol{z}$ and latent tree $\boldsymbol{t}$ with a Viterbi algorithm for the PCFG, keeping the number and score times of chords unchanged.
- **Split operation:** Randomly choose one of the chords and split it into two adjacent chords.
- **Merge operation:** Randomly choose two adjacent chords in $\boldsymbol{z}$ that form a subtree with two leaves in $\boldsymbol{t}$ and merge them.
- **Rhythm update:** Randomly choose one chord n and move its onset score time ϕ_n back or forth.

Although it is more proper to use the inside-filtering-outside-sampling algorithm for the global update, the Viterbi algorithm is used for efficient optimization in the posterior space.

In the MH sampling, one of these proposals is randomly selected. More specifically, the sampler proposes a sample $s^* = (\boldsymbol{t}, \boldsymbol{z}, \boldsymbol{\phi})^*$ from a current sample $s = (\boldsymbol{t}, \boldsymbol{z}, \boldsymbol{\phi})$. The sampler then judges whether s^* is accepted as the next sample or not according to the acceptance ratio given by

$$g(s^*, s) = \min\left\{1, \frac{p(s^*)p(s|s^*)}{p(s)p(s^*|s)}\right\}, \tag{16}$$

where $p(s)$ is the complete joint likelihood of s based on the proposed model and $p(s^*|s)$ is a proposal distribution that should be set appropriately. If the proposal is rejected, $\boldsymbol{t}$, $\boldsymbol{z}$, and $\boldsymbol{\phi}$ are not updated. In our method, there are three types of proposal distributions for the second to fourth proposals in the above list. We estimate the most plausible $\mathbf{z}$ and ϕ by iterating the MH sampling a sufficient number of times and then getting the latent variables that maximize the likelihood of complete data.

4.3 Updating Chord Symbols

We describe how to update the chord symbols $\boldsymbol{z}$ and the corresponding latent tree $\boldsymbol{t}$ according to the conditional posterior distribution $p(\boldsymbol{t}, \boldsymbol{z}|\boldsymbol{\phi}, \boldsymbol{x}, \boldsymbol{\psi}, \boldsymbol{\Theta})$.

4.3.1 Viterbi Algorithm

Given a melody with pitches $\boldsymbol{x}$ and onset score times $\boldsymbol{\psi}$, we can efficiently sample a sequence of chord symbols $\boldsymbol{z}$ and the corresponding latent tree $\boldsymbol{t}$ by using the Viterbi algorithm. Our algorithm differs from a standard Viterbi algorithm used for estimating $\boldsymbol{t}$ for a given $\boldsymbol{z}$ because both $\boldsymbol{t}$ and $\boldsymbol{z}$ are the latent variables to be estimated in this paper.

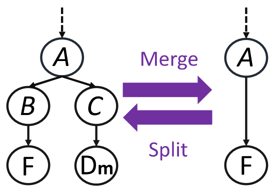


Figure 2: The split-merge operations of the MH sampling.

We first recursively calculate the inside probabilities from the layer of terminal symbols $\boldsymbol{z}$ to the start symbol S according to

$$p^A_{n,n} = \lambda_A \max_{c\in\Sigma} \eta_{A\to c}\, p(\boldsymbol{x}_n|c)^{1/I_n}, \tag{17}$$
$$p^A_{n,n+k} = (1-\lambda_A) \max_{\substack{B,C\in V\\ 1\le l\le k}} \theta_{A\to BC} p^B_{n,n+l-1} p^C_{n+l,n+k},$$

where $p(\boldsymbol{x}_n|c)$ is the probability that a pitch subsequence $\boldsymbol{x}_n$ is generated conditionally on a chord c:

$$p(\boldsymbol{x}_n|c) = p(x_{n,1}|c)\prod_{i=2}^{I_n} p(x_{n,i}|x_{n,i-1}, c). \tag{18}$$

The most likely $\boldsymbol{t}$ and $\boldsymbol{z}$ are obtained by recursively back-tracking the most likely paths from the start symbol S.

4.3.2 Split-Merge Operations

Using the MH sampler, we can split a chord or merge adjacent chords by considering the underlying tree $\boldsymbol{t}$ and the emission probability of the melody. Note the split and merge operations are inverse to each other and that the latent tree $\boldsymbol{t}$ is locally updated by these operations (Fig. 2).

In the split operation, a new sample s^* is proposed by stochastically selecting a chord z_n from $\boldsymbol{z}$, splitting z_n into z^L and z^R, selecting the new onset score time $\phi^* \in (\phi_n, \phi_{n+1}) = [\phi_n + 1, \phi_{n+1} - 1]$, and splitting the non-terminal symbol $t_{n:n}$ into two non-terminal symbols t^L and t^R. The proposal distribution $p(s^*|s)$ is thus

$$p(s^*|s) = \begin{cases} \frac{\theta_{t_{n:n}\to t^L t^R}\eta_{t^L\to z^L}\eta_{t^R\to z^R}}{N(\phi_{n+1}-\phi_n-1)}, & \phi_{n+1}\ge\phi_n+1; \\ 0, & \text{otherwise}. \end{cases} \tag{19}$$

The reverse proposal distribution $p(s|s^*)$, on the other hand, is same as the proposal distribution for the merge operation in which a sample s is proposed by stochastically selecting a pair of adjacent chords z^L and z^R, merging those chords into z_n, by selecting a chord z_n according to the probability $\eta_{t_{n:n}\to z_n}$. Thus we have

$$p(s|s^*) = \frac{\eta_{t_{n:n}\to z_n}}{\#\text{MergeableNodes}(s^*)}, \tag{20}$$

where #MergeableNodes(s^*) is the number of pairs of adjacent chords that can be merged in s^*, i.e., those chords forming a subtree with two leaves.

The likelihood ratio of $p(s^*)$ to $p(s)$ is then given by

$$\frac{p(s^*)}{p(s)} = \frac{(1-\lambda_{t_{n:n}})\theta_{t_{n:n}\to t^L t^R}\lambda_{t^L}\eta_{t^L\to z^L}\lambda_{t^R}\eta_{t^R\to z^R}}{\lambda_{t_{n:n}}\,\eta_{t_{n:n}\to z_n}} \cdot \frac{p(\boldsymbol{x}^L|z^L)p(\boldsymbol{x}^R|z^R)p(\phi^*|\phi_n)p(\phi_{n+1}|\phi^*)}{p(\boldsymbol{x}_n|z_n)p(\phi_{n+1}|\phi_n)}, \tag{21}$$

where $\boldsymbol{x}^L$ and $\boldsymbol{x}^R$ are the subsequences of pitches obtained by splitting $\boldsymbol{x}_n$ at the score time ϕ^*. Using Eqs. (19), (20), and (21), we can calculate the acceptance ratio of s^* according to Eq. (16).

In the merge operation, on the other hand, a new sample s^* is proposed in a similar way to the split operation. More specifically, the acceptance ratio of s^* given by Eq. (16) can be calculate by exchanging s and s^* in Eqs. (19), (20), and (21). Through the split-merge operations, the number of chords N is optimized stochastically.

4.4 Updating Chord Rhythms

We describe how to update the chord rhythms $\boldsymbol{\phi}$ according to the conditional posterior distribution $p(\boldsymbol{\phi}|\boldsymbol{t},\boldsymbol{z},\boldsymbol{x},\boldsymbol{\psi},\boldsymbol{\Theta})$. A new sample s^* is proposed by stochastically selecting a chord n and moving ϕ_n to a new score time $\phi_n^* \in (\phi_{n-1},\phi_{n+1})$. The proposal distribution $p(s^*|s)$ and the reverse proposal distribution $p(s|s^*)$ are given by

$$p(s^*|s) = p(s|s^*) = \frac{1}{N-1}\frac{1}{\phi_{n+1}-\phi_{n-1}-1}. \tag{22}$$

The likelihood ratio of $p(s^*)$ to $p(s)$ is given by

$$\frac{p(s^*)}{p(s)} = \frac{p(\boldsymbol{x}^*_{n-1}|z_{n-1})p(\boldsymbol{x}^*_n|z_n)p(\phi^*_n|\phi_{n-1})q(\phi_{n+1}|\phi^*_n)}{p(\boldsymbol{x}_{n-1}|z_{n-1})p(\boldsymbol{x}_n|z_n)p(\phi_n|\phi_{n-1})p(\phi_{n+1}|\phi_n)}, \tag{23}$$

where $\boldsymbol{x}^*_{n-1}$ and $\boldsymbol{x}^*_n$ are the subsequences of pitches in the time spans of chords $n-1$ and n with the new onset score time ϕ^*_n. Using Eqs. (22) and (23), we can calculate the acceptance ratio of s^* according to Eq. (16).

5. EVALUATION

In this section, we report two experiments conducted to quantitatively evaluate the proposed generative model and the proposed method of automatic harmonization based on the model and discuss examples of chord sequences generated by the method.

5.1 Experimental Conditions

To learn the PCFG unsupervisedly, we extracted 1002 chord sequences corresponding to sections (e.g., verse, bridge, and chorus) from 468 pieces of popular music included in the SALAMI dataset [26]. Only those sequences with a length between 8 and 32 were chosen. The vocabulary of chord symbols was given by the combinations of 12 root notes {C, C#, D, ..., B} and 2 chord types {major, minor}, and a special "other". The values of the hyperparameters were all set to 0.1.

To train the two Markov models in a supervised manner, we extracted 9902 pairs of melodies and the corresponding chord sequences from 194 pieces of popular music included in Rock Corpus [4]. The values of the hyperparameters were all set to 0.1.

In the testing phase, we extracted 339 pairs of melodies and the corresponding chord sequences as ground-truth data for evaluation from 69 pieces of popular music included in the RWC music database [7, 8]. Note that all the data (SALAMI, Rock Corpus, and RWC) were transposed to C major or C minor.

5.2 Evaluating Ability of Melody Prediction

To evaluate the hierarchical generative model based on the PCFG in terms of the ability of melody prediction, we calculated the marginal likelihood for the melodies extracted from the RWC music database. The number of kinds of non-terminal symbols, or the complexity of the PCFG, K was changed from 1 to 20. In each of cases for K, we obtained different PCFG's parameters with Gibbs sampling and calculated the marginal likelihood for each parameter set. The number of different parameter sets were between 37 and 50 depending on the computational complexity. We assumed that the chord onsets were completely synchronized with bar lines such that the chord sequences were marginalized analytically. The proposed model was compared with an HMM that learns the chord-symbol transition between adjacent units which were either musical notes or measures. When minimum time units were musical notes, each note was assumed to be generated conditionally on the chord symbol at the time. When minimum time units were musical measures, notes accompanying each chord were assumed to be generated according to the probability described in Eq. (4).

The marginal likelihood of the trained model parameters $\boldsymbol{\Theta}$ for an unseen melody $\boldsymbol{X}$ with $\boldsymbol{\psi}$ can be calculated with the inside algorithm in Section 3.2. To sum over all possibilities of a latent chord sequence $\boldsymbol{Z}$ and a latent tree $\boldsymbol{T}$, $p^A_{i,i}$ in Eq. (11) is replaced with

$$p^A_{i,i} = \lambda_A \sum_{c\in\Sigma}\eta_{A\to c}\,p(X_i|c), \tag{24}$$

where $p(X_i|c)$ is given by Eq. (18). The average marginal likelihood $\mathcal{L}$ per note for the melody is given by

$$\mathcal{L} = \frac{1}{I}\log p(\boldsymbol{X}|\boldsymbol{\psi},\boldsymbol{\Theta}) = \frac{1}{I}\log p^S_{0,N-1}, \tag{25}$$

where I (N) is the number of notes (chords).

The experimental results are shown in Fig. 3. The proposed model outperformed the HMM, whether the minimum time unit is a musical note ($\mathcal{L} = -3.2813$) or a measure ($\mathcal{L} = -2.3218$). The likelihood tended to decrease as the value of K increased to eight, and the likelihood tended to increase as the value of K increased beyond eight.

5.3 Evaluating Predictive Ability of Chord Sequences

To evaluate the proposed harmonization method in terms of the predictive ability of unseen chord sequences, we generated chord sequences for the melodies of the RWC

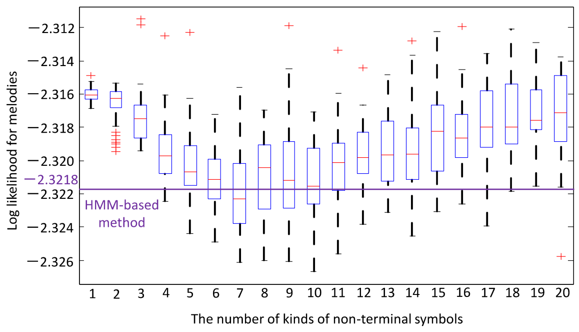


Figure 3: Marginal likelihood for melodies per note. In the box plots, the red line, the black cross and the red crosses indicate the median, the mean and outliers, respectively.

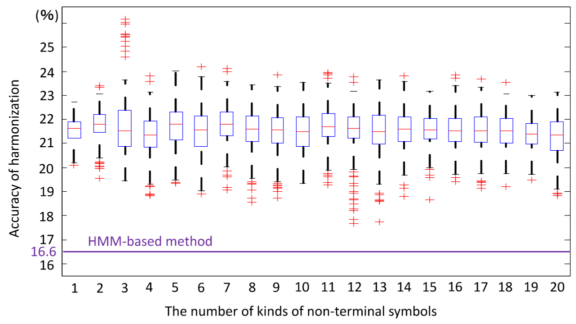


Figure 4: Accuracy of harmonization per note. Indicators in the box plots are the same as those in Fig. 3

music database and calculated the accuracy at a 16th-note level compared with the ground-truth. The complexity of the PCFG, K was changed from 1 to 20. The proposed method was compared with a conventional HMM-based method that represents chord transitions on a 16th-note-level grid.

The experimental result are shown in Fig. 4. The proposed model clearly outperformed the HMM-based method with an accuracy of 16.6 %. While a certain range of K showed much higher accuracy (e.g., 26 %) than the HMM-based method, there was little correlation between K and the median values of accuracies.

5.4 Generated Example and Discussion

Fig. 5 shows how the proposed MH sampling method with split-merge operations worked for automatic harmonization[1] . The number of kinds of non-terminal symbols, K, was set to 12. The chord sequence at the top shows an initial sample in which the chord symbols were optimized by the Viterbi algorithm, but the chord onsets were located at the bar lines. The second chord sequence shows a sample proposed by moving the onset positions of 5th and 6th chords (G major and C major). The third chord sequence shows a sample proposed by merging the 7th and 8th chords (F major and C major) into one chord (C major). The bottom chord sequence shows the best sample that maximizes the likelihood for the given melody. In each of the processes, the likelihood increased. The result indicates that the proposed method can successfully generate a variable-length sequence of chords by considering the latent tree structure behind the chord sequence.

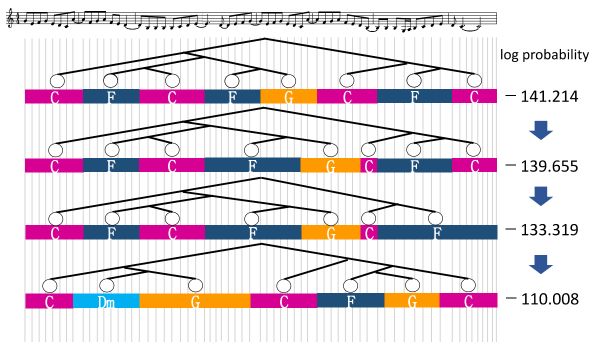


Figure 5: Sampling-based estimation of the most likely chord sequence for a given melody.

We found some problems to be tackled in the future. The proposed method tended to generate simple chords (e.g., C major and A minor). This is because the chord symbols were refined by using the Viterbi algorithm. In addition, the number of the most plausible chord sequences selected in our experiment was rarely more than those initialized at the beginning of sampling. This is because the proposals of the split operation were accepted less frequently than the proposals of the merge operation.

6. CONCLUSION

This paper presented an automatic harmonization method that generates a variable-length chord sequence for a given melody based on music rules hidden in corpora of popular music. The experimental results showed that the proposed model outperformed the HMM-based method in terms of predictive ability and has a large potential for statistical music composition or arrangement.

Since our method is based on statistical learning, it was found to prefer simpler and basic chord sequences. More specifically, the number of generated chords tends to be less than the number of measures. This problem could be solved by giving more chances to the split operation in MCMC sampling. To increase the diversity of generated chord symbols, a sampling or beam-search method is considered to be effective instead of the Viterbi algorithm that tends to find a popular chord sequence that has the highest posterior probability from the statistical viewpoint.

We still need further studies on our model. In this paper, one measure with the time signature of 4/4 is divided into 16 time units. It is therefore important to investigate the best time resolution and extend the model to deal with other kinds of time signatures. In addition, to evaluate the musical appropriateness of generated chord sequences, we plan to conduct a subjective listening test and evaluate how consistent our model is with music theories or musical intuition.

Acknowledgement: This study was partially supported by JSPS KAKENHI Grant Numbers 26700020, 16H01744, and 16J05486 and JST ACCEL No. JPMJAC1602.

[1] Some chord sequences generated in this experimental are available online: http://sap.ist.i.kyoto-u.ac.jp/members/tsushima/ismir2017/

7. REFERENCES

[1] M. Allan and C. Williams. Harmonising chorales by probabilistic inference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 25–32, 2005.

[2] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Association for Computational Linguistics (ACL)*, pages 310–318, 1996.

[3] 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*, pages 57–64, 2007.

[4] T. D. Clercq and D. Temperley. A corpus analysis of rock harmony. *Popular Music*, 30(01):47–70, 2011.

[5] K. Ebcioğlu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51, 1988.

[6] S. Fukayama, K. Yoshii, and M. Goto. Chord-sequence-factory: A chord arrangement system modifying factorized chord sequence probabilities. In *ISMIR*, 2013.

[7] M. Goto. Aist annotation for the RWC music database. In *ISMIR*, pages 359–360, 2006.

[8] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical and jazz music databases. In *ISMIR*, volume 2, pages 287–288, 2002.

[9] R. Groves. Automatic harmonization using a hidden semi-Markov model. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (Boston, MA)*, pages 48–54, 2013.

[10] M. Hamanaka, K. Hirata, and S. Tojo. Implementing 'A generative theory of tonal music'. *Journal of New Music Research*, 35(4):249–277, 2006.

[11] M. Hamanaka, K. Hirata, and S. Tojo. Musical structural analysis database based on GTTM. In *ISMIR*, pages 325–330, 2014.

[12] M. Johnson, T. L. Griffiths, and S. Goldwater. Bayesian inference for PCFGs via Markov chain Monte Carlo. In *North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT)*, pages 139–146, 2007.

[13] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music*. MIT press, 1985.

[14] W. Maler. *Beitrag zur Durmolltonalen Harmonielehre I (7th ed.)*. F. E. C. Leuckart, 2007.

[15] E. Nakamura, M. Hamanaka, K. Hirata, and K. Yoshii. Tree-structured probabilistic model of monophonic written music based on the generative theory of tonal music. In *IEEE ICASSP*, pages 276–280, 2016.

[16] J. F. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. In *ISMIR*, pages 312–319, 2005.

[17] J. F. Paiement, D. Eck, and S. Bengio. Probabilistic melodic harmonization. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 218–229, 2006.

[18] G. Papadopoulos and G. Wiggins. AI methods for algorithmic composition: A survey, a critical view and future prospects. In *AISB Symposium on Musical Creativity*, pages 110–117, 1999.

[19] R. D. Prisco and R. Zaccagnino. An evolutionary music composer algorithm for bass harmonization. *Applications of Evolutionary Computing*, pages 567–572, 2009.

[20] D. Quick. Learning production probabilities for musical grammars. *Journal of New Music Research*, 45(4):295–313, 2016.

[21] S. A. Raczyński, S. Fukayama, and E. Vincent. Melody harmonization with interpolated probabilistic models. *Journal of New Music Research*, 42(3):223–235, 2013.

[22] H. Riemann. *Harmony Simplified: or, The Theory of the Tonal Functions of Chords*. Augener, 1896.

[23] M. Rohrmeier. Mathematical and computational approaches to music theory, analysis, composition and performance. *Journal of Mathematics and Music*, 5(1):35–53, 2011.

[24] R. Scholz, E. Vincent, and F. Bimbot. Robust modeling of musical chord sequences using probablistic N-grams. In *IEEE ICASSP*, pages 53–56, 2009.

[25] 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*, pages 725–734. ACM, 2008.

[26] 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*, pages 555–560, 2011.

[27] M. J. Steedman. A generative grammar for jazz chord sequence. *Music Perception*, 2(1):52–77, 1984.

[28] M. Towsey, A. Brown, S. Wright, and J. Diederich. Towards melodic extension using genetic algorithms. *Educational Technology & Society*, 4(2):54–65, 2001.

[29] K. Yoshii and M. Goto. A vocabulary-free infinity-gram model for nonparametric bayesian chord progression analysis. In *ISMIR*, pages 645–650, 2011.

HIGH-LEVEL MUSIC DESCRIPTOR EXTRACTION ALGORITHM BASED ON COMBINATION OF MULTI-CHANNEL CNNS AND LSTM

Ning Chen
East China University of Science and Technology
nchen@ecust.edu.cn

Shijun Wang
East China University of Science and Technology
sqwsj@hotmail.com

ABSTRACT

Although Convolutional Neural Networks (CNNs) and Long Short Term Memory (LSTM) have yielded impressive performances in a variety of Music Information Retrieval (MIR) tasks, the complementarity among the CNNs of different architectures and that between CNNs and LSTM are seldom considered. In this paper, multi-channel CNNs with different architectures and LSTM are combined into one unified architecture (Multi-Channel Convolutional LSTM, MCCLSTM) to extract high-level music descriptors. First, three channels of CNNs with different shapes of filter are applied on each spectrogram image chunk to extract the pitch-, tempo-, and bass-relevant descriptors, respectively. Then, the outputs of each CNNs channel are concatenated and then passed through a fully connected layer to obtain the fused descriptor. Finally, LSTM is applied on the fused descriptor sequence of the whole track to extract its long-term structure property to obtain the high-level descriptor. To prove the efficiency of the MCCLSTM model, the obtained high-level music descriptor is applied to the music genre classification and emotion prediction task. Experimental results demonstrate that, when compared with the hand-crafted schemes or conventional deep learning (Multi Layer Perceptrons (MLP), CNNs, and LSTM) based ones, MCCLSTM achieves higher prediction accuracy on three music collections with different kinds of semantic tags.

1. INTRODUCTION

The amount of online music tracks is constantly growing, which makes it difficult to tag them manually. Without accurate labels, most of the tracks cannot be accessed. So, auto-tagging technique has become a hot topic in the field of Music Information Retrieval (MIR) for the past two decades. It can be used in music classification, music retrieval, and music recommendation systems.

In the past ten years, deep learning models, such as Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs), and Long Short Term Memory (LSTM) [9] have achieved tremendous success for a variety of MIR tasks, such as onset detection [20], emotion recognition [13], chord estimation [5], rhythm stimuli recognition [22], auto-tagging [3,16], source separation [10], or music recommendation [25], etc. It has been proved that deep learning based models are superior to hand-crafted ones in music content analysis because [16]: i) The nonlinear mapping in deep learning model (e.g. CNNs) is suitable for describing the time-varying nonlinear property of music signal. ii) The hierarchical architecture of deep learning model is fit for representing the hierarchical nature of music in both time domain (onset, rhythm) and frequency domain (note, chord) [16]. iii) Long-term dependencies property of music (music structure or recurrent harmonies), which is important for human music perception and understanding, can be modeled by deep learning model (e.g. LSTM) very well [11].

Despite the rich potential of CNNs and LSTM in describing music properties, they are individually limited in their modeling capability [19]. In [16], the CNNs were adopted to learn high-level descriptor from the spectrogram image of the music signal, and the filter shape of CNNs was studied to make it suitable for representing different music relevant descriptors. It was verified that wider filters and higher filters may be capable of learning longer temporal dependencies and more spread timbral features, respectively. This scheme achieved competitive results in auto-tagging on the Ballroom dataset [8]. However, as shown in [16, 19], CNNs may only model the local context, such as instrument's timbre or musical units, well, but not the long-term dependencies, such as music structure or recurrent harmonies, of the music. As for the LSTMs based schemes, their main issues are two aspects. On the one hand, the temporal modeling is usually done on the low-level descriptor, which makes it difficult to disentangle underlying factors of variation within the input [12]. On the other hand, as shown in [15], there is no intermediate nonlinear hidden layer in LSTM, so the history of previous inputs cannot be summarized efficiently.

To take advantage of the complementarity between CNNs and LSTM, some researchers proposed to combine them in a unified architecture [2, 4]. In [2], LSTM and CNNs are combined in parallel to exploit sequential correlation and local spectro-temporal information. Then, the outputs of the CNNs and LSTM are combined by the fully connected layers to obtain the fused descriptor. Experimental results demonstrated that this scheme outperformed the conventional DNNs, CNNs, or LSTM based ones in acoustic scene classification task. In [19], considering that: "CNNs are good at reducing frequency variations, LSTMs are good at temporal modeling, and DNNs are appropriate for mapping features to a more separable space", the three models are combined into one unified architecture to take advantage of the complementarity among them. This scheme achieved better performances than the LSTM based one in the voice search task.

In this paper, a new deep learning based architecture (called Multi-Channel Convolutional LSTM, MCCLSTM) is proposed for high-level music descriptor extraction. MCCLSTM is different from the methods discussed above as it takes advantage of the complementarity among CNNs with different architectures and that between CNNs and LSTM in modeling music properties. Considering that different musical properties correspond to different time-frequency resolutions [16], the descriptor extracted by one CNN with a specific filter may not characterize the music property comprehensively. So, in the proposed scheme, three channels of CNNs with different shapes of filters are adopted to extract pitch-, tempo-, and bass-relevant descriptors from the spectrogram image, respectively. Then, the outputs of each CNNs are concatenated and then passed trough a fully connected layer to map to the fused descriptor. Finally, since it has been verified in [15] that the performance of LSTM can be improved greatly when provided with high-level descriptor, the LSTM is put as a higher level of the fully connected layer in the proposed scheme to learn the time dependency in the fused descriptor sequence to extract the long-term structure of the whole track. Since the obtained high-level descriptor contains both local context based and long-term structure based information of the music, it may describe the music property more comprehensively. Experimental results demonstrate that the proposed model is superior to the hand-crafted schemes [14, 18] and the conventional deep learning (Multi Layer Perceptrons (MLP) [21], CNNs [16], and LSTM) based ones in music auto-tagging task on three music collections with different kinds of semantic tags.

The rest of this paper is organized as follows. The proposed scheme is described in detail in Section 2. The performances of the proposed scheme in music auto-tagging task in comparison with other state-of-the-art schemes are evaluated and discussed in Section 3. Conclusions and prospects on future work are given in Section 4.

2. MCCLSTM MODEL

The MCCLSTM architecture is shown in Figure 1.

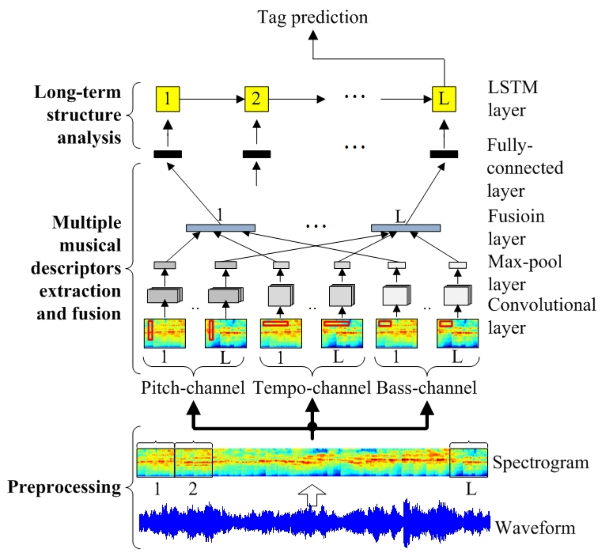


Figure 1: Multi-Channel Convolutional LSTM (MCCLSTM) architecture.

2.1 Preprocessing

The same preprocessing procedure shown in [16] is adopted in the proposed scheme. First, the Short-Time Fourier Transform (STFT) is applied to the input music audio signal, whose sampling rate is 44100 Hz, to obtain the spectrum of it. In STFT, a Blackman-Harris window of 2048 samples is chosen, and the hop size is 1024 samples. Next, the 40-band Mel filter-bank is applied on the obtained spectrum to generate the corresponding spectrogram image of it. Then, the whole spectrogram image is split into L chunks without overlapping. The size of each spectrogram chunk is $M \times N$, where M and N stand for the number of frequency bins and that of frames, respectively.

2.2 Multiple Musical Descriptors Extraction and Fusion

There is no one universal deep learning architecture or hand-crafted scheme that performs well in modeling multiple music properties at the same time. To solve this problem and describe the music content more comprehensively, three channels of CNNs with different shape of filters are adopted and combined in the proposed scheme to obtain fused descriptor, which contains tempo-, pitch-, and bass-relevant information of the input music. This idea was first proposed in [16] to combine the tempo- and pitch-relevant information and was modified in this paper by adding another bass-relevant information.

- Pitch-channel CNNs: in this channel, a $m \times 1 (m \ll M)$ frequency filter is chosen. This type of filter is designed for modeling frequency features. The upper layer can represent some temporal dependencies from the resulting activations as well [16]. In the proposed scheme, this channel of CNNs are responsible for extracting pitch, timbre, or equalization setups relevant descriptor of the music.

- Tempo-channel CNNs: in this channel, a $1 \times n (n \ll N)$ temporal filter is adopted. This kind of filter will be suitable for learning temporal dependencies but not frequency dependencies. Also, the upper layer may exploit the frequency relations [16]. In the proposed scheme, this channel of CNNs try to learn rhythmic/tempo relevant patterns of the music.

- Bass-channel CNNs: in this channel, a $m \times n (m \ll M, n \ll N)$ filter is taken. This type of filter is capable of learning time and frequency features at the same time. Different musical aspects can be learned by such filters with different combination of m and n. Considering that the task of this channel of CNNs is to model the bass- or kick-relevant feature, which most entails finding changes over time, a filter that is wide in time and narrow in frequency (i.e. $m < n$) is adopted [16].

To take advantage of the complementarity among the obtained pitch-, tempo-, and bass-relevant descriptors, they are concatenated on the fusion layer and then passed through a fully connected layer to generate the fused descriptor.

2.3 Long-Term Structure Analysis

Although CNNs may model the local context in the spectrogram chunk well, it may not model the long-term structure (music structure or recurrent harmonies) of the whole track, which is quite important for human music perception and understanding [11]. To solve this problem, a LSTM layer is added on top of the fully connected layer. It will help to learn the time dependence in the fused descriptor sequence of the whole track. The number of the nodes in LSTM is equal to that of the chunks included in the whole track. Since the obtained high-level descriptor contains the information of different music properties (pitch, tempo, and bass) and that of long-term structure of the music as well, it may be more suitable for auto tagging of music with different kinds of semantic tags.

3. EXPERIMENTS

In the experiment, the input of the architecture is 40-dimensional log-mel filterbank based spectrogram images, computed every 3.712s (i.e., $M = 40, N = 80$). As shown in Figure 1, all three CNNs channels are composed of 1 convolutional layer and 1 max-pooling layer. The size of each kernel in the multi-channel convolutional neural network is listed in Table 1. When 2 (or 3) CNNs channels are fused, the fully connected layer and the LSTM layer contain 200 (or 400) and 100 (or 200) units, respectively. The number of nodes in LSTM is equal to that of the chunks included in the whole spectrogram. The weights for all CNN and LSTM layers are randomly initialized to be Gaussian, with a variance of 1. And a softmax layer is added upon the LSTM layer to discriminate the tag of the overall input music. Six kinds of architectures based on different combinations of CNNs and LSTM (see Table 2) are studied in the experiment. To verify the efficiency of the proposed high-level musical descriptor extraction scheme, its performances in music auto-tagging task are tested on three music collections with different kinds of semantic tags, in comparison with those obtained by the hand-crafted schemes or conventional deep learning-based ones.

The whole architecture shown in Figure 1 is trained together with the categorical cross-entropy criterion, using the asynchronous stochastic gradient descent optimization strategy. The weights for all CNN and LSTM layers are randomly initialized to be Gaussian, with a variance of 1. The prediction accuracies obtained by deep learning-based schemes are computed using 10-fold cross validation with a randomly generated train-validation-test split of 80%-10%-10%.

Layer name	P-channel	T-channel	B-channel
Convolutional layer	(32,1)	(1,60)	(13,9)
Max-pooling layer	(1,80)	(40,1)	(4,4)

Table 1: Size of each kernel in the multi-channel convolutional neural network.

ID	Architecture
R0	CNNs (T)
R1	LSTM only
R2 [16]	CNNs (T+P)
R3	CNNs (T+P+B)
R4	CNNs (T+P)+LSTM
MCCLSTM	CNNs (T+P+B)+LSTM

Table 2: Six architectures studied in the experiments.

3.1 Datasets

The following three music collections with different kinds of semantic tags are adopted to test the performances of the proposed scheme in auto-tagging task.

- GTZAN genre collection [24]: Although GTZAN dataset suffers from some repetitions, mislabelings and distortions problems [17], it is often adopted to evaluate genre classification accuracy. This dataset is composed of 1000 audio tracks, each 30 seconds long. It contains 10 genres (blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock), each of which is represented by 100 tracks.

- Ballroom dataset: This dataset comprises 698 audio tracks, each around 30 seconds long. It contains 8 ballroom dancing genres (cha-cha-cha 111, jive 60, quickstep 82, rumba 98, samba 86, tango 86, viennese valtz 65, and slow waltz 110).

- Soundtracks dataset [7] for music and emotion: This dataset contains 470 film music excerpts, each 15-30 seconds long. The tag of each excerpts is one of the five discrete emotions: anger 61, fear 116, sadness 108, happiness 89, and tenderness 96 [6].

3.2 Experimental Results

To verify the efficiency of the MCCLSTM model in music auto-tagging task, its performance, in terms of prediction accuracy, is compared with those obtained by the conventional hand-crafted schemes and other deep learning-based ones (MLP, CNNs, and LSTM) on each of above datasets.

3.2.1 Baselines

In the experiment, as shown in Table 3-5, two hand-crafted schemes [14, 18] and four deep learning-based ones (MLP [21], R0, R1, and CNNs-3) are included as baselines. In R0, the output of the tempo-channel CNNs is used for tag prediction, directly. While, in R1, the LSTM is learned directly on the spectrogram chunk sequence. The CNNs-3 scheme is composed of two convolutional levels, two max-pooling levels, and one fully connected level (200 units). The sizes of the two convolutional layers are (5,5) and (3,3), respectively. The sizes of the two max-pooling layers are both (2,2). The two max-pooling layers are alternated with convolutional layers. It should be noted that, since the codes of the schemes in [14], [18], and [21] are not available, we just include the prediction accuracies obtained by them on specific music collections. As shown in Table 3-5, when compared with the hand-crafted schemes [14, 18], the MCCLSTM scheme can enhance the prediction accuracy from 3.50% to 16.95%. As for the deep learning-based ones (MLP [21], R0, R1, and CNNs-3), MCCLSTM scheme can achieve a higher prediction accuracy of 1.69%-32.99%, 4.90%-27.90%, and 9.85%-22.35%, on GTZAN, Ballroom, and Soundtracks datasets, respectively. So, it is verified that the proposed scheme is superior to the hand-crafted schemes and conventional deep learning-based ones in auto tagging task across the datasets included.

Schemes	Accuracy: mean%$\pm$std
[14]	72.80
[21] (3 layers)	83.00 ± 1.10
CNNs-3	79.80 ± 1.70
R0	51.70 ± 2.60
R1	59.30 ± 2.20
R2 [16]	78.40 ± 1.90
R3	82.90 ± 2.11
R4	83.70 ± 1.10
MCCLSTM	84.69 ± 1.76

Table 3: Performance comparison on GTZAN dataset.

Schemes	Accuracy: mean%$\pm$std
[14]	88.40
CNNs-3	87.00 ± 1.32
R0	81.79 ± 4.72
R1	64.00 ± 1.50
R2 [16]	87.68 ± 4.44
R3	89.45 ± 2.18
R4	90.32 ± 1.12
MCCLSTM	91.90 ± 2.33

Table 4: Performance comparison on Ballroom dataset.

Schemes	Accuracy: mean%$\pm$std
[18]	57.40 ± 5.50
CNNs-3	60.87 ± 3.76
R0	52.00 ± 2.20
R1	64.50 ± 2.00
R2 [16]	57.28 ± 2.85
R3	63.04 ± 2.16
R4	73.70 ± 2.47
MCCLSTM	74.35 ± 1.63

Table 5: Performance comparison on Soundtracks dataset.

3.2.2 Multi-Channel CNNs Based Schemes

In [16] (denoted as R2 in this paper), the outputs of the pitch-channel CNNs and the tempo-channel CNNs in Figure 1 were concatenated to obtain the fused musical descriptor, which then contains both pitch- and tempo-relevant information. To make the fused descriptor contain bass relevant information also, a bass-channel CNNs is added in R2 to obtain the three-channel CNNs based one (denoted as R3). As shown in Table 3-5, R3 achieves higher prediction accuracy than [16] on all three datasets. Especially, for the music mood auto-tagging (see Table 5), R3 enhances the prediction accuracy by 5.76% when compared with [16]. The latent reason may be that the bass relevant information plays a crucial role in mood classification [1].

3.2.3 Multi-Channel CNNs + LSTM Based Schemes

Music can be described as sequences of events that are structured in pitch and time [23]. So, how to learn and represent such complex event sequences (or long-term structure) is very important for music perception and cognition. However, CNNs may only model the local context well but not the long-term dependencies contained in the whole track [16], which will affect the accurate describing of the music properties. Considering that LSTM is good at extracting the sequential information from the consecutive features, a LSTM layer is added on the top of the fully connected layer (as shown in Figure 1) to model the long-term structure property of the music. To show the benefits of LSTM, it is applied on the two-channel and three-channel CNNs fused descriptor, respectively, to construct R4 and MCCLSTM schemes. The experimental results shown in Table 3-5 indicate that for the two-channel CNNs based scheme (R2), the introducing of LSTM layer can help to enhance the prediction accuracy of 5.30%, 2.64%, and 16.42% on GTZAN, Ballroom, and Soundtracks, respectively. For the three-channel CNNs based scheme (R3), the adding of LSTM layer can help to enhance the prediction accuracy of 1.79%, 2.45%, and 11.31% on GTZAN, Ballroom, and Soundtracks, respectively. So, it is verified that adopting LSTM to analyze the time dependencies contained in the fused descriptor sequence further may help to describe the musical characteristic more accurately and comprehensively.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we present a unified deep learning architecture (MCCLSTM) for high-level musical descriptor extraction. First, the CNNs with different resolutions are utilized to analyze each spectrogram chunk of the input music to extract different music property-relevant descriptors, respectively. Then, the outputs of each CNNs channels are concatenated and then passed through a fully connected layer to obtain the fused descriptor. Finally, the LSTM is performed on the fused descriptor sequence to model the long-term structure of the input music. To verify the efficiency of the MCCLSTM scheme, its performance in music auto tagging are compared with those obtained by the hand-crafted schemes and the conventional deep learning (MLP, CNNs, and LSTM) based ones on three music collections with different kinds of semantic tags. Experimental results demonstrate that the proposed scheme is superior to hand-crafted schemes in [14, 18] and other deep learning-based ones ([16, 21], R0, R1, R3, R4, and CNNs-3). However, since the fused descriptor is obtained by concatenating the outputs of each CNNs channel, the complementarity among these three descriptors cannot be fused efficiently. So, our future work is to study new fusion mechanism, which can utilize the common as well as complementary aspects of each musical descriptors more efficiently.

5. ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (No. 61271349).

6. REFERENCES

[1] Jakob Abeßer, Hanna Lukashevich, Christian Dittmar, and Gerald Schuller. Genre classification using bass-related high-level features and playing styles. In *Proceedings of 10th International Conference on Music Information Retrieval (ISMIR 2009)*, pages 453–458, 2009.

[2] Soo Hyun Bae, Inkyu Choi, and Nam Soo Kim. Acoustic scene classification using parallel combination of lstm and cnn. In *2016 Detection and Classification of Acoustic Scenes and Events (DCASE 2016)*, pages 1–5, 2016.

[3] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. Proceedings of 17th International Conference on Music Information Retrieval (ISMIR 2016), 2016.

[4] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[5] Junqi Deng and Yu-Kwong Kwok. A hybrid gaussian-hmm-deep-learning approach for automatic chord estimation with very large vocabulary. Proceedings of 17th International Conference on Music Information Retrieval (ISMIR 2016), 2016.

[6] Tuomas Eerola and Jonna K Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 2010.

[7] Tuomas Eerola and Jonna K Vuoskoski. A comparison of the discrete and dimensional models of emotion in music. *Psychology of Music*, 39(1):18–49, 2011.

[8] Fabien Gouyon, Anssi Klapuri, Simon Dixon, Miguel Alonso, George Tzanetakis, Christian Uhle, and Pedro Cano. An experimental comparison of audio tempo induction algorithms. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1832–1844, 2006.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(12):2136–2147, 2015.

[11] Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

[12] Bernhard Lehner, Gerhard Widmer, and Sebastian Bock. A low-latency, real-time-capable singing voice detection method with lstm recurrent neural networks. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 21–25. IEEE, 2015.

[13] Xinxing Li, Haishu Xianyu, Jiashen Tian, Wenxiao Chen, Fanhang Meng, Mingxing Xu, and Lianhong Cai. A deep bidirectional long short-term memory based multi-scale approach for music dynamic emotion prediction. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 544–548. IEEE, 2016.

[14] Athanasios Lykartsis and Alexander Lerch. Beat histogram features for rhythm-based musical genre classification using multiple novelty functions. In *Proceedings of 16th International Conference on Music Information Retrieval (ISMIR 2015)*, pages 434–440, 2015.

[15] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, 2014.

[16] Jordi Pons, Thomas Lidy, and Xavier Serra. Experimenting with musically motivated convolutional neural networks. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2016.

[17] Francisco Rodrıguez-Algarra, Bob L Sturm, and Hugo Maruri-Aguilar. Analysing scattering-based music content analysis systems: Wheres the music? In *Proceedings of 17th International Conference on Music Information Retrieval (ISMIR 2016)*, 2016.

[18] Pasi Saari, Tuomas Eerola, and Olivier Lartillot. Generalizability and simplicity as criteria in feature selection: Application to mood classification in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(6):1802–1812, 2011.

[19] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

[20] Jan Schlüter and Sebastian Böck. Improved musical onset detection with convolutional neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6979–6983. IEEE, 2014.

[21] Siddharth Sigtia and Simon Dixon. Improved music feature learning with deep neural networks. In *Proceedings of 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6959–6963. IEEE, 2014.

[22] Sebastian Stober, Daniel J Cameron, and Jessica A Grahn. Using convolutional neural networks to recognize rhythm stimuli from electroencephalography recordings. In *Advances in neural information processing systems*, pages 1449–1457, 2014.

[23] Barbara Tillmann. Music and language perception: expectations, structural integration, and cognitive sequencing. *Topics in cognitive science*, 4(4):568–584, 2012.

[24] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[25] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in neural information processing systems*, pages 2643–2651, 2013.

IDENTIFYING RAGA SIMILARITY THROUGH EMBEDDINGS LEARNED FROM COMPOSITIONS' NOTATION

Joe Cheri Ross[1] Abhijit Mishra[3] Kaustuv Kanti Ganguli[2]
Pushpak Bhattacharyya[1] Preeti Rao[2]
[1] Dept. of Computer Science & Engineering, [2] Dept. of Electrical Engineering
Indian Institute of Technology Bombay, India
[3] IBM Research India
joe@cse.iitb.ac.in

ABSTRACT

Identifying similarities between ragas in Hindustani music impacts tasks like music recommendation, music information retrieval and automatic analysis of large-scale musical content. Quantifying raga similarity becomes extremely challenging as it demands assimilation of both *intrinsic* (*viz.*, notes, tempo) and *extrinsic* (*viz.* raga singing-time, emotions conveyed) properties of ragas. This paper introduces novel frameworks for quantifying similarities between ragas based on their melodic attributes alone, available in the form of *bandish* (composition) notation. Based on the hypothesis that *notes in a particular raga are characterized by the company they keep*, we design and train several deep recursive neural network variants with Long Short-term Memory (LSTM) units to learn *distributed representations* of notes in ragas from *bandish* notations. We refer to these distributed representations as *note-embeddings*. Note-embeddings, as we observe, capture a raga's identity, and thus the similarity between note-embeddings signifies the similarity between the ragas. Evaluations with perplexity measure and clustering based method show the performance improvement in identifying similarities using note-embeddings over n-gram and uni-directional LSTM baselines. While our metric may not capture similarity between ragas in their entirety, it could be quite useful in various computational music settings that heavily rely on melodic information.

1. INTRODUCTION

Hindustani music is one of the Indian classical music traditions developed in northern part of India getting influences from the music of Persia and Arabia [17]. The south Indian music tradition is referred to as Carnatic music [30]. The compositions and their performances in both these classical traditions are strictly based on the grammar prescribed by the raga framework. A raga is a melodic mode or tonal matrix providing the grammar for the notes and melodic phrases, but not limiting the improvisatory possibilities in a performance [25].

Raga being one of the most prominent categorization aspect of Hindustani music, identifying similarities between them is of prime importance to many Hindustani music specific tasks like music information retrieval, music recommendation, automatic analysis of large-scale musical content *etc*. Generally similarity between ragas is inferred through attributes associated with the ragas. For instance, in Hindustani music, classification of ragas based on the tonal material involved is termed as *thaat*. There are 10 *thaats* in Hindustani music [8]. *prahar*, *jati*, *vadi*, *samvadi etc.* are the other important attributes. Most of the accepted similarities between ragas encompass the similarities in many of these attributes. But these similarities cannot always be derived exclusively from these attributes. Melodic similarity is a strong substitute and close to perceived similarity. The melodic similarity between Hindustani ragas is not largely available in documented form. This necessitates systems for raga similarity measurement to be devised, even though the number of ragas in the Hindustani classical framework is fixed.

A composed musical piece termed as *bandish* is written to perform in a particular raga, giving ample freedom to the performer to improvise upon. As the literal meaning suggests, *bandish* is tied to its raga, *tala* (rhythm) and lyrics. *Bandish* is taken as the basic framework for a performance which gets enriched with improvisation while the performer renders it. Realization of a *bandish* in a performance brings out all the colors and characteristics of a raga. Given this fact, audio performances of the *bandishes* can be deemed to be excellent sources for analyzing raga similarities from a computational perspective. However, methods for automatic transcription of notations from audio performances have been elusive; this restricts the possibilities of exploiting audio-resources. Our work on raga similarity identification, thus, relies on notations having abstract representation of a performance covering most dimensions of the composition's raga. We use *bandish* notations dataset available from `swarganga.org` [16].

Our proposed approach, based on deep recursive neural network with bi-directional LSTM as recurrent

units, learns note-embeddings for each raga from the *bandish* notations available for that raga. We partition our data by raga and train the model independently for each raga. It produces as many note-embeddings, as many different ragas we have represented in the dataset. The *cosine similarity* between the note-embeddings serves for analyzing the similarity between the ragas. Our evaluations with perplexity measure and clustering based methods show the performance improvement in identifying similarities using note-embeddings using our approach over (a) a baseline that uses n-gram overlaps of notes in *bandish* for raga similarity computation (b) a baseline that uses pitch class distribution (PCD) and (c) our approach with uni-directional LSTM. We believe, our approach can be seamlessly adopted to the Carnatic music style as it follows most of the principles as Hindustani music.

2. RELATED WORK

To the best of our knowledge no such attempts to identify raga similarity have been made so far. The work closest to ours is by Bhattacharjee and Srinivasan [5] who discuss raga identification of Hindustani classical audio performances through a transition probability based approach. Here they also discuss about validating the raga identification method through identifying known raga relationship between 10 ragas considered for this work. A good number of research works have been carried out pertaining to raga identification in Hindustani music using note intonation [3], chromagram patterns [11], note histogram [12]. Pandey *et al.* [22] proposed an HMM based approach on automatically transcribed notation data from audio. There has been quite a few raga recognition attempts in Carnatic music also [28, 4, 27, 24].

3. RAGA SIMILARITY BASED ON NOTATION: MOTIVATION AND CENTRAL IDEA

While the general notion of raga similarity is based on various dimensions of ragas like *thaat*, *prahar*, *jati*, *vadi*, *samvadi etc.*, the similarities perceived by humans (musicians and expert listeners) is predominantly substantiated upon the melodic structure. A raga-similarity method solely based on notational (melodic) information can be quite relevant to computational music tasks involving Indian classical music.

Theoretically, the identity of a raga lies in how certain notes and note sequences (called phrases) are used in its compositions. We hypothesize that capturing the semantic association between different notes appearing in the composition can possibly reveal the identity of a raga. Moreover, it can also provide insights into how similar or dissimilar two ragas can be, based on how similar / dissimilar the semantic associations of notes in the compositions are. We believe , notes for a specific raga can be represented in *distributed forms* (such as vectors), reflecting their semantic association with other notes in the same raga (analogous to words having distributed representations in the domain of computational linguistics [18]). These representations could account for how notes are preceded and succeeded by other notes in compositions.

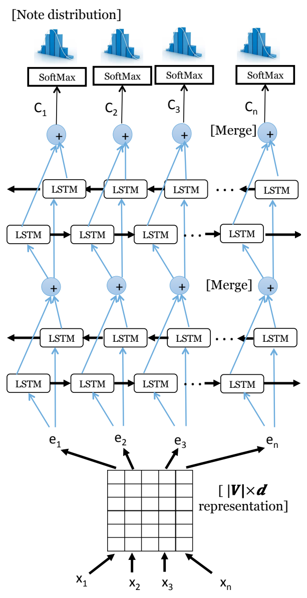


Figure 1. Bi-directional LSTM architecture for learning note-embeddings

Formally, in a composition, a note $x \in V$ (where V represents a vocabulary all notes in three octaves) can be represented as a d dimensional vector that captures semantic-information specific to the raga that the compositions belong to. Such distributed note-representations, referred to as *note-embeddings* ($|V| \times d$ matrix) can be expected to capture more information than other forms of sparse representations (like presenting notes with unique integers). We propose a bi-directional LSTM [14] based architecture that is motivated by the the work of Huang and Wu [15] to learn note-embeddings characterizing a particular style of music. We learn note-embeddings for each raga separately from the compositions available for the raga.

How can note-embeddings help capture similarities between ragas? We hypothesize that embeddings learned for a given note for similar ragas will have more similarity. For example, the representation for note *Ma-elevated* (equivalent note *F#* in C-scale) in raga *Yaman* can be expected to be very similar to that of *Yaman Kalyan* as both of these ragas share very similar melodic characteristics.

4. NEURAL NETWORK ARCHITECTURE FOR LEARNING *NOTE-EMBEDDINGS*

We design a deep recurrent neural network (RNN), with bi-directional LSTMs as recurrent units, that learns to predict the forth-coming notes that are highly likely to appear in a *bandish* composition, given input sequences of notes. This is analogous to neural language models built for speech and text synthesis [19]. While our network tries to achieve this objective, it learns distributed note representations by regularly updating the note-embedding matrix. The choice of this architecture is due to the facts that

(a) for sequence learning problems like ours, RNNs with LSTM blocks have proven useful [29, 13], and (b) in Hindustani music a note rendered at a moment has dependence on patterns preceding and succeeding it, motivating us to use bi-directional LSTM.

The model architecture is shown in Figure 1. Supposing that a sequence in a composition has n notes (n to be kept constant by padding wherever necessary), denoted as $x_1, x_2, x_3, ..., x_n$, where $\forall i \in n, x_i \in V$. The note x_i can be represented in *one-hot* format, with the j^{th} component of a $|V|$ dimensional zero-vector set to 1, if x_i is the j^{th} element of vocabulary V. Each note is input to a *note-embedding* layer W of dimension $|V| \times d$ where d is the note-embedding dimension. The output of this layer is a sequence of embeddings e_i of dimension d, obtained by performing a matrix multiplication between x_i with W. The embedding sequences $e_1, e_2, e_3, ..., e_n$ are input to two layers of bi-directional LSTMs.

For each time-step ($i \in n$), the context-representations learned by the outer-bidirectional LSTM layer (C_i) is passed through a `softmax` layer that computes the conditional probability distribution of all possible notes given the context representations given by LSTM layers.

For each time-step, the prediction of the forthcoming note in the sequence is done by choosing the note that maximizes the likelihood given the context i.e.

$$\hat{x} = \underset{j \in |V|}{argmax} \quad P(x_{i+1} = v_j | C_i) \tag{1}$$

where C_i is the merged context representations learned by the forward and backward sequences in the bi-directional LSTM layers. Probability of a note at a time-step is computed by the `softmax` function as,

$$P(x_{i+1} = v_j | C_i) = \frac{exp({U_j}^T C_i + b_j)}{\sum_{k=1}^{|V|} exp({U_k}^T C_i + b_k)} \tag{2}$$

where U is the weight matrix in the `softmax` layer and b_j is bias term corresponding to note v_j.

The embedding layer is initialized randomly and during training, errors (in terms of cross-entropy) are back propagated upto the embedding layer, resulting in the updation of the embedding-matrix. Cross-entropy is computed as,

$$\frac{1}{M \times T} \sum_{i=1}^{M} \sum_{t=1}^{T} cross\_entropy(y_t^i, \hat{y}_t^i) \tag{3}$$

$$cross\_entropy(y, \hat{y}) = - \sum_{p=1}^{|V|} y_p \; log \; \hat{y}_p \tag{4}$$

Where M is the number of note sequences in a raga and T is the sequence length. y_t^i denotes the expected distribution of i^{th} note sequence at time-step t (bit corresponding to the expected note set to 1 and rest to 0s) and $\hat{y}_t^i$ denotes the predicted distribution. Since our main objective is to learn semantic representation of notes through note-embeddings (and not predict note sequences), we do not heavily regularize our system. Moreover, our network design is inspired by Mikolov *et al.* [18], who also do not heavily regularize their system while learning word-embeddings.

4.1 Raga Similarities from Note-embeddings

For each raga our network learns a $|V| \times d$ matrix representing $|V|$ note-embeddings. We compute (dis)similarity between two ragas by computing *pairwise cosine distance* between embedding vectors of every note in V and then averaging over all notes. This is based on the assumption that distributed representations of notes (as captured by the embeddings) will be similar across ragas that are similar. The choice of cosine similarity (or cosine distance) for computing the similarity between the note-embeddings is driven by its robustness as a measure of vector similarity for vectors and its predominant usage for measuring word embedding similarity [20]. Appropriate distance measures have been adopted for non-LSTM based baselines.

5. BASELINES FOR COMPARISON

To confirm the validity, we compare our approach with a few baseline approaches.

5.1 N-gram Based Approach

The N-gram based baseline creates an n-gram profile based on the count of each n-gram from the available compositions in a raga. We compute the n-gram for n ranging from 1 to 4. The distance between two ragas is computed using the out-of-place measure described in Cavnar *et al.* [7]. Out-of-place measure depends on the rank order statistics of the two profiles. It computes how far 2 profiles are out-of-place *w.r.t* the n-gram rank order statistics. The distance is taken as the l_2 norm of all the n-gram rank differences, normalized by the number of n-grams. Intuitively, the more similar two ragas are, more would the N-gram profiles overlap, reducing the l_2 norm.

5.2 Pitch Class Distribution (PCD)

This method computes the distribution of notes from the count of notes in a raga's *bandish* dataset. 36 notes(across 3 octaves) are considered separately for computing PCD. As the method describes, sequence information is not captured here. The similarity distance between two ragas is computed by taking the *euclidean distance* between the corresponding pitch class distributions; the assumption is that each pitch class two similar ragas will share similar probability value, thereby reducing the euclidean distance. For the raga recognition task by Chordia *et al.* [9], euclidean distance is used for computing the distance between pitch class distributions in one of their approaches. This baseline is to verify the relevance of sequence information in capturing raga similarity.

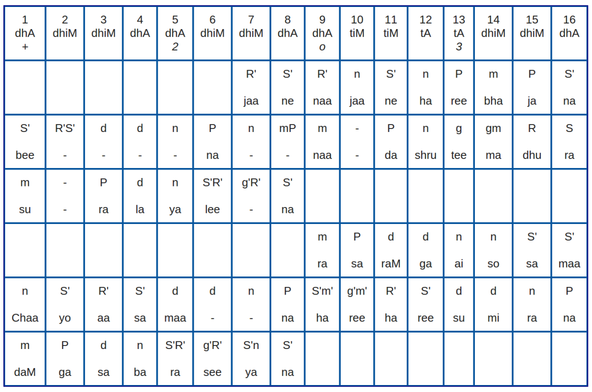
5.3 Uni-directional LSTM

The effectiveness of a bi-directional LSTM for modeling Hindustani music is verified with this baseline. The architecture is same as described in Figure 1, except for the replacement of bi-directional LSTMs with uni-directional LSTMs. Since there is only forward pass in uni-directional

LSTM, the merge operation in bi-directional LSTM design is not required here.

6. DATASET

Our experiments are carried out with the Hindustani *bandish* dataset available from `swarganga.org`, created by Swarganga music foundation. This website is intended to support beginners in Hindustani music. This has a large collection of Hindustani *bandishes*, with lyrics, notation, audio and information on raga, *tala* and *laya*. Figure 2



1 dhA +	2 dhiM	3 dhiM	4 dhA	5 dhA *2*	6 dhiM	7 dhiM	8 dhA	9 dhA *o*	10 tiM	11 tiM	12 tA	13 tA *3*	14 dhiM	15 dhiM	16 dhA
						R' jaa	S' ne	R' naa	n jaa	S' ne	n ha	P ree	m bha	P ja	S' na
S' bee	R'S' -	d -	d -	n -	P na	n -	mP -	m naa	- -	P da	n shru	g tee	gm ma	R dhu	S ra
m su	- -	P ra	d la	n ya	S'R' lee	g'R' -	S' na								
								m ra	P sa	d raM	d ga	n ai	n so	S' sa	S' maa
n Chaa	S' yo	R' aa	S' sa	d maa	d -	n -	P na	S'm' ha	g'm' ree	R' ha	S' ree	d su	d mi	n ra	P na
m daM	P ga	d sa	n ba	S'R' ra	g'R' see	S'n ya	S' na								

Figure 2. A *bandish* instance from swarganga website.

shows a *bandish* instance from swarganga. The name of this *bandish* is *'jaane naa jaane haree'* in raga *Adana* and in *teen taal* (16 beats cycle). The first row contains the *bol* information which details the tabla strokes corresponding to the *tala* of the *bandish*. Other rows have lyrics (bottom) along with the notes (top) corresponding to the lyrical sections. Each row corresponds to a *tala* cycle. In Hindustani notation system *S r R g G m M P d D n N* corresponds to C C$^{\#}$ D D$^{\#}$ E F F$^{\#}$ G G$^{\#}$ A A$^{\#}$ B notes in western music notation system, when the tonic is at C. A note followed by a single quotation at the right shows it is in the higher octave and a single quotation at the left implies lower octave. Notes mentioned within parenthesis are *kan* notes (grace notes). Each column represents a beat duration.

From this dataset we have considered 144 ragas for our study which are represented well with sufficient number of *bandishes*. Table 1 presents dataset statistics.

#bandishes	**#ragas**	**#notes**	**#kan swaras** (grace notes)
2955	144	2,95,411	50,749

Table 1. Dataset

6.1 Data Pre-processing

We take all *bandishes* in a raga for training the note-embeddings for the raga. *Kan* notes are also treated in the same way as other notes in the composition, since the *kan* notes also follow the raga rules. The notes are encoded into 36 unique numbers. The notes corresponding to a *tala* (rhythm) cycle is taken as a sequence. The input sequence length is determined by taking the average length of the sequences in a raga dataset; zero-padding (to the left) and left-trimming of sequences are applied to sequences shorter and longer than the average length respectively. If the length of a sequence is more than double the defined sequence length, it is split into 2 separate sequences.

7. EXPERIMENTS

7.1 Evaluation Methods

We rely on 2 different evaluation methods to validate our approach. The first one is based on *perplexity* that evaluates how well a note-sequence generator model (neural-network based, n-gram based *etc.*) can predict a new sequence in a raga. Since note-embeddings are an integral part of our architecture, a low-perplexed note-sequence generator model should learn more accurate note embeddings. The second method relies on *clustering* of ragas based on different raga-similarity measures computed using our approach and baselines.

7.1.1 Perplexity

Perplexity for a language model [2], is computed based on the probability values a learned model assigns to a validation set [10]. For a given model, perplexity (PP) of a validation set with notes $N_1, N_2, ..., N_n$ is defined as

$$PP(N_1, N_2, ..., N_n) = \sqrt[n]{\frac{1}{P(N_1, N_2, ..., N_n)}} \qquad (5)$$

where $P(N_1, N_2, ..., N_n)$ is the joint probability of notes in the validation set. A better performing model will have a lower perplexity over the validation set. For each raga dataset, perplexity is measured with a validation set taken from the dataset. For the LSTM based methods, the learned neural model provides the likelihood of a note, whereas the n-gram baseline uses the learned probabilities for different n-grams.

7.1.2 Clustering

For this evaluation, we take 14 ragas for which similarities between all the ragas and subsets of these ragas are known. These similarities are determined with the help of a professional Hindustani musician. The selected ragas are *Shuddha Kalyan, Yaman Kalyan, Yaman, Marwa, Puriya, Sohni, Alhaiya Bilawal, Bihag, Shankara, Kafi, Bageshree, Bhimpalasi, Bhairav and Jaunpuri*. The first clustering (`Clustering 1`) checks if all the 14 ragas are getting clustered according to their *thaat*. *Thaat* wise grouping of these 14 ragas are shown in Table 2. Since there are 6 different *thaats*, k is taken as 6 for this clustering. For the other clusterings, different subsets of ragas are selected according to the similarities to be verified. Other similarities and the ragas chosen (from the 14 ragas) to verify that are as listed below

- `Clustering 2`: *Sohni* is more similar to *Yaman* and *Yaman Kalyan* compared to ragas in other *thaats* because they share the same characteristic

Thaat	Ragas
Kalyan	Shuddha Kalyan, Yaman Kalyan, Yaman
Marwa	Marwa, Puriya, Sohni
Bilawal	Alhaiya Bilawal, Bihag, Shankara
Kafi	Kafi, Bageshree, Bhimpalasi
Bhairav	Bhairav
Asavari	Jaunpuri

Table 2. Thaat based grouping of the selected ragas

phrase (*MDNS*). To verify this, *Sohni, Yaman, Yaman Kalyan, Kafi, Bhairav* are considered taking k=3 and we expect the first 3 ragas to get clustered together and, *Kafi and Bhairav* in 2 different clusters.

- `Clustering 3`: Within *Kafi thaat*, *Bhimpalasi* and *Bageshree* are more similar compared to their similarity with *Kafi* because of the similarity in these ragas' characteristic phrases (*mDnS, mPnS*). To verify this, these 3 ragas are considered for clustering taking k=2 and we expect *Bhimpalasi* and *Bageshree* to get clustered together and *Kafi* in another cluster.

- `Clustering 4`: Raga *Jaunpuri* is more similar to *Kafi thaat* ragas because they differ only by a note. To verify this, *Jaunpuri, Kafi, Bageshree, Bhimpalasi, Bhairav, Shuddha Kalyan, Puriya, Bihag* are considered taking k=5. We expect *Jaunpuri* to be clustered together with *Kafi, Bageshree and Bhimpalasi* and the other ragas in 4 different clusters.

We apply these four clustering methods on our test dataset and evaluation scores pertaining to each clustering method is averaged to get a single evaluation score.

7.2 Setup

For the experiments, we consider notes from 3 octaves, amounting to a vocabulary size of 37 (including the null note). The common hyper-parameters for the LSTM based methods (our approach and one of the baselines) are kept the same. The number of LSTM blocks used in the LSTM layer is set to the sequence length. Each LSTM block has 24 hidden units, mapping the output to 24 dimensions. For all our experiments, embedding dimension is empirically set to 36. We use tensorflow (version: 0.10.0) [1] for the LSTM implementations. Note sequences are picked from each raga dataset ensuring the presence of $\sim$100 notes in total for the validation set. This size is made variable in order to accommodate variable length sequences. While training the network, the perplexity of the validation set is computed during each epoch and used for setting the early-stopping criterion. Training stops on achieving minimum perplexity and the note-embeddings at that instance are taken for our experiments.

For the clustering baseline, we employ one of the hierarchical clustering methods, agglomerative clustering (*linkage:complete*). In our setting, a hierarchical method is preferred over K-means because, K-means work well only with isotropic clusters [21] and it is empirically observed that our clusters are not always isotropic. Also when experimented, the clustering scores with K-means are less compared to agglomerative clustering for all the approaches. For implementing the clustering methods (both agglomerative and k-means) we use scikit-learn toolkit [23].

8. RESULTS

Before reporting our qualitative and quantitative results, to get a feel of how well note-embeddings capture raga similarities, we first visualize the 37 × 36 note-embedding matrices by plotting their heatmaps, higher intensity indicating higher magnitude of the vector component. Figure 3 shows heatmaps of embedding matrices for three ragas *viz.* *Yaman Kalyan*, *Yaman* and *Pilu*. *Yaman Kalyan* and *Yaman* are more similar to each other than *Pilu*. This is quite evident from the embedding heatmaps.

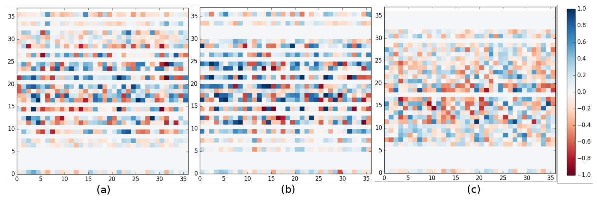


Figure 3. Note-embeddings visualization of (a) Yaman Kalyan (b) Yaman (c) Pilu

The results of quantitative evaluation is now reported with the evaluation methods described in Section 7.1. Further, a manual evaluation is done with the help of trained Hindustani musician considering all the 144 ragas mentioned in the dataset, to better understand the distinctions between bi-LSTM and uni-LSTM. Table 3 shows perplexity values (averaged across all the ragas in the dataset) with the validation set for our approach (bi-LSTM) and the baseline approaches with n-gram and uni-directional LSTM (uni-LSTM). We can not report perplexity for the PCD approach as the likelihood of the notes (and hence, the perplexity of the model) can not be determined with PCD. We observe that the perplexity values of n-gram and uni-LSTM are quite similar. The lower perplexity value with bi-LSTM shows its capability in generating a new notes sequence adhering to the raga rules. This shows the performance advantage of bi-LSTM over the baselines on note-sequence generation task, thereby providing indications on the goodness of the note-embeddings learned. Moreover, the bi-LSTM model, having the lowest perplexity, is able to capture the semantic association between notes more accurately, yielding more accurate note-embeddings.

Experiment	Perplexity
N-gram	6.39
uni-LSTM	6.40
bi-LSTM	**2.31**

Table 3. Results: Comparison with perplexity on validation set (Best performance in bold)

Experiment	Homogeneity	Completeness	V-measure
N-gram	0.3973	0.4036	0.4004
PCD	0.6430	0.6488	0.6451
uni-LSTM	0.7828	0.7858	0.7843
bi-LSTM	**0.9008**	**0.9069**	**0.9038**

Table 4. Results: Comparison of clustering results with different clustering metrics (Best performance in bold)

Table 4 shows the results of clustering using a standard set of metrics for clustering, *viz.* homogeneity, completeness and V-measure [26]. The clustering scores with n-gram and PCD baselines show their inability towards identifying the known similarities between the ragas. The bi-LSTM approach performs better compared to the baselines; the performance of uni-LSTM baseline is comparable with bi-LSTM approach. On analyzing each individual clustering, we observed,

- N-gram approach does not do well for all the individual clusterings, resulting in poor clustering scores compared to other approaches. A relatively better performance is observed only with `Clustering 4`.
- PCD has better scores compared to n-gram as it out-performs n-gram with a huge margin in `Clustering 1`. PCD's performance in `Clustering 1` is superior to the LSTM approaches as well. However, its performance is quite inferior to that of other approaches in the other three clustering settings. PCD's ability in modeling notes distribution efficiently helps in *thaat* based clustering (`Clustering 1`), because *thaat* based classification quite depends on the distribution of tonal material.
- uni-LSTM performance is better than bi-LSTM in `Clustering 1` where the ragas are supposed to be clustered according to the *thaat*. But it fails to cluster *Sohni*, *Yaman* and *Yaman Kalyan* in the same cluster, leading to poor performance in `Clustering 2`
- Even though bi-LSTM gives slightly lower scores with `Clustering 1`, it does perfect clustering for the other three clustering schemes. This gives an indication on the capability of bi-LSTM approach for identifying melodic similarities beyond *thaat*.

Overall, these observations show the practicality of both the LSTM based methods to learn note-embeddings with the aim of identifying raga similarity.

Figures 4 show Multi-Dimensional Scaling (MDS) [6] visualizations showing the similarity between note-embeddings of the selected 14 ragas (same color specifies same *thaat*) with bi-LSTM approach. These visualizations give an overall idea on how well the similarities are captured. The finer similarities observed in the clustering evaluations are not clearly perceivable from these visualizations.

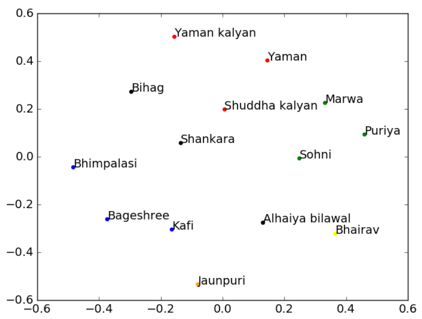


Figure 4. MDS visualization of bi-LSTM note-embeddings similarities

We have also carried out separate experiments by including note duration information along with the notes by pre-processing the data, but the performance is worse compared to the reported results. Chordia [9] has also reported that weighting by duration had no impact on their raga recognition task.

To confirm the validity of our approach, one expert musician checked the MDS visualizations of similarities between all 144 ragas with bi-LSTM and uni-LSTM approaches [1]. The musician identified clusters of similar ragas in both the visualizations matching with his musical notion. A few observations made are: *Asavari thaat* ragas appear to be closer to each other with bi-LSTM compared to uni-LSTM. Also *Miyan ki todi, Multani, Gujari Todi* which are very similar ragas are found closer in bi-LSTM. But the same *thaat* ragas *Marwa, Puriya and Sohni* are found to be more similar to each other with uni-LSTM.

9. CONCLUSION AND FUTURE WORK

This paper investigated on the effectiveness of note-embeddings for unveiling the raga similarities and on methods to learn note-embeddings. The perplexity based evaluation shows the superior performance of bi-directional LSTM method over unidirectional-LSTM and other baselines. The clustering based evaluation also confirms this, but it also shows that the performance of unidirectional approach is comparable to the bi-directional approach for certain cases.

The utility of our approach is not confined only to raga similarity; it can also be extended to verify if a given *bandish* complies with the raga rules. This immensely benefits to Hindustani music pedagogy; for instance, it helps to select the right *bandish* for a learner. In future, for better learning of note-embeddings, we plan to design a network to handle duration information effectively. The current experiments take one line in the *bandish* as a sequence. We plan to experiment with more meaningful segmentation schemes like lyrical phrase delimited by a long pause.

[1] The note-embeddings of all 144 ragas are available for download from `https://github.com/joecheriross/raga-note-embeddings`

10. ACKNOWLEDGMENTS

We would like to thank Swarganga.org and its founder Adwait Joshi for letting us to use the rich bandish dataset for research. We also thank Anoop Kunchukuttan, Arun Iyer and Aditya Joshi for their valuable suggestions. This work received partial funding from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement 267583 (CompMusic).

11. REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[2] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, pages 179–190, 1983.

[3] Shreyas Belle, Rushikesh Joshi, and Preeti Rao. Raga identification by using swara intonation. *Journal of ITC Sangeet Research Academy*, 23, 2009.

[4] Ashwin Bellur, Vignesh Ishwar, and Hema A Murthy. Motivic analysis and its relevance to raga identification in carnatic music. In *Proceedings of the 2nd CompMusic Workshop; 2012 Jul 12-13; Istanbul, Turkey. Barcelona: Universitat Pompeu Fabra; 2012. p. 153-157*. Universitat Pompeu Fabra, 2012.

[5] Arindam Bhattacharjee and Narayanan Srinivasan. Hindustani raga representation and identification: a transition probability based approach. *International Journal of Mind, Brain and Cognition*, 2(1-2):66–91, 2011.

[6] I Borg and P Groenen. Modern multidimensional scaling: theory and applications. *Journal of Educational Measurement*, 40(3):277–280, 2003.

[7] William B Cavnar and John M Trenkle. N-gram-based text categorization. *Ann Arbor MI*, 48113(2):161–175, 1994.

[8] Soubhik Chakraborty, Guerino Mazzola, Swarima Tewari, and Moujhuri Patra. *Computational Musicology in Hindustani Music*. Springer, 2014.

[9] Parag Chordia. Automatic raag classification of pitch-tracked performances using pitch-class and pitch-class dyad distributions. In *Proceedings of the International Computer Music Conference*, 2006.

[10] Philip Clarkson and Tony Robinson. Improved language modelling through better language model evaluation measures. *Computer Speech & Language*, 15(1):39–53, 2001.

[11] Pranay Dighe, Parul Agrawal, Harish Karnick, Siddartha Thota, and Bhiksha Raj. Scale independent raga identification using chromagram patterns and swara based features. In *IEEE International Conference on Multimedia and Expo Workshops (ICMEW) 2013*, pages 1–4. IEEE, 2013.

[12] Pranay Dighe, Harish Karnick, and Bhiksha Raj. Swara histogram based structural analysis and identification of indian classical ragas. In *The 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 35–40, 2013.

[13] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103, 2002.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Allen Huang and Raymond Wu. Deep learning for music. *arXiv preprint arXiv:1606.04930*, 2016.

[16] Adwait Joshi. swarganga.org, 2004.

[17] Manfred Junius, Alain Daniélou, Ernst Waldschmidt, Rose Waldschmidt, and Walter Kaufmann. The ragas of northern indian music, 1969.

[18] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[19] Tomas Mikolov, Stefan Kombrink, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5528–5531. IEEE, 2011.

[20] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 12th annual conference of the North American Chapter of the Association for Computational Linguistics*, volume 13, pages 746–751, 2013.

[21] George Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56(5):836–863, 1968.

[22] Gaurav Pandey, Chaitanya Mishra, and Paul Ipe. Tansen: A system for automatic raga identification. In *Proceedings of the 1st Indian International Conference on Artificial Intelligence*, pages 1350–1363, 2003.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[24] HG Ranjani, S Arthi, and TV Sreenivas. Carnatic music analysis: Shadja, swara identification and raga verification in alapana using stochastic models. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 29–32. IEEE, 2011.

[25] Suvarnalata Rao and Preeti Rao. An overview of hindustani music in the context of computational musicology. *Journal of New Music Research*, 43(1):24–33, 2014.

[26] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing-CoNLL*, volume 7, pages 410–420, 2007.

[27] Surendra Shetty, KK Achary, and Sarika Hegde. Clustering of ragas based on jump sequence for automatic raga identification. In *Wireless Networks and Computational Intelligence*, pages 318–328. Springer, 2012.

[28] Rajeswari Sridhar, Manasa Subramanian, BM Lavanya, B Malinidevi, and TV Geetha. Latent dirichlet allocation model for raga identification of carnatic music. *Journal of Computer Science*, 7(11):1711, 2011.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[30] T Viswanathan and Matthew Harp Allen. Music in south india, 2004.

IMPROVING NOTE SEGMENTATION IN AUTOMATIC PIANO MUSIC TRANSCRIPTION SYSTEMS WITH A TWO-STATE PITCH-WISE HMM METHOD

Dorian Cazau[1] **Yuancheng Wang**[2] **Olivier Adam**[3]
Qiao Wang[2] **Grégory Nuel**[4]
[1] Lab-STICC, ENSTA-Bretagne
[2] School of Information Science and Engineering, Southeast China
[3] Institut d'Alembert, UPMC
[4] LPMA, UPMC
dorian.cazau@ensta-bretagne.fr

ABSTRACT

Many methods for automatic piano music transcription involve a multi-pitch estimation method that estimates an activity score for each pitch. A second processing step, called note segmentation, has to be performed for each pitch in order to identify the time intervals when the notes are played. In this study, a pitch-wise two-state on/off first-order Hidden Markov Model (HMM) is developed for note segmentation. A complete parametrization of the HMM sigmoid function is proposed, based on its original regression formulation, including a parameter α of slope smoothing and β of thresholding contrast. A comparative evaluation of different note segmentation strategies was performed, differentiated according to whether they use a fixed threshold, called "Hard Thresholding" (HT), or a HMM-based thresholding method, called "Soft Thresholding" (ST). This evaluation was done following MIREX standards and using the MAPS dataset. Also, different transcription and recording scenarios were tested using three units of the Audio Degradation toolbox. Results show that note segmentation through a HMM soft thresholding with a data-based optimization of the $\{\alpha, \beta\}$ parameter couple significantly enhances transcription performance.

1. INTRODUCTION

Work on Automatic Music Transcription (AMT) dates back more than 30 years [21], and has known numerous applications in the fields of music information retrieval, interactive computer systems, and automated musicological analysis [16]. Due to the difficulty in producing all the information required for a complete musical score, AMT is commonly defined as the computer-assisted process of analyzing an acoustic musical signal so as to write down the musical parameters of the sounds that occur in it, which are basically the pitch, onset time, and duration of each sound to be played. In this study, we will restrict ourselves to this task of "low-level" transcription. Despite this large enthusiasm for AMT challenges, and several audio-to-MIDI converters available commercially, perfect polyphonic AMT systems are out of reach of today's technology.

The diversity of music practice, as well as supports of recording and diffusion, makes the task of AMT very challenging. These variability sources can be partitioned based on three broad classes: 1) instrument based, 2) music language model based and 3) technology based. The first class covers variability from tonal instrument timbre. All instruments possess a specific acoustic signature, that makes them recognizable among different instruments playing a same pitch. This timbre is defined by acoustic properties, both spectral and temporal, specific to each instrument. The second class includes variability from the different ways an instrument can be played, that vary with the musical genre (e.g. tonality, tuning, rhythm), the playing techniques (e.g. dynamics, plucking modes), and the personal interpretations of a same piece. These first two classes induce a high complexity of note spectra over time, whose non-stationarity is determined both by the instrument and the musician playing characteristics. The third class includes variability from electromechanics (e.g. transmission channel, microphone), environment (e.g. background noise, room acoustics, distant microphone), data quality (e.g. sampling rate, recording quality, audio codec/compression). For example, in ethnomusicological research, extensive sound datasets currently exist, with generally poor quality recordings made on the field, while a growing need for automatic analysis appears [9, 18, 20, 25].

Concerning AMT methods, many studies have used rank reduction and source separation methods, exploiting both the additive and oscillatory properties of audio signals. Among them, spectrogram factorization methods have become very popular, from the original Non-negative Matrix Factorization (NMF) to the recent developments of the Probabilistic Latent Component Analysis (PLCA) [2, 5]. PLCA is a powerful method for Multi-Pitch Estimation (MPE), representing the spectra as a linear combination of vectors from a dictionary. Such models take advantage of the inherent low-rank nature of magnitude spectrograms to provide compact and informative descriptions. Their output generally takes the form of a pianoroll-like matrix showing the "activity" of each spectral basis against time, that is itself discretized into successive time frame of analysis (of the order of magnitude of 11 ms). From this activity matrix, the next processing step in view of AMT is note segmentation, that aims to identify for each pitch the time intervals when the notes are played. To perform this operation, most spectrogram factorization-based transcription methods [11, 15, 22] use a simple threshold-based detection of the note activations from the pitch activity matrix, followed by a minimum duration pruning. One of the main drawback of this PLCA method with a simple threshold is that all successive frame are processed independently from one another, and thus temporal correlation between successive frames is not modeled. One solution that has been proposed is to jointly learn spectral dictionaries as well as a Markov chain that describes the structure of changes between these dictionaries [5, 22, 23].

In this paper, we will focus on the note segmentation stage, using a pitch-wise two-state on/off first-order HMM, initially proposed by Poliner et al. [24] for AMT. This HMM allows taking into account the dependence of pitch activation across time frames. We review the formalism of this model, including a full parametrization of the sigmoid function used to map HMM observation probabilities into the $[0, 1]$ interval, with a term α of slope smoothing and β of thresholding contrast. After demonstrating the relevance of an optimal adjustment of these parameters for note segmentation, a supervised approach to estimate the sigmoid parameters from a learning corpus is proposed. Note that Cheng et al. [8] explicitly modeled the different stages of a piano sound for note tracking, while we rather focus on more general musical features such as dynamics. Also, the Audio Degradation toolbox [19] was used to build three "degraded" sound datasets that have allowed to evaluate transcription performance on real life types of audio recordings, such as radio broadcast and MP3 compressed audio, that are almost never dealt with in transcription studies.

2. METHODS

2.1 Background on PLCA

PLCA is a probabilistic factorization method [26] based on the assumption that a suitably normalized magnitude spectrogram, V, can be modeled as a joint distribution over time and frequency, $P(f, t)$, with f is the log-frequency index and $t = 1, \dots, T$ the time index with T the number of time frames. This quantity can be factored into a frame probability $P(t)$, which can be computed directly from the observed data (i.e. energy spectrogram), and a conditional distribution over frequency bins $P(f|t)$, as follows [7]

$$P(f|t) = \sum_{p,m} P(f|p, m)P(m|p, t)P(p|t) \qquad (1)$$

where $P(f|p, m)$ are the spectral templates for pitch $p = 1, \dots, N_p$ (with N_p the number of pitches) and playing mode m, $P(m|p, t)$ is the playing mode activation, and $P(p|t)$ is the pitch activation (i.e. the transcription). In this paper, the playing mode m will refer to different playing dynamics (i.e. note loudness). To estimate the model parameters $P(m|p, t)$ and $P(p|t)$, since there is usually no closed-form solution for the maximization of the log-likelihood or the posterior distributions, iterative update rules based on the Expectation-Maximization (EM) algorithm [10] are employed (see [4] for details). The pitch activity matrix $P(p, t)$ is deduced from $P(p|t)$ with the Bayes' rule

$$P(p, t) = P(t)P(p|t) \qquad (2)$$

PLCA note templates are learned with pre-recorded isolated notes, using a one component PLCA model (i.e. m = 1 in Eq. (1). Three different note templates per pitch are used during MPE. In this paper, we use the PLCA-based MPE system developed by Benetos and Weyde [6] [1].

In the following, for $p = 1, \dots, N_p$ and $t = 1, \dots, T$, we define the logarithmic pitch activity matrix as

$$X_{p,t} = \log\big(P(p, t)\big) \qquad (3)$$

2.2 Note Segmentation Strategies

2.2.1 HT: Hard Thresholding

The note segmentation strategy HT consists of a simple thresholding β_{HT} of the logarithmic pitch activity matrix $X(p, t)$, as it is most commonly done in spectrogram factorization-based transcription or pitch tracking systems, e.g. in [11, 15, 22]. This HT is sometimes combined with a minimum duration constraint with typical post filtering like "all runs of active pitch of length smaller than k are set to 0".

2.2.2 ST: Soft Thresholding

In this note segmentation strategy, initially proposed by Poliner and Ellis [24], each pitch p is modelled as a two-state on/off HMM, i.e. with underlying states $q_t \in \{0, 1\}$ that denote pitch activity/inactivity. The state dynamics, transition matrix, and state priors are estimated from our "directly observed" state sequences, i.e. the training MIDI data, that are sampled at the precise times corresponding to the analysis frames of the activation matrix.

For each pitch p, we consider an independent HMM with observations $X_{p,t}$, that are actually observed, and hidden binary Markov sequence $Q = q_1, \dots, q_T$, illustrated in figure 1. The Markov model then follows the law:

[1] Codes are available at `https://code.soundsoftware.ac.uk/projects/amt_mssiplca_fast`.

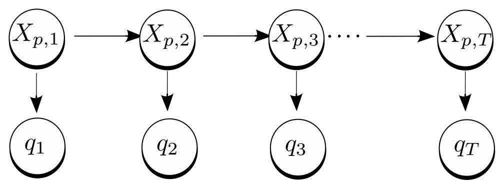


Figure 1. Graphical representation of the two-state on/off HMM. $q_t \in \{0, 1\}$ are the underlying states label at time t, and o_t the the probability observations.

$$P(Q,X) \propto P(q_1)\prod_{t=2}^{T} P(q_t|q_{t-1})\prod_{t=1}^{T} P(q_t|X_{p,t}) \quad (4)$$

where $\propto$ means "proportional to", as the probabilities do not sum to 1. For $t = 1, \dots, T$, we assume that:

$$P(q_t = 0|q_t = 0) = 1-\tau_0 \quad P(q_t = 1|q_t = 0) = \tau_0 \quad (5)$$

$$P(q_t = 0|q_t = 1) = \tau_1 \quad P(q_t = 1|q_t = 1) = 1-\tau_1 \quad (6)$$

with $\tau_0, \tau_1 \in [0, 1]$ the transition probabilities, and the convention that $q_0 = 0$ because all notes are inactive at the beginning of a recording. The transition probabilities τ correspond to the state transitions: on/on, on/off, off/on, off/off. Parameter τ_0 (resp. τ_1) is directly related to the prior duration of inactivity (resp. activity) of pitch p. Without observation, the length of an inactivity run (resp. activity run) would be geometric with parameter τ_0 (resp. τ_1) with average length $1/\tau_0$ (resp. $1/\tau_0$).

The observation probabilities are defined as follows, using a sigmoid curve with the PLCA pitch activity matrix $X_{p,t}$ as input,

$$P(q_t = 0|X_{p,t}) \propto 1/Z \quad (7)$$

$$P(q_t = 1|X_{p,t}) \propto e^{[e^{\alpha}(X_{p,t}-\beta)]/Z} \quad (8)$$

with $\alpha, \beta \in \mathbb{R}$, and Z defined such as $\sum_{q_t} P(q_t|X_{p,t}) = Z$. The parameter of the model is denoted $\theta = (\tau, \alpha, \beta)$ which includes the specific value for all pitches. The HMM model is solved using classical forward-backward recursions for all $t = 1, \dots, T$, i.e. $P_\theta(q_t = s|X_{p,t}) = \eta_s(t) \propto F_t(s)B_t(s)$.

Note that the HMM definition combines both the spatial pitch dependence (the Markov model) with a PLCA generative model. As a result of this combination, the resulting model is defined up to a constant factor, but this is not a problem since we will exploit this model to compute posterior distribution. In contrast, in the initial model [24], one should note that a similar model is suggested where the PLCA generative part is associated with the so-called "virtual observation". We here preferred the fully generative formulation presented above, but both models are totally equivalent.

Using logarithmic values, the parameters $\{\alpha, \beta\}$, expressed in dB, are directly interpretable by physics. β is an offset thresholding parameter, which allows separating signal from noise (or in other words, i.e. the higher its value, the more pitch candidates with low probability will be discarded.), while α is a contrast parameter, a value superior to 0 is used for a fast switch from noise to signal (i.e. low degree of tolerance from threshold), and a value inferior to 0 for a smoother switch. Figure 2 shows a sigmoid curve with different values of β and α. This suggested parametrization $\{\alpha, \beta\}$ can therefore be seen as a generalization of the initial [24]'s model.

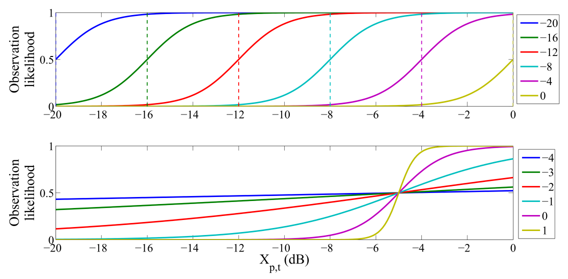


Figure 2. Effects of the parameters β (top) and α (bottom) on the theoretical sigmoid given by Eq. (8). On top, a fixed value of 0 is set to α, and on bottom, a fixed value of -5 is set to β.

For this note segmentation strategy ST, we use the set of parameters $\{\alpha, \beta\} = \{0, \beta_{\mathrm{HT}}\}$, as used in previous studies [5, 24].

2.2.3 *OST: Optimized Soft Thresholding*

The note segmentation strategy OST is based on the same HMM model as the ST strategy, although the parameters $\{\alpha, \beta\}$ are now optimized for each pitch. Given the ground truth of a musical sequence test, we use the Nelder-Mead optimizer of the R software to iteratively find the optimal $\{\alpha, \beta\}$ parameters that provide the best transcription performance measure. The Nelder-Mead method is a simplex-based multivariate optimizer known to be slow and imprecise but generally robust and suitable for irregular and difficult problems. For optimization, we use the Least Mean Square Error (LMSE) metric between ground truths and marginal posterior probabilities of pitch activation, as it allows to take into account the precise shape of activation profiles. Figure 3 provides an example of this optimization through the contour graph of the $\log_{10}$(LMSE) function. However, classical AMT error metrics (see Sec. 2.3.3) will be used as display variables for graphics as they allow direct interpretation and comparison in terms of transcription performance.

In real world scenarios of AMT, the ground truth of a musical piece is never known in advance. A common strategy to estimate model or prior knowledge parameters is to train them on a learning dataset that is somewhat similar to the musical piece to be transcribed. This was done in this study for the $\{\alpha, \beta\}$ parameters, through a cross-validation procedure with the LMSE-optimization (see Sec. 2.3.2).

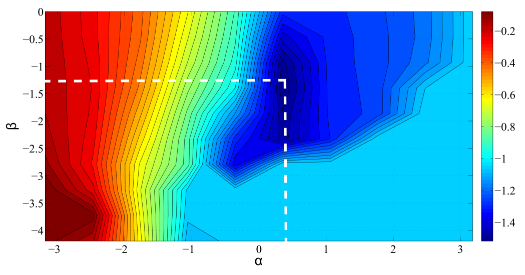


Figure 3. Example of a data-based optimization of the $\{\alpha, \beta\}$ parameters through the contour graph of the $\log_{10}$(LMSE) function, using the musical piece `MAPS_MUS-alb_esp2_AkPnCGdD`. The dashed white lines point to the local minimum.

2.3 Evaluation Procedure

2.3.1 *Sound Dataset*

To test and train the AMT systems, three different sound corpus are required: audio musical pieces of an instrument repertoire, the corresponding scores in the form of MIDI files, and a complete dataset of isolated notes for this instrument. Audio musical pieces and corresponding MIDI scores were extracted from the MAPS database [12], belonging to the solo classical piano repertoire. The 56 musical pieces of the two pianos labelled AkPnCGdD and ENSTDkCl were used, and constituted our evaluation sound dataset called Baseline. The first piano model is the virtual instrument Akoustik Piano (concert grand D piano) developed by the software Native Instruments. The second one is the real upright piano model Yamaha Disklavier Mark III. Three other sound datasets of musical pieces have then been defined as follows:

- MP3 dataset. It corresponds to the same musical pieces of the dataset Baseline, but modified with the Strong MP3 Compression degradation from the Audio Degradation toolbox [19]. This degradation compresses the audio data to an MP3 file at a constant bit rate of 64 kbps using the Lame encoder ;

- Smartphone dataset. It corresponds to the same musical pieces of the dataset Baseline, but modified with the Smartphone Recording degradation from the Audio Degradation toolbox [19]. This degradation simulates a user holding a phone in front of a speaker: 1. Apply Impulse Response, using the IR of a smartphone microphone ("Google Nexus One"), 2. Dynamic Range Compression, to simulate the phone's auto-gain, 3. Clipping, 3 % of samples, 4. Add Noise, adding medium pink noise ;

- Vinyl dataset. It corresponds to the same musical pieces of the dataset Baseline, but modified with the Vinyl degradation from the Audio Degradation toolbox [19]. This degradation applies an Impulse Response, using a typical record player impulse response, adds Sound and record player crackle, a Wow Resample, imitating wow-and-flutter, with the wow-frequency set to 33 rpm (speed of Long Play records), and adds Noise and light pink noise.

For all datasets, isolated note samples were extracted from the RWC database (ref. 011, CD 1) [14].

2.3.2 *Cross-validation*

During a cross-validation procedure, the model is fit to a training dataset, and predictive accuracy is assessed using a test dataset. Two cross-validation procedures were used for training the $\{\alpha, \beta\}$ parameters of the OST strategy, and testing separately the three thresholding strategies. The first one is the "leave-one-out" cross-validation procedure, using only one musical piece for parameter training and testing all others. This process is iterated for each musical piece. The second one is a repeated random sub-sampling validation, also known as Monte Carlo cross-validation. At each iteration, the complete dataset of musical pieces is randomly split into training and test data accordingly to a given training/test ratio. The results are then averaged over the splits. The advantage of this method (over k-fold cross validation) is that the proportion of the training/test split is not dependent on the number of iterations (folds). A number of 20 iterations was used during our simulations. We also tested different training/test ratio, ranging from 10/90 % to 80/20 % in order to evaluate the influence of the training dataset on transcription performance.

2.3.3 *Evaluation Metrics*

For assessing the performance of our proposed transcription system, frame-based evaluations are made by comparing the transcribed output and the MIDI ground-truth frame by frame using a 10 ms scale as in the MIREX multiple-F_0 estimation task [1]. We used the frame-based recall (TPR), precision (PPV), the F-measure (FMeas) and the overall accuracy (Acc)

$$\text{TPR} = \frac{\sum_{t=1}^{T} \text{TP}[t]}{\sum_{t=1}^{T} \text{TP}[t] + \text{FN}[t]} \tag{9}$$

$$\text{PPV} = \frac{\sum_{t=1}^{T} \text{TP}[t]}{\sum_{t=1}^{T} \text{TP}[t] + \text{FP}[t]} \tag{10}$$

$$\text{FMeas} = \frac{2.\text{PPV}.\text{TPR}}{\text{PPV} + \text{TPR}} \tag{11}$$

$$\text{Acc} = \frac{\sum_{t=1}^{T} \text{TP}[t]}{\sum_{t=1}^{T} \text{TP}[t] + \text{FP}[t] + \text{FN}[t]} \tag{12}$$

where T is the total number of time frames, and TP[t], TN[t], FN[t] and FP[t] are the numbers of true positive, true negative, false negative and false positive pitches at frame t. The recall is the ratio between the number of relevant and original items; the precision is the ratio between the number of relevant and detected items; and the F-measure is the harmonic mean between precision and recall. For all these evaluation metrics, a value of 1 represents a perfect match between the estimated transcription and the reference one.

2.3.4 MPE Algorithms on the Benchmark

In this study, we tested the four following MPE algorithms:

- Tolonen2000, this algorithm[2] [27] is an efficient model for multipitch and periodicity analysis of complex audio signals. The model essentially divides the signal into two channels, below and above 1000 Hz, computes a "generalized" autocorrelation of the low-channel signal and of the envelope of the high-channel signal, and sums the autocorrelation functions ;

- Emiya2010, this algorithm[3] [12] models the spectral envelope of the overtones of each note with a smooth autoregressive model. For the background noise, a moving-average model is used and the combination of both tends to eliminate harmonic and sub-harmonic erroneous pitch estimations. This leads to a complete generative spectral model for simultaneous piano notes, which also explicitly includes the typical deviation from exact harmonicity in a piano overtone series. The pitch set which maximizes an approximate likelihood is selected from among a restricted number of possible pitch combinations as the one ;

- HALCA, the Harmonic Adaptive Latent Component Analysis algorithm[4] [13] models each note in a constant-Q transform as a weighted sum of fixed narrowband harmonic spectra, spectrally convolved with some impulse that defines the pitch. All parameters are estimated by means of the EM algorithm, in the PLCA framework. This algorithm was evaluated by MIREX and obtained the 2^{nd} best score in the Multiple Fundamental Frequency Estimation & Tracking task, 2009-2012 [1] ;

- Benetos2013, this PLCA-based MPE system[5] [3] uses pre-fixed templates defined with real note samples, without updating them in the maximization step of the EM algorithm. It has been ranked first in the MIREX transcription tasks [1].

2.4 Setting the HT Threshold Value

We need to define the threshold value β_{HT} used in the note segmentation strategies HT and ST. Although most studies in AMT literature [11, 15, 22] use this note segmentation strategy, threshold values are barely reported and procedures to define them have not yet been standardize. Most of the time, one threshold value is computed across each evaluation dataset, which is dependent on various parameters of the experimental set-up, such as the used evaluation metric, input time-frequency representation, normalization of input waveform. In this paper, we will use a similar empirical dataset-based approach to define the HT threshold value. ROC curves (True Positives against False Positives) are computed over the threshold range [0 ; -5] dB so as to choose the value that maximizes True Positive and minimizes False Positives, i.e. that increases transcription performance at best over each dataset.

3. RESULTS AND DISCUSSION

All following results on transcription performance have been obtained using the Benetos2013 MPE system, except for figure 6 where all MPE systems are comparatively evaluated. Figure 4 represents the boxplots of the optimal $\{\alpha, \beta\}$ values obtained for each pitch. The "leave-one-out" cross-validation procedure has been applied to the different datasets, from top to bottom. For each dataset, we can see that the data-based pitch-wise optimization leads to β values drastically different from the threshold value β_{HT} used in the ST and HT thresholding strategies (represented by the horizontal red lines). Differences range from 0.5 to 2 dB, that have an important impact for note segmentation. Slighter differences are observed in values of α, although slightly positive values of α (around + 1 dB) tend to contribute to reduce the LMSE metric used in optimization. Also, note that optimal β_{HT} values are also dependent on the datasets, varying from -1.8 to -2.8 dB.

Now, let's see how this optimization of $\{\alpha, \beta\}$ in the method OST impacts real transcription performance. Table 1 shows transcription results obtained with the "leave-one-out" cross-validation procedure, applied to the different thresholding strategies. In comparison to the methods HT and ST, important gains in transcription performance are brought by the proposed method OST. These gains are the highest for the baseline dataset D_1, in the order of magnitude of 5 to 8 % for the two metrics Acc and FMeas. They remain systematically positive for the other datasets, with a minimum gain of 4 % whatever the dataset, error metric and compared thresholding strategy. Altogether, these gains are very significant in regards to common gains in transcription performance reported in literature, and demonstrate the validity of our proposed method.

In Figure 5, we evaluated the dependency of transcription performance on the training dataset size, through a Monte Carlo cross-validation procedure with different training/test ratios, ranging from 10 to 60 % of the complete dataset of musical pieces, plus the "leave-one-out" (labelled LOM) ratio. This figure shows that increasing the size of the training set directly induces average transcription gains from 0.5 to 6 % of the metric FMeas with the OST method, in comparison to the HT method. We note that once the curves reach the **60**/40 % training/test ratio, all systems find a quick convergence to the gain ceiling achieved with the LOM ratio.

Eventually, we studied the dependency of OST transcription performance on the MPE system used, in comparison to the method HT. Figure 6 shows the differences between the FMeas obtained with the methods OST and HT. We can observe that these differences are relatively

[2] We used the source code implemented in the MIR toolbox [17], called mirpitch(..., 'Tolonen').

[3] Source code courtesy of the primary author.

[4] Source codes are available at `http://www.benoit-fuentes.fr/publications.html`.

[5] Source codes are available at `https://code.soundsoftware.ac.uk/projects/amt_mssiplca_fast`.

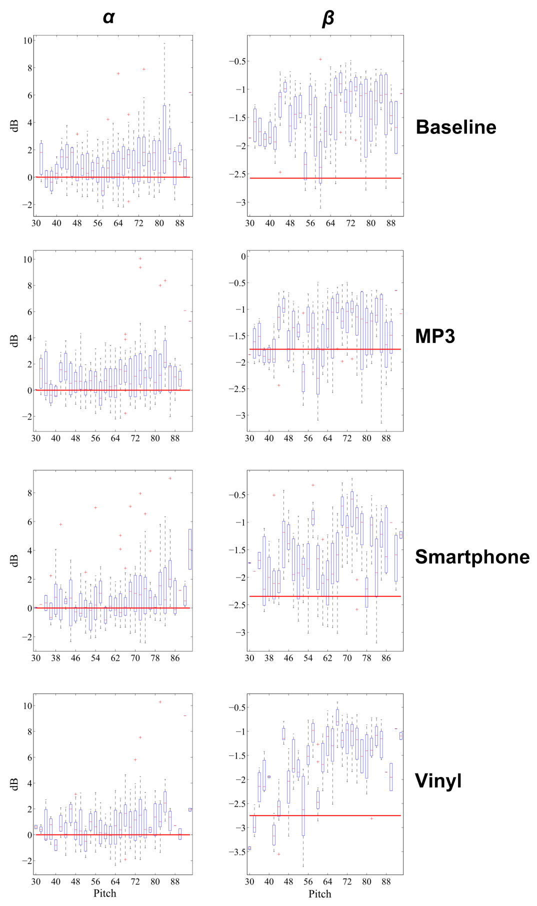


Figure 4. Boxplots of the optimal $\{\alpha, \beta\}$ values obtained for each pitch, and for each evaluation dataset. The horizontal red lines in each boxplot represents the parameter values used in the ST and HT thresholding strategies.

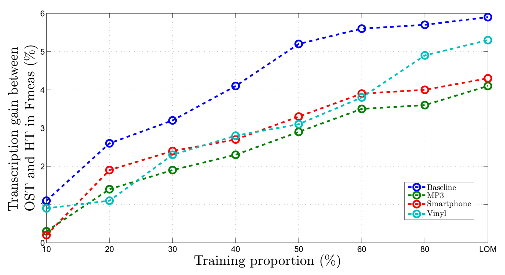


Figure 5. Difference between the F-measures obtained with the OST and HT note segmentation methods, using 20 iterations of the repeated random sub-sampling validation method with training/test ratio ranging from **10**/90 % to **60**/40 %, plus the "leave-one-out" (labelled LOM) ratio.

Datasets	Note segmentation strategies	Acc (%)	Fmeas (%)
Baseline	HT	54.9	53.3
	ST	57.6	55.3
	OST	**62.3**	**59.2**
MP3	HT	51.9	52.6
	ST	52.2	50.1
	OST	**55.6**	**56.7**
Smartphone	HT	52.2	51.9
	ST	53.1	51.3
	OST	**58.4**	**56.5**
Vinyl	HT	50.8	48.8
	ST	51.1	49.2
	OST	**57.8**	**54.1**

Table 1. Averages of error metrics FMeas and Acc obtained with the different thresholding strategies, i.e. ST, OST and HT, using a leave-one-out cross-validation procedure.

small, i.e. inferior to 2 %. This demonstrates that the proposed OST method improves transcription performance in a rather universal way, as independent from the characteristics of activation matrices as long as MPE system specific training datasets are used. Only MPE system Tolonen2000 shows higher transcription gains (especially for the datasets D_3 and D_4) brought by the OST method as this system outputs the worst activation matrices.

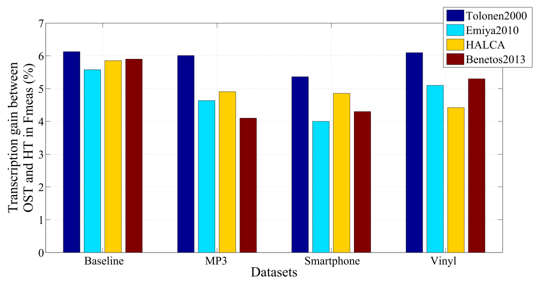


Figure 6. Difference between the F-measures obtained with the OST and HT note segmentation methods, using different MPE systems.

4. CONCLUSION

In this study, an original method for the task of note segmentation was presented. This task is a crucial processing step in most systems of automatic music transcription. The presented method is based on a two-state pitch-wise Hidden Markov Model method, augmented with two sigmoid parameters on contrast and slope smoothing that are trained with a learning dataset. This rather simple method has brought significant results in transcription performance on piano music datasets with different characteristics. It can also be used as a universal post-processing block after any pitch-wise activation matrix, showing great promise for future use, although it remains to be tested on different instrument repertoires.

5. REFERENCES

[1] MIREX (2007). Music information retrieval evaluation exchange (mirex). 2011. available at http://music-ir.org/mirexwiki/ (date last viewed January 9, 2015).

[2] V. Arora and L. Behera. Instrument identification using PLCA over stretched manifolds. In *Communications (NCC), 2014 Twentieth National Conference on*, pages 1–5, Feb 2014.

[3] E. Benetos, S. Cherla, and T. Weyde. An efficient shift-invariant model for polyphonic music transcription. In *6th Int. Workshop on Machine Learning and Music, Prague, Czech Republic*, 2013.

[4] E. Benetos and S. Dixon. A shift-invariant latent variable model for automatic music transcription. *Computer Music Journal*, 36:81–84, 2012.

[5] E. Benetos and S. Dixon. Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *J. Acoust. Soc. Am.*, 133:1727–1741, 2013.

[6] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *16th International Society for Music Information Retrieval Conference, Malaga , Spain*, pages 355–360, 2015.

[7] D. Cazau, O. Adam, J. T. Laitman, and J. S. Reidenberg. Understanding the intentional acoustic behavior of humpback whales: a production-based approach. *J. Acoust. Soc. Am.*, 134:2268–2273, 2013.

[8] T. Cheng, S. Dixon, and M. Mauch. Improving piano note tracking by HMM smoothing. In *23th EUSIPCO conference*, pages 2054–2058, 2015.

[9] O. Cornelis, M. Lesaffre, D. Moelants, and M. Leman. Access to ethnic music: Advances and perspectives in content-based music information retrieval. *Signal Proc.*, 90:1008–1031, 2010.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[11] A. Dessein, A. Cont, and G. Lemaitre. Real-time polyphonic music transcription with nonnegative matrix factorization and beta-divergence. In *11th International Society for Music Information Retrieval Conference, Utretcht, Netherlands*, pages 489–494, 2010.

[12] V. Emiya, R. Badeau, and G. Richard. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Trans. on Audio, Speech, Lang. Proc.*, 18:1643–1654, 2010.

[13] B. Fuentes, R. Badeau, and G. Richard. Harmonic adaptive latent component analysis of audio and application to music transcription. *IEEE Trans. on Audio Speech Lang. Processing*, 21:1854–1866, 2013.

[14] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. Rwc music database: Popular, classical, and jazz music databases. In *3rd International Conference on Music Information Retrieval, Baltimore,MD.*, pages 287–288, 2003.

[15] G. Grindlay and D. P. W. Ellis. Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments. *IEEE J. Sel. Topics Signal Proc.*, 5:1159–1169, 2011.

[16] A. Klapuri. Automatic music transcription as we know it today. *J. of New Music Research*, 33:269–282, 2004.

[17] O. Lartillot and P. Toiviainen. A matlab toolbox for musical feature extraction from audio. In *Proc. of the 10th Int. Conference on Digital Audio Effects (DAFx-07), Bordeaux, France, September 10-15, 2007*, 2007.

[18] T. Lidy, C. N. Silla, O. Cornelis, F. Gouyon, A. Rauber, C. A. A. Kaestner, and A. L. Koerich. On the suitability of state-of-the-art music information retrieval methods for analyzing, categorizing and accessing non-western and ethnic music collections. *Signal Proc.*, 90:1032–1048, 2010.

[19] M. Mauch and S. Ewert. The Audio Degradation toolbox and its application to robustness evaluation. In *14th International Society for Music Information Retrieval Conference, Curitiba, PR, Brazil*, pages 83–88, 2013.

[20] D. Moelants, O. Cornelis, M. Leman, J. Gansemans, R. T. Caluwe, G. D. Tré, T. Matthé, and A. Hallez. The problems and opportunities of content-based analysis and description of ethnic music. *International J. of Intangible Heritage*, 2:59–67, 2007.

[21] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, 1:32–38, 1977.

[22] G. J. Mysore and P. Smaragdis. Relative pitch estimation of multiple instruments. In *International Conference on Acoustical Speech and Signal Processing, Taipei, Taiwan*, pages 313–316, 2009.

[23] M. Nakano, J. Le Roux, H. Kameoka, O. Kitano, N. Ono, and S. Sagayama. Nonnegative matrix factorization with markov-chained bases for modeling time-varying patterns in music spectrograms. In *LVA/ICA 2010, LNCS 6365, V. Vigneron et al. (Eds.)*, pages 149–156, 2010.

[24] G. Poliner and D. Ellis. A discriminative model for polyphonic piano transcription. *J. on Advances in Signal Proc.*, 8:1–9, 2007.

[25] J. Six and O. Cornelis. Computer-assisted transcription of ethnic music. In *3th International Workshop on Folk Music Analysis, Amsterdam, Netherlands*, pages 71–72, 2013.

[26] P. Smaragdis, B. Raj, and M. Shanshanka. A probabilistic latent variable model for acoustic modeling. In *Neural Information Proc. Systems Workshop, Whistler, BC, Canada*, 2006.

[27] T. Tolonen and M. Karjalainen. A computationally efficient multipitch analysis model. *IEEE Trans. on speech and audio processing*, 8:708–716, 2000.

LARGE VOCABULARY AUTOMATIC CHORD ESTIMATION WITH AN EVEN CHANCE TRAINING SCHEME

Junqi Deng and Yu-Kwong Kwok
Department of Electrical and Electronic Engineering
The University of Hong Kong
{jqdeng,ykwok}@eee.hku.hk

ABSTRACT

This paper presents a large vocabulary automatic chord estimation system implemented using a bidirectional long short-term memory recurrent neural network trained with a skewed-class-aware scheme. This scheme gives the uncommon chord types much more exposure during the training process. The evaluation results indicate that: compared with a normal training scheme, the proposed scheme can boost the weighted chord symbol recalls of some uncommon chords and significantly improve the average chord quality accuracy, at the expense of the overall weighted chord symbol recall.

1. INTRODUCTION

Automatic chord estimation (ACE) is one of the central problems in music informatics. It asks for an algorithm to extract the harmonic progression within a piece of tonal music and label each harmony region with a chord symbol and a time stamp. For any artificial intelligence that is able to perform music analysis, an ACE algorithm will definitely be an important part of it.

For around two decades, ACE researches have been focusing around a very small vocabulary such as *major* and *minor* (or *majmin*) [1, 7, 16, 18, 22, 25, 30, 31]. Larger vocabularies are mostly only considered in some early works [12, 19, 26, 29]. Until recently, the large vocabulary issue has been brought back to the field [6, 9, 20], but except for the bass-treble chromagram proposed by Mauch and Dixon [21], the pre-segmented large vocabulary chord classification proposed by Deng and Kwok [8], and the Bayesian scaled likelihood estimation proposed by Humphrey [15], there is no technique specially designed for large vocabulary automatic chord estimation (LVACE).

Recently there has been a trend of using deep neural nets to solve ACE problems. Notable examples are: a convolutional neural network (CNN) based system [16], a hybrid fully connected neural network (FCNN) + recurrent neural network (RNN) system [3], a hybrid deep belief network (DBN) + RNN system [27], and a hybrid DBN + hidden Markov model (HMM) system [33]. They all show promising results comparable with or better than the state-of-the-art in terms of metrics that are based on *major* and *minor* triads. While last year there is a hybrid DBN + Gaussian-mixture-hidden-Markov-model (GMM-HMM) system [9] that tries to address the LVACE problem, it does not really pay special attention to the uncommon chords during the training process.

This paper, on the other hand, proposes a scheme that is dedicated to the uncommon and long-tail chords in the large vocabulary. The LVACE system is implemented with a standard feature extraction process and a bidirectional long short-term memory recurrent neural network (BLSTM-RNN) sequence decoder. Unlike the scaled likelihood estimation [15] that incorporates the prior distribution of chords into the estimation system, our large vocabulary strategy is to make sure each chord type has an uniform probability of being "seen" by the network at the start of each training case. This is called the "even chance" training scheme. Compared with a normal scheme that picks training cases at random, evaluation results show that the even chance training scheme can achieve much better uncommon weighted chord symbol recalls and significantly better average chord quality accuracy.

This paper is organized as follows: Section 2 elaborates on the LVACE system design; Section 3 describes the experimental setup, which contains the details of the proposed even chance training scheme; Section 4 reports and discusses the evaluation results; and finally Section 5 concludes the paper with the key findings and gives some possible future directions of LVACE.

2. THE LVACE SYSTEM

Figure 1 shows an overview of the LVACE system, which mainly contains a feature extraction module and a BLSTM-RNN sequence segmentation and classification module. In the following we will first elaborate on the feature extraction process, and then discuss the working mechanisms of the BLSTM-RNN.

2.1 Feature Extraction

The feature extraction process resembles the one described by Deng and Kwok [9]. It starts by resampling the raw

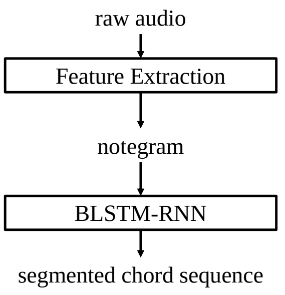


Figure 1. BLSTM-RNN LVACE system overview. The raw audio is transformed by a feature extraction process into a piece of notegram, and then decoded by a BLSTM-RNN into a segmented chord sequence.

audio input at 11025 Hz, which is followed by a short-time-Fourier-transform (STFT) with 4096-point Hamming window and 512-point hop size. It then proceeds to transform the linear-frequency spectrogram (2049-bin) to the log-frequency spectrogram (252-bin, 3 bins per semitone ranging from MIDI note 21 to 104) using the two cosine interpolation kernels proposed by Mauch [20]. The output at this step is a log-spectrogram $X_{k,m}$, where k is the index of frequency bins, and m is the index of time frames. The total number of frames is M, and the total number of bins in each spectrum is K (in this context K = 252).

The process then estimates the amount of deviation from standard tuning using the algorithm in [10], where the amount of detuning is estimated as:

$$\delta = \frac{wrap(-\varphi - 2\pi/3)}{2\pi}, \tag{1}$$

where $wrap$ is a function wrapping its input to $[-\pi, \pi)$ and φ is the phase angle at $2\pi/3$ of the discrete-Fourier-transform (DFT) of $\sum_m X_{k,m}/M$. The tuning frequency τ is then given by:

$$\tau = 440 \cdot 2^{\delta/12}, \tag{2}$$

and the original tuning is thus updated by interpolating the original spectrogram $X_{k,\cdot}$ at $X_{k+p,\cdot}$, where:

$$p = (\log(\tau/440)/\log(2)) \times 36, \tag{3}$$

since there are 36 bins per octave (3 bins per semitone) in $X_{k,\cdot}$. The interpolation results will update the original $X_{k,m}$, and the new $X_{k,m}$ spectrogram will be referred to as “notegram”, which will be the input feature of the BLSTM-RNN sequence decoder.

2.2 Recurrent Neural Network

An RNN is a neural network with cyclical connections, so that the network can be recurrently unrolled into multiple frames [11, 17]. It can be used to model the conditional probability of an output sequence $Y(Y^1, Y^2, ...)$ given an input sequence $X(X^1, X^2, ...)$, where the superscripts denote time steps. With a forward hidden layer, it models this relationship in a sequential manner, so that every output frame Y^t is conditioned on not only the current input frame X^t but also all previous input frames $X^{1:t}$. Besides, if a backward hidden layer is added to the model, Y^t will be conditioned on the whole input sequence X. This modified network is called *bidirectional recurrent neural network* (BRNN).

When the training sequence is long, the learning signal may die down gradually via the back-propagation-through-time (BPTT) [24]. This *gradient vanishing* phenomenon [2] often makes the training ineffective or unsuccessful. Using LSTM [13] units instead of normal non-linearities within a (B)RNN is a useful way to circumvent this undesirable effect.

2.3 BLSTM-RNN Architecture

The proposed LVACE system uses a BRNN with LSTM units, or a BLSTM-RNN. It has a forward and a backward hidden layer both with 800 LSTM units. The input layer has 252 real-value nodes, connected to a notegram spectrum. The output layer is a *#-chord-way* softmax layer. In this implementation, we use a typical LSTM configuration, that all LSTM gates employ sigmoid activations, and that both the LSTM cell and the LSTM output use hyperbolic tangent activations. Note that this network is different from the one in [9], in that this BLSTM-RNN could take a variable length input sequence and generate multiple outputs, but the other one is designed to handle a fixed length of input with a single softmax regression output.

3. EXPERIMENTAL SETUP

This section describes the vocabulary, datasets and training-validation schemes used in the experiments.

3.1 Vocabulary

The large vocabulary supported by the proposed system is the *SeventhsBass* introduced in MIREX ACE 2013 [1]. It contains the “*NC*” chord [2], all *maj* and *min* triads, all *maj7*, *min7*, and *7* chords, and all of their inversions.

3.2 Datasets and Data Augmentation

Six datasets of 546 tracks are used during the experiments. They contain both eastern and western pop/rock songs. They are: 20 tracks from the Chinese pop song dataset (CNPop20, or C) [3]; 29 tracks from the JayChou dataset (JayChou29, or J) [3]; 26 tracks from the Carole King + Queen dataset (K) dataset[4]; 191 songs from the USPop dataset (U)[5]; 100 tracks from the RWC dataset (R)[6]; and 180 tracks from the TheBeatles180 (B) dataset [14]. The combination of datasets is notated by concatenating their

[1] music information retrieval exchange:`http://www.music-ir.org/mirex/wiki/MIREX\_HOME`

[2] means “not a chord”, or “no chord”

[3] `http://tangkk.net/label/`

[4] `http://isophonics.net/datasets`

[5] `https://github.com/tmc323/Chord-Annotations`

[6] `https://staff.aist.go.jp/m.goto/RWC-MDB/`

letter codes. For example, a combination of all datasets is denoted as "CJKURB".

To generate the training data, all raw audios are transformed to notegram representations. The original segment-wise ground truth annotations are upsampled to become frame-wise annotations with 1-to-1 mappings to the notegram frames. Due to the absence of phase information in notegram, all data can be transposed to 12 keys to yield 12 times the original amount of data [16].

3.3 Training and Cross-validation

Two different training schemes are used. The only difference between them are the way of choosing training cases at each iteration:

- completely random (CR): a random training case is chosen.
- even chance (EC): a training case starting with a certain chord type is chosen, and each chord type has an even chance to be chosen as the start.

The EC training scheme is inspired by the skewed class sensitive training methods [5]. Considering a skewed distribution of chords in the training set [4], a random sampling scheme like CR will inevitably draw samples based on that same distribution, which causes lack of exposure of uncommon chords. The EC scheme, however, gives each uncommon chord much more exposure during the training process. Concretely, the EC scheme is formalized as follows in Algorithm 1:

Algorithm 1 EvenChanceTraining

```
Require: training data set - $(X, y)$; number of chord
  classes - $nclass$; early stopping flag - $es$.

  $od$ = BalancedOrderedDict($y$, $nclass$)
  $iter$ = 0
  while not early-stopping do
    if mod($iter$, $nclass$) is 0 then
      $coidx$ = random_shuffle(0:$nclass$-1)
    $tclist$ = $od(coidx_{mod(iter,nclass)})$
    draw a random item $e$ from $tclist$
    update network with $(X, y)_e$
    $iter$++
```

The core of this procedure is the "*BalancedOrdered-Dict*" which generates a dictionary of *(track index, chord change position)* tuples indexed by chord classes. It is formalized in Algorithm 2, where each entry of *od* contains a list of *(track index, chord change position)* tuples.

It should be pointed out that, besides chord classification, the BLSTM-RNN has to also perform segmentation, which means the training samples have to contain chord segmentation boundaries for the network to learn from. As a result, we set the length of each training case to be 500 frames, which contains multiple chords. Because of this, there is still uneven distribution of common and uncommon chords during the training process. The EC scheme

Algorithm 2 BalancedOrderedDict

```
Require: labels of training data set - $y$; number of chord
  classes - $nclass$.

  for each class $i$ from 0 to $nclass - 1$ do
    initialize an empty list $od[i]$
  for each track index $j$ in $y$ do
    for each frame poistion $k$ in $y[j]$ do
      if $k$ is a chord change position then
        append ($j$,$k$) to $od[y[j][k]]$
  return $od$
```

can guarantee a uniform chord distribution at the start of each training case, but it does not try to alter the sampling of the other chords. In effect, it only boosts the exposure of uncommon chords to a certain level, but could not make the chances of common and uncommon chords totally even.

The following describes the remaining training procedures that apply throughout the experiments. We try to report the precise settings of every parameter so that the readers may reproduce the results:

- Each training case contains 500 frames of audio content with ground truth labels;
- The network update signal is computed by an Adadelta optimizer [32];
- The training is regularized with dropout [28] and early-stopping [23];
- All dropout probabilities are set to 0.5;
- All early-stopping criteria are monitored using the validation error of the CNPop20 dataset, which is not in any cross-validation set; The validation cycle is 100 iterations;
- The model with the lowest validation loss will be saved; If the current validation loss is smaller than 0.996 of the best one, the early-stopping patience will increase by 0.3 times the current number of iterations;
- Training stops when the early-stopping patience is less than the current number of iterations.

For evaluation, five-fold cross-validation (CV) is performed throughout all experiments. Each fold is a combination of approximately 1/5 tracks of each dataset. Every model is trained on four folds and cross-validated on the remaining fold, resulting in a total number of five validation scores, the average of which will be the final scores to be reported in Section 4. For this research to be reproducible, all implementation details are made available online [7].

4. RESULTS AND DISCUSSIONS

Throughout this section, we use the MIREX ACE standard evaluation metric, "weighted chord symbol recall"

[7] `https://github.com/tangkk/tangkk-mirex-ace`

(*WCSR*), to report system performances. The "chord symbol recall" (*CSR*) is defined as follows:

$$CSR = \frac{|S \cap S^*|}{|S^*|}, \tag{4}$$

where S and S^* represents the automatic estimated segments, and ground truth annotated segments, respectively, and the intersection of S and S^* is the part where they overlap and have equal chord annotations. *WCSR* is the weighted average of all tracks' *CSR*s by the lengths of these tracks:

$$WCSR = \frac{\sum Length(Track_i) * CSR_i}{\sum Length(Track_i)}, \tag{5}$$

where the subscript i denotes the i^{th} track. Likewise, the *WCSR* of a specific chord type is:

$$WCSR_C = \frac{\sum Length(C_i) * CSR_i}{\sum Length(C_i)}, \tag{6}$$

where the subscript i denotes the i^{th} instance of chord C within the data set.

To measure the balanced performance of a system, we report "average chord quality accuracy" (*ACQA*) [6]:

$$ACQA = \frac{\sum WCSR_C}{\# \, of \, chords}. \tag{7}$$

which sums up the *WCSR*s of all chord types in the vocabulary. Systems that over-fit a few chord types or neglect uncommon chords tend to get lower *ACQA*s, while those well balanced systems will have higher *ACQA*s.

The original scores in this section are computed using the MusOOEvaluator [8].

4.1 Sevenths, Inversions and ACQA

Table 1 shows the comparison between CR and EC training schemes on some uncommon (*non-majmin*) [4] chords' *WCSR*s as well as the *ACQA*. The six chord types in the table are chosen because they have relatively more weights in pop/rock songs than the more long-tail ones such as *min/5* and *min/b3*. Note that *maj/5* and *maj/3* are also included in two other large vocabularies proposed by Mauch [20] and Cho [6].

	maj7	7	min7	maj/5	maj/3	7/b7	ACQA
CR	7.3	6.6	24.1	4.5	24.5	0.0	10.8
EC	14.6	9.9	30.9	12.0	32.4	7.8	13.2

Table 1. Comparison between CR and EC: seventh chords, inversions and ACQA scores; Dataset: JKURB

The results show that EC outscores CR in all categories, some of which by very large amount such as *maj/5* and *maj/3*. Although not all chord types' results are shown, the *ACQA* results suggest that the EC training scheme could lead to a much more balanced LVACE system under a skewed class distribution.

[8] `https://github.com/jpauwels/MusOOEvaluator`

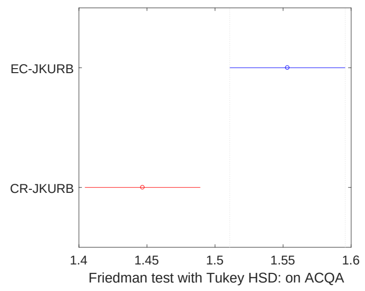


Figure 2. Multiple comparison test on ACQAs

We perform a Friedman test on the track-wise *ACQA* results of both systems. After that we use the Tukey HSD (honest significant difference) to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. The results as shown in Figure 2 confirm that EC is significantly better than CR in *ACQA*.

4.2 Major, Minor and WCSR

The EC trained system has a more balanced performance than the CR's, however, it scarifies common chords' *WCSR*s. Table 2 shows the comparison between CR and EC on some common (*majmin*) [4] chords' *WCSR*s as well as on the overall SeventhsBass *WCSR*.

	maj	min	WCSR
CR	74.2	52.2	52.0
EC	67.8	51.4	50.6

Table 2. Comparison between CR and EC: major, minor and WCSR scores; Dataset: JKURB

Although the two schemes have very close scores on *min*, there is a large difference in *maj*. Due to the dominantly large weight of *maj* chords in the JKURB dataset combination, it eventually leads to CR's much higher *WCSR*, despite EC performs better in most of the other chord types. CR's much higher *maj WCSR* is not unexpected: since it draws each training case at random, the probability that each chord type gets "seen" by the neural net is subject to the distribution of chord types in the training dataset, and therefore the *maj* chords are "learned" much more than the other chords.

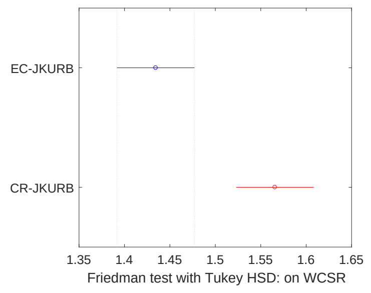


Figure 3. Multiple comparison test on WCSRs

We perform a Friedman test on the track-wise *WCSR* results of both systems. After that we use the Tukey HSD to perform a multiple comparison test on the Friedman test's statistics with a significance level of 0.05. The results as shown in Figure 3 confirm that CR is significantly better than EC in *WCSR*.

4.3 On Different Datasets

For more convincing comparison results, the same experiment is run 4 times using different dataset combinations. Table 3 shows the results of JK, JKU, JKUR and JKURB. We only report the *WCSR* and *ACQA* for brevity.

	CR-WCSR	EC-WCSR	CR-ACQA	EC-ACQA
JK	46.4	46.4	13.5	15.5
JKU	50.4	49.1	11.2	13.5
JKUR	50.1	49.6	12.8	14.5
JKURB	52.0	50.6	10.8	13.2

Table 3. Comparison between CR and EC: WCSR and ACQA on different datasets.

In all these experiments, the EC systems get higher *ACQA*s, but lower or equal *WCSR*s, than the CR systems. It is sufficient to say that EC is better at training a balanced performing LVACE system under skewed class distribution, while CR is better at training an LVACE system with higher overall performance.

For both training schemes, the increment of training data will lead to the increase of *WCSR*, but the same thing does not happen in *ACQA*. Assuming that every dataset contains a certain amount of noise (i.e., mis-labeled or mis-segmented chord regions), this observation could be tentatively explained as follows. *WCSR* is mostly relying on the quality of *majmin* chord labels, which are on average easier to be labeled. Therefore the increment of data will also increase the *WCSR* score. *ACQA*, however, is mostly relying on the quality of *non-majmin* chord labels, which are on average more difficult to be labeled. Therefore the increment of data could not guarantee the increase of *ACQA* score, since it is hard to guarantee the proportion of *non-majmin* noise in the incremental data is smaller than those of the original data.

5. CONCLUSIONS

This paper presents a BLSTM-RNN based LVACE system, trained using a skewed class oriented "even chance" scheme. This scheme is compared with a more intuitive "completely random" scheme that chooses training case randomly at each iteration. Evaluation results demonstrate that the EC training scheme is superior in both the uncommon (*non-majmin*) chords' *WCSR*s and the *ACQA*, at the expense of the common (*majmin*) chords' *WCSR*s and the overall *WCSR*.

A successful LVACE system is marked by both high *WCSR* and high *ACQA*, because human chord recognition experts are able to achieve both of them. The EC training scheme is a technique to improve a system's *ACQA*, thus it is a valuable approach to consider when we design an LVACE system in the future.

The fundamental driving force of LVACE research should be the ground-truth data and their qualities, especially the qualities of the uncommon or long-tail chords. As we see in the discussion above, *ACQA* is very vulnerable to uncommon chords' quality. Therefore, it might be possible that in the future as we gradually increase the amount of ground-truth data, we could use *ACQA* in a way to perform sanity check on the quality of the incremental data.

6. REFERENCES

[1] Juan Pablo Bello and Jeremy Pickens. A robust mid-level representation for harmonic content in music signals. In *Proceedings of the 6th International Society for Music Information Retrieval Conference, ISMIR*, volume 5, pages 304–311, 2005.

[2] Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[3] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR*, pages 335–340, 2013.

[4] John Ashley Burgoyne, Jonathan Wild, and Ichiro 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*, volume 11, pages 633–638, 2011.

[5] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

[6] Taemin Cho. *Improved techniques for automatic chord recognition from music audio signals*. PhD thesis, New York University, 2014.

[7] Taemin Cho, Ron J Weiss, and Juan Pablo Bello. Exploring common variations in state of the art chord recognition systems. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 1–8, 2010.

[8] Junqi Deng and Yu-Kwong Kwok. Automatic chord estimation on SeventhsBass chord vocabulary using deep neural network. In *Proceedings of the 41th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[9] Junqi Deng and Yu-Kwong 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.

[10] Karin Dressler and Sebastian Streich. Tuning frequency estimation using circular statistics. In *Proceedings of the 8th International Society for Music Information Retrieval Conference, ISMIR 2007*, pages 357–360, 2007.

[11] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[12] Takuya Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the 25th International Computer Music Conference*, volume 1999, pages 464–467, 1999.

[13] Alex Graves. Supervised sequence labelling. 2012.

[14] Christopher Harte. *Towards automatic extraction of harmony information from music signals*. PhD thesis, Department of Electronic Engineering, Queen Mary, University of London, 2010.

[15] Eric Humphrey. *An Exploration of Deep Learning in Content-based Music Informatics*. PhD thesis, New York University, 2015.

[16] Eric J Humphrey and Juan P Bello. Rethinking automatic chord recognition with convolutional neural networks. In *Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 357–362. IEEE, 2012.

[17] Michael I Jordan. Attractor dynamics and parallellism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Meeting of the Cognitive Science Society*, pages 531–546. Lawrence Erlbaum Associates, 1986.

[18] Maksim Khadkevich and Maurizio Omologo. Use of hidden Markov models and factored language models for automatic chord recognition. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009*, pages 561–566, 2009.

[19] Namunu C Maddage, Changsheng Xu, Mohan S Kankanhalli, and Xi Shao. Content-based music structure analysis with applications to music semantics understanding. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 112–119. ACM, 2004.

[20] Matthias Mauch. *Automatic chord transcription from audio using computational models of musical context*. PhD thesis, School of Electronic Engineering and Computer Science Queen Mary, University of London, 2010.

[21] Matthias Mauch and Simon Dixon. Approximate note transcription for the improved identification of difficult chords. In *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR*, pages 135–140, 2010.

[22] Yizhao Ni, Matt McVicar, Raul Santos-Rodriguez, and Tijl De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1771–1783, 2012.

[23] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[24] David E Rumelhart, James L McClelland, PDP Research Group, et al. *Parallel distributed processing*, volume 1. IEEE, 1988.

[25] Matti P Ryynänen and Anssi P Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[26] Alexander Sheh and Daniel PW Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proceedings of the 4th International Society for Music Information Retrieval Conference, ISMIR*, pages 185–191. International Symposium on Music Information Retrieval, 2003.

[27] Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[28] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[29] Borching Su and Shyh-Kang Jeng. Multi-timbre chord classification using wavelet transform and self-organized map neural networks. In *Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Citeseer, 2001.

[30] Jan Weil and Jean-Louis Durrieu. An HMM-based audio chord detection system: Attenuating the main melody. *The Music Information Retrieval Exchange*, 2008.

[31] Adrian Weller, Daniel Ellis, and Tony Jebara. Structured prediction models for chord transcription of music audio. In *International Conference on Machine Learning and Applications, ICMLA*, pages 590–595. IEEE, 2009.

[32] Matthew D Zeiler. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[33] Xinquan Zhou and Alexander Lerch. Chored detection using deep learning. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, volume 53, 2015.

LOCAL INTERPRETABLE MODEL-AGNOSTIC EXPLANATIONS FOR MUSIC CONTENT ANALYSIS

Saumitra Mishra, Bob L. Sturm, Simon Dixon
Centre for Digital Music, Queen Mary University of London, United Kingdom
{saumitra.mishra, b.sturm, s.e.dixon}@qmul.ac.uk

ABSTRACT

The interpretability of a machine learning model is essential for gaining insight into model behaviour. While some machine learning models (e.g., decision trees) are transparent, the majority of models used today are still black-boxes. Recent work in machine learning aims to analyse these models by explaining the basis of their decisions. In this work, we extend one such technique, called local interpretable model-agnostic explanations, to music content analysis. We propose three versions of explanations: one version is based on temporal segmentation, and the other two are based on frequency and time-frequency segmentation. These explanations provide meaningful ways to understand the factors that influence the classification of specific input data. We apply our proposed methods to three singing voice detection systems: the first two are designed using decision tree and random forest classifiers, respectively; the third system is based on convolutional neural network. The explanations we generate provide insights into the model behaviour. We use these insights to demonstrate that despite achieving 71.4% classification accuracy, the decision tree model fails to generalise. We also demonstrate that the model-agnostic explanations for the neural network model agree in many cases with the model-dependent saliency maps. The experimental code and results are available online. [1]

1. INTRODUCTION

Music content analysis (MCA) research aims to build systems with the sensitivity and intelligence required to work with information in acoustic environments. Recent advances in this domain have been made by leveraging large amounts of data with statistical machine learning, e.g., [5, 6]. The complexity of the resulting systems, however, makes it extremely difficult to understand their behaviours, or to predict their success in the real world.

Recent work seeks to ascribe certain functions or sensitivities to architectural elements of a trained system. For instance, analyses of deep computer vision systems find the first layer to be sensitive to edges, points and colour gradients, and deeper layers appear sensitive to higher-level concepts like faces, trees and cars [23,25,26]. Similar work for deep MCA systems has found that the first layer is sensitive to frequency bands, and deeper layers appear sensitive to timbres and temporal patterns, e.g., [3, 6]. In a different direction, other research focuses on approaches to explain individual predictions. One approach to explain individual predictions substitutes complex black-box models with inherently interpretable models whose predictions can be summarised by simple if-else rules [11,24]. Other methods use sensitivity analysis [7] or Taylor series expansion [15] to analyse the prediction function locally. Sensitivity analysis aims to capture the local behaviour of the prediction function when the input dimensions are perturbed. Variants of this approach include saliency maps [20], explanation vectors [1], "horse" detection [22] and local interpretable model-agnostic explanations (LIME) [17]. In this paper, we focus on extending LIME for MCA.

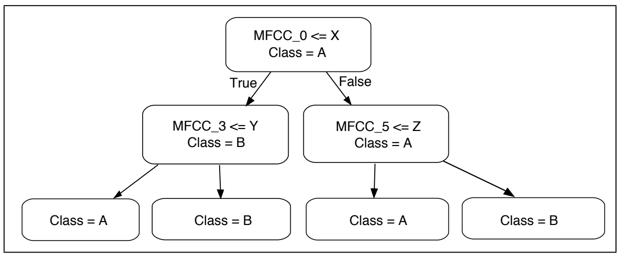


Figure 1: A binary decision tree for classifying audio using the values of three MFCC feature dimensions.

LIME is an algorithm that provides instance-based explanations to predictions of any classifier. These explanations are locally faithful to the instance, independent of the classifier model type, and are learned over interpretable representations of the instance. For example, for an e-mail classification system, LIME generates a list of words of an e-mail as an explanation for its classification to some category. To produce the explanation, LIME approximates the classifier locally with an interpretable model (e.g., sparse linear models, decision trees).

We introduce three different versions of explanations to apply LIME to MCA. We call this extended framework as Sound LIME (SLIME). Each version works in the time, frequency and time-frequency domains, respectively. SLIME pinpoints the time or time-frequency region that contributes most to a decision. This transforms a non-intuitive feature-based classifier decision into a more intuitive temporal and spectral description. We demon-

[1] https://code.soundsoftware.ac.uk/projects/SoundLIME

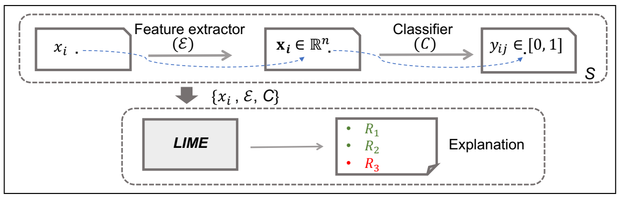


Figure 2: Schematic representation of LIME explaining why an MCA system S applies label j to instance x_i with probability y_{ij}.

strate SLIME for three trained singing voice detection systems, and show how the generated explanations are useful in gaining insight into model behaviour, and in identifying an untrustworthy model that fails to generalise.

2. MOTIVATION

Consider a simple MCA system, the classification component of which is the binary decision tree (BDT) shown in Fig. 1. The input to this system is a T-sec excerpt of audio, from which the system extracts D Mel-frequency cepstral coefficients (MFCC) [4]. This D-dimensional feature vector is labeled by the system as either "class A" or "class B" based on the values in specific dimensions. The particular dimensions, and the thresholds of the decisions, are found through training.

A binary decision tree is a transparent classifier because one can trace the reason for a particular outcome - in this case in terms of the MFCC coefficients and thresholds. As shown in Fig. 1, if the value of the zeroth MFCC is less than X and that of the third MFCC is less than Y, then this system classifies the instance as "class A". What does this mean in terms of the qualities of the input sound, however? Why X? Is this a real-world general principle? Or does it arise from a peculiarity of the training dataset?

MFCCs were introduced for speech recognition [4], but have been argued as suitable for machine music listening [9, 12]. Extracting MFCC features from audio involves windowing (typically on the order of 10-100 ms), a Mel-scale based smoothing of the log magnitude spectrum, and discrete cosine transform (DCT)-based compression. Although MFCC features are pseudo-invertible [2], they are difficult to interpret in terms of the qualities of the underlying sound. This comes in part from frequency bin grouping and the log magnitude operations, which destroy the bijective mapping between the audio and its spectrum.

One might still roughly approximate the meaning of particular MFCCs: low MFCC dimensions relate to broad spectral structures (e.g., formants); high MFCC dimensions relate to fine spectral structures (e.g., pitch and harmonics); and the zeroth MFCC relates to the energy of a signal. But, as shown in Fig. 1, values along several MFCC dimensions and their thresholds jointly contribute to a prediction. This combination makes interpretation even harder. It is hard to understand what audible qualities are captured by the combination of the zeroth MFCC with either the third or the fifth MFCC. Thus, though the decision tree has clear decision rules, they are not easy to relate to audible qualities of inputs. With other machine learning systems, e.g., deep neural networks or support vector machines, this task becomes harder still. This motivates the use of "interpretable representations" for explaining system behaviours for specific inputs.

3. INTERPRETABLE EXPLANATIONS FOR MUSIC CONTENT ANALYSIS

We first present the local interpretable model-agnostic explanations (LIME) proposed in [17]. We then extend it to working with MCA systems.

3.1 Summary of LIME [17]

Section 2 shows how the rules guiding a classifier's output can be difficult to interpret in terms of content, even for transparent classifiers. This interpretability becomes increasingly difficult when the model becomes complex (e.g., support vector machine) or the feature extraction is replaced by feature learning (e.g., convolutional neural network). LIME uses an interpretable representation of data to maintain interpretability in the generated explanations. Such explanations are easier because they show a more direct mapping between the input and its prediction.

LIME is an algorithm that generates interpretable, locally faithful and model-agnostic explanations to predictions of any classifier. Fig. 2 depicts a high-level overview of what LIME aims to perform. LIME helps illuminate reasons for a system S applying label j to instance x_i with probability y_{ij}. For example, for the input x_i, LIME lists three reasons: R_1, R_2 and R_3, to explain the prediction. R_1 and R_2 are positively correlated with the decision and R_3 is negatively correlated.

Locally faithful explanations refer to capturing the classifier behaviour in the neighbourhood of the instance to be explained. To learn a local explanation, LIME approximates the classifier's decision boundary around a specific instance using an interpretable model. LIME is *model-agnostic*, i.e., it considers the model as a black-box and makes no assumptions about the model behaviour. This makes LIME applicable to any classifier.

Formally, let $C : \mathbb{R}^n \rightarrow \mathbb{R}$ be a classifier, mapping a feature vector to a class label. For a feature vector $\mathbf{x}_i = \varepsilon(x_i)$, denote $y_{ij} = C(\mathbf{x}_i)$ as the probability that $\mathbf{x}_i$ takes the class label j. Define a sequence $\mathcal{X}_i$, which is composed of elements that are in some sense meaningful with respect to the classification of the instance x_i. For example, for a text classification system, $\mathcal{X}_i$ could be the sequence of unique words in e-mail. LIME defines an *interpretable space* $\mathcal{T} = \{0,1\}^{|\mathcal{X}_i|}$, where its kth dimension corresponds to the kth element of $\mathcal{X}_i$. Then $\mathbf{x}'_i \in \mathcal{T}$ is the *interpretable representation* of x_i. Thus, LIME transforms the input instance x_i to a binary vector $\mathbf{x}'_i$ whose elements correspond to presence and absence of elements of $\mathcal{X}_i$.

LIME defines an *interpretable explanation* as a model $g \in G$, where G denotes a class of interpretable models (e.g., linear models, decision trees). LIME learns a model g over the interpretable space by the optimisation:

$$\min_{g \in G} L(C, g, \rho_{x_i}) + \Delta(g) \quad (1)$$

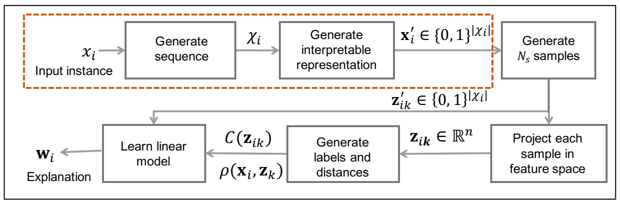

x_i
Input instance
Generate sequence
$\mathcal{X}_i$
Generate interpretable representation
$\mathbf{x}'_i \in \{0,1\}^{|\mathcal{X}_i|}$
Generate N_s samples
$\mathbf{z}'_{ik} \in \{0,1\}^{|\mathcal{X}_i|}$
Project each sample in feature space
$\mathbf{z}_{ik} \in \mathbb{R}^n$
Generate labels and distances
$C(\mathbf{z}_{ik})$
$\rho(\mathbf{x}_i, \mathbf{z}_k)$
Learn linear model
$\mathbf{w}_i$
Explanation

Figure 3: Functional block diagram of SLIME depicting the steps in generation of the explanation $\mathbf{w}_i$ for the prediction of the instance x_i.

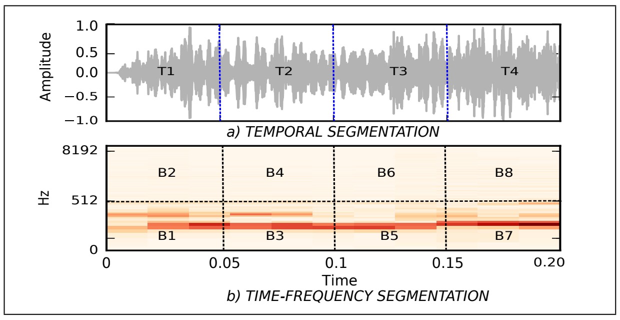


Figure 4: Segmentation based sequence generation for SLIME. (a) Temporal segmentation of instance x_i into four super samples (T_i), each of duration 50 ms. (b) Time-frequency segmentation of instance x_i into 8 blocks (B_i).

where $L(C, g, \rho_{x_i})$ is a locally-weighted loss function that for an instance x_i measures how well the model g approximates the classifier C in the neighbourhood defined by ρ_{x_i}, and $\Delta(g)$ is a measure of model complexity (e.g. sparsity in linear models). Thus, LIME minimises this function to explain why C maps x_i to class label j.

3.2 Extending LIME to MCA

Fig. 3 depicts the functional block diagram of SLIME. This consists of two components: the first one is our contribution (dotted box in Fig. 3), which defines interpretable sequences for an input audio. The second one is the LIME algorithm that uses the defined representations to generate explanations.

The first step in SLIME is to define a sequence denoted $\mathcal{X}_i$ from an input instance x_i. We define three kinds of sequences: temporal $\mathcal{X}_i^t$, spectral $\mathcal{X}_i^f$ and time-frequency $\mathcal{X}_i^{tf}$. We call each element of $\mathcal{X}_i^t$ a *super sample*, which we generate by temporal partitioning of x_i. For example, the instance shown in Fig. 4(a) is uniformly segmented into four super samples each notated T_i. Hence, $\mathcal{X}_i^t = (T_1, T_2, T_3, T_4)$. Similarly, each element of $\mathcal{X}_i^f$, notated A_i is a spectral magnitude in a corresponding frequency bin, obtained by the Fourier transform of x_i. Hence, $\mathcal{X}_i^f = (A_1, A_2, A_3, ...)$. Lastly, each element of $\mathcal{X}_i^{tf}$, notated B_i is obtained by segmenting the magnitude spectrogram of the input instance, both along the time and frequency axes. For example, in Fig. 4(b) the spectrogram of the instance is non-uniformly segmented into eight time-frequency blocks. Hence, $\mathcal{X}_i^{tf} = (B_1, B_2, ..., B_7, B_8)$. We call each element of a sequence as an *interpretable component*. Thus, for a temporal sequence each interpretable component is a supersample and for spectral and time-frequency sequences each interpretable component is a spectral bin and time-frequency block, respectively.

The next step is to map the input instance with feature representation denoted as $\mathbf{x}_i \in \mathbb{R}^n$ to its interpretable representation denoted as $\mathbf{x}'_i \in \{0,1\}^{|\mathcal{X}_i|}$. Thus, each of the above mentioned sequences is used to define an interpretable space $\mathcal{T}$ and an interpretable representation $\mathbf{x}'_i$. This creates three interpretable representations for the input instance x_i. We denote temporal, spectral and time-frequency interpretable representations as $\mathbf{x}_i^{t'}$, $\mathbf{x}_i^{f'}$ and $\mathbf{x}_i^{tf'}$ respectively. These representations provide us three ways of understanding a prediction, each highlighting the temporal, spectral or time-frequency segments of the instance influencing the prediction most.

To find an explanation, SLIME approximates the classifier $C : \mathbb{R}^n \to \mathbb{R}$ with a linear model $\{g(\mathbf{z}') = \mathbf{w}^T\mathbf{z}'; \mathbf{z}' \in \mathcal{T}\}$. To do this SLIME first generates N_s samples from $\mathcal{T}$ in a way that depends on $\mathbf{x}'_i$, i.e., randomly setting to zero the dimensions of $\mathbf{x}'_i$. Hence, for the interpretable sequence $\mathcal{X}_i^t$ in Fig. 4(a), one possible $\mathbf{z}_i^{t'} = (1, 0, 1, 0)$. This synthetic sample indicates the absence of super samples T_2 and T_4. Formally, for an instance with N_s super samples, a total of 2^{N_s} synthetic samples exists. With an assumption that there exists a surjective map from $\mathbb{R}^n$ to $\mathcal{T}$, each synthetic sample is projected to $\mathbb{R}^n$, weighted using an exponential kernel learned over cosine distance (we used the same ρ_{x_i} as in [17]) and mapped to its corresponding probability $C(z)$. SLIME learns the linear model g_t by minimising the squared loss and model complexity as in (1) over this dataset of synthetic samples and their probabilities. Formally, denote the kth sample as $\mathbf{z}'_k$ and its projection $\mathbf{z}_k$. Define a weight function $\rho_{x_i} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. The locally-weighted loss used by SLIME is given by

$$L(C, g, \rho_{x_i}) = \sum_{(\mathbf{z}'_k, \mathbf{z}_k) \in Z} \rho(\mathbf{x}_i, \mathbf{z}_k)[C(\mathbf{z}_k) - g(\mathbf{z}'_k)]^2 \quad (2)$$

Similarly, SLIME randomly samples $\mathbf{x}_i^{f'}$ and $\mathbf{x}_i^{tf'}$ to learn the linear models g_f and g_{tf}, respectively. Each of these models provides interpretable explanations in terms of their learned weights. The magnitude of the coefficients relates to the importance of the temporal segment (super sample) or the spectral component (bin frequency) or the time-frequency block in the classification of $\mathbf{x}_i$. Thus, if w_1 and w_2 denote the coefficients of super samples T_1 and T_2 respectively, then $|w_1| \geq |w_2|$ implies super sample T_1 has more influence on a classification prediction than T_2. Similarly, the polarity of regression weights refers to the correlation between the segment and the classifier prediction. For example, if $w_1 < 0$ and $w_2 > 0$, then the temporal segments T_1 and T_2 are negatively and positively correlated with the classifier prediction. The weight function ρ_{x_i} controls the contribution each synthetic sample has in the learned model g. Thus, a distant sample in interpretable space $\mathcal{T}$ will have lower contribution to g facilitating better learning in cases where the random sampling produces samples with highly imbalanced class distributions.

Classifier	*Acc[%]*	*Prec.*	*Recall*	*F- score*
Decision tree	71.4	0.72	0.81	0.75
Random forest	76.3	0.75	0.88	0.79

Table 1: Singing voice class evaluation results for the two selected shallow SVD systems (a) Binary decision tree of depth 8 and information gain as the split criterion (b) Random forest of 64 trees, each with depth 16.

4. DEMONSTRATION

We now use SLIME to explain the predictions of three singing voice detection (SVD) systems that classify an audio excerpt into two categories: music without singing voice, and music with singing voice. Two systems are based on a shallow architecture proposed in [10]. The other one is based on hierarchical feature learning [19].

4.1 Explaining Predictions of Shallow Vocal Detectors

Several shallow vocal detection systems have been proposed [10, 13, 16, 18]. We adapt the method proposed in [10] that uses only MFCC features to reach state of the art performance. Our system calculates FFTs on a frame size of 200 ms with 50% overlap at a sampling frequency of 22050 Hz. It uses a set of 30 Mel-filters to extract 30 MFCC coefficients (including the 0^{th}) and their first-order derivatives from each audio frame using *Librosa* [14]. The system performs classification over a 1 sec excerpt, hence it calculates the median and standard deviation of the 60 dimensional vector over five frames [18], constructing a feature vector of 120 dimensions.

We train two systems: the first (S_1) combines a binary decision tree (BDT) with the feature vector from above and the second (S_2) replaces the BDT with a random forest (RF) classifier. The *Jamendo* dataset, introduced in [16] is used to train, validate and evaluate both the models on three non-overlapping sets. Table 1 reports the results of the evaluation for singing voice class. The vocal detection systems designed using the BDT and RF classifiers achieve an overall accuracy of 71.4% and 76.3%, respectively. The vocal class occupies 57.5% of the test dataset which suggests that these two systems may have learnt some representation of singing voice that helps to detect vocals. We now apply SLIME to determine if these systems are trustworthy [22]. In other words, are the vocal predictions caused by content where there actually is voice?

In order to generate temporal explanations, we segment the instance (1 sec) into ten super samples, each of 100 ms duration. We first generate 1000 samples in the interpretable space. We then approximate each classifier's decision boundary in a neighbourhood of the instance by a linear model learnt over the interpretable space. The number of interpretable components needed to explain an instance may vary from one instance to the other, but to reduce the model complexity ($\Delta(g)$ in (1)), we generate explanations with a fixed number of components. To do this we first use the synthetic dataset of perturbed samples and their probabilities to select the top-3 super samples by forward selection, and then learn a linear model [17].

Id.	*Dur. (s)*	*Prob-Vocal*		*SS-Pred.*		*SS-True*
		BDT	*RF*	*BDT*	*RF*	
41	1.0	0.97	0.85	6,7,9	2,0,7	0-9
178	1.0	0.86	0.86	9,8,4	9,6,0	0-9
58	0.4	0.80	0.76	6,5,3	0,2,6	0-3
124	0.4	0.92	0.84	0,4,6	6,9,8	6-9

Table 2: Instance-based temporal explanations generated by SLIME. Id: instance index, Dur: vocal duration, SS: super samples, Prob-Vocal: probability assigned by the SVD system that the instance contains singing voice, SS-Pred: super sample indices that are the most influential upon the classification of the input instance to vocal class, SS-True: super sample indices that actually contain singing voice.

Table 2 reports the temporal explanations generated by SLIME for four instances extracted from the "03 - Say me Good Bye.mp3" test file in the Jamendo dataset. The super samples are arranged in the decreasing order of influence on the prediction. The magnitude of the weights learned for each super sample determines the influence it has on the prediction. This analysis of the temporal explanations helps to gain insight about how the models are forming their predictions. For example, instance 41 is correctly predicted by both the models (true positive). But, the temporal explanations for both the models are very different. The same is the case with another instance 178. Listening to all the predicted super samples for instances 41 and 178, highlights an interesting observation. For most of the predicted super samples for the decision tree model there is a presence of 'strong' instrumental onset along with the singing voice. Thus, it might be the case that instead of "listening" to the singing voice in the super sample, the decision tree model is paying attention to instrumental onset.

To verify the above hypothesis, we select true positive instances that have instrumental music and singing voice as separate temporal sections. We apply SLIME to two such instances: 58 and 124, which have singing voice in the first and last 400 ms, respectively. The temporal explanations generated for the BDT highlight that even though the prediction score is high for the model, the super samples it believes to contain singing voice have only instrumental music in most of the explanations. This raises questions about the generalisation capability of such a model. Based on the explanations generated for the RF model, it appears that the model is looking at the right temporal sections to form a prediction. Thus, the temporal explanations are helpful in identifying an untrustworthy model.

Temporal explanations help to understand the predictions but under some limitations. First, for the explanations to be clearly audible super samples should be at least 100ms long. Second, for the cases as in instance 41, where singing voice and instrumental music are present for complete duration, temporal explanations do highlight which temporal sections are useful for prediction but not what in that section is important. One way to solve this problem is to use SLIME to generate the spectral or time-frequency explanations as demonstrated below.

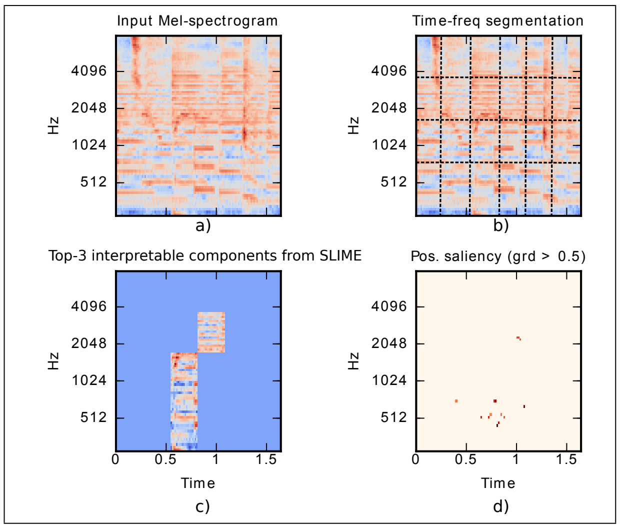


Figure 5: Comparing the positive explanation from SLIME with the positive saliency map for a 1.6s excerpt.

4.2 Explaining Predictions of a Deep Vocal Detector

We now demonstrate SLIME working with the convolutional neural network (CNN)-based system proposed in [19]. We generate time-frequency explanations for the predictions of the system and compare the generated explanations with the saliency maps [20, 21, 26]. Due to space restriction we have skipped the demonstration of spectral explanations, but such explanations can be easily derived from time-frequency explanations by expanding the temporal analysis window to full length of the excerpt.

The system proposed in [19] takes in a Mel-spectrogram representation of a 1.6 second audio excerpt and returns the probability that it contains singing voice. In order to explain the predictions of the system, we map the Mel-spectrogram to the time-frequency interpretable representation proposed in subsection 3.2. We segment the time-frequency axis of the input in 6 and 4 segments, respectively. Thus, the temporal axis of each of the first 5 segments is 266 ms in duration and that of the last segment is 280 ms. We aim to keep the temporal axis of the resulting interpretable components long enough to facilitate audition in the temporal domain. Similarly, segmentation along the frequency axis results in 4 spectral sections, each with 20 spectral bins. Thus, the input Mel-spectrogram is mapped to a sequence of time-frequency blocks $\mathcal{X}_i^{tf} = (B_1, ..., B_{24})$, where each block represents a dimension in the interpretable space. Fig. 5(a), (b) depict the Mel-spectrogram and its time-frequency segmentation, respectively for an input excerpt from “03 - Say me Good Bye.mp3” file from the Jamendo test dataset.

SLIME generates 2000 samples in the neighbourhood of the input, approximates the non-linear decision boundary by a linear model, and selects the top-3 interpretable components (time-frequency blocks) with the highest positive weights. Fig. 5(c) depicts the positive explanation for the prediction of the audio excerpt. We call an explanation positive if the weights of the interpretable components in the explanation are positive. The input excerpt chosen for analysis has singing voice with musical accompaniment for the first 900 ms and only musical accompaniment for the last 700 ms. We invert the time-frequency blocks in the explanation to temporal domain and on listening find that all the components in the explanation have the presence of singing voice. This raises confidence in the predictions of the model. Moreover, all the components in the negative explanation (not shown due to space restriction), fall in the temporal sections after 1s. This indicates that the time-frequency segments containing only instrumental music are negatively correlated with the classifier prediction. This also seems to be correct behaviour. Thus, the time-frequency explanations help to understand what sections in the input are influencing the prediction most.

We now compare SLIME-based explanations with saliency maps. Saliency maps, like time-frequency explanations, are tools to analyse black-box neural network models. They highlight how each input dimension influences the prediction. The gradient of the output prediction with respect to each input dimension is calculated to compute the saliency maps [20]. Thus, they depict the effect of modifying the input along any dimension, on the network prediction. Instead of allowing all the gradients to flow back, techniques proposed in [21, 26] only allow the positive gradient to flow back resulting in cleaner visualisations. Using the technique proposed in [26], we employ a leaky-ReLU non-linearity [8] in the backward path to reduce the magnitude of the negative gradients flowing back. We compare the positive time-frequency explanations with the positive saliency map. This map will highlight the input dimensions that are positively correlated with the classifier prediction. Not all the dimensions influence the predictions equally, thus we select only those dimensions whose normalised gradient is more than 0.5. We generate such maps for the output layer of the network. Fig. 5(d) shows the thresholded positive saliency map.

It is important to note that saliency maps highlight individual dimensions in the input while SLIME based explanations are time-frequency blocks. One way to compare the two is by visually verifying whether all the dimensions highlighted by the saliency maps are captured in the explanations created by SLIME. A visual comparison for the example in Fig. 5 shows that SLIME's explanation includes most of the key dimensions highlighted by the saliency map. Numerically we measure how many dimensions highlighted by the saliency map are enclosed in the explanation generated by SLIME. For the audio excerpt shown in Fig. 5, this agreement is 62.5%. We expand this analysis to a set of 1349 randomly chosen excerpts from the Jamendo test dataset. We found that on an average SLIME achieves 46.50 % numerical agreement when compared with the positive saliency maps. Instance-based analysis reveals that in some instances the numerical agreement is 100%, but there are cases where this number is less than 10%. One possible explanation for this is the shape of decision boundary near the instance. If the decision boundary is highly non-linear, approximating it with a linear model will result in poor explanations from SLIME.

We have not performed an exhaustive comparison (by

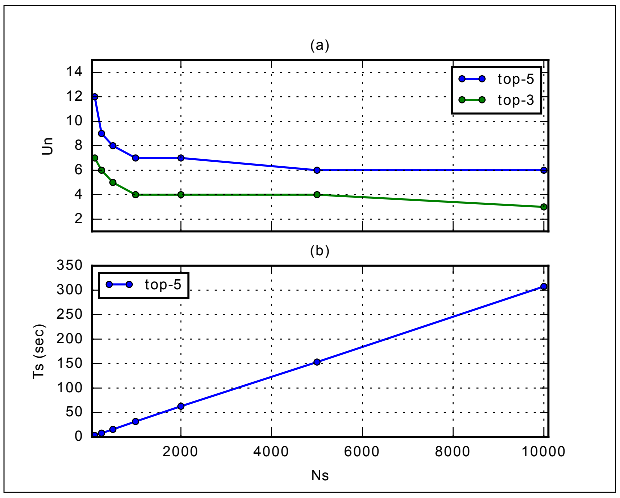


Figure 6: Plotting the effect of the number of samples (N_s) on (a) the stability of generated explanations (b) the time taken to generate them.

varying the preset factors, e.g. threshold, number of components) between the two techniques. The above analysis aims to provide an estimate about the performance of model-agnostic SLIME against a model-dependent technique for some preset values. It is obvious that the numerical agreement will be high if the constraints are softer and vice versa. Although saliency maps are accurate in highlighting the input dimensions that are influential to a classifier's output, they can suffer from lack of temporal context around the dimensions. On the other hand, SLIME-based explanations can be readily inverted to an acoustic form for audition, which may provide additional insights into how a classifier is forming its prediction for an input.

4.3 Discussion on the Number of Samples (N_s)

As discussed in subsection 3.2, to explain a prediction LIME generates N_s samples in the interpretable space ($\mathcal{T}$). In [17] there is no discussion about how many samples should be used to generate each explanation. We believe that exploring this is important for two reasons. First, it affects the time taken (T_s) to generate an explanation. Second, it affects the stability of the generated explanation. Ideally, an explanation should remain the same (at least the interpretable components, but their order and weights might change) even on multiple iterations of applying LIME to the same instance. But, empirically we find that the generated explanations do change on multiple iterations. This happens because LIME samples randomly in $\mathcal{T}$. In this section we seek to understand the effect of N_s on the stability of the explanations and on the time to generate one explanation.

For the experiment, we use the trained model, dataset and SLIME set-up from subsection 4.2. We randomly select 5 excerpts from each test file in the Jamendo dataset. We apply SLIME to generate explanation for the prediction of each excerpt in a batch of 80 and select the top-k interpretable components per explanation (we try k = 3 and 5). We iterate this process 5 times, each time randomly sampling 80 excerpts, generating explanations and selecting the top-k interpretable components.

We define the *stability* of an explanation to be inversely proportional to the number of unique interpretable components (U_n) from the sequence $\mathcal{X}_i$ that appear in explanations generated with m iterations. For example, if we apply SLIME $m = 2$ times to an instance and select the top-3 interpretable components in each iteration. Say the selected time-frequency segments are denoted as sets $\xi_1 = \{B_1, B_2, B_3\}$ and $\xi_2 = \{B_2, B_6, B_5\}$. Then U_n = 5, as B_2 appears twice in 6 components. To understand the effect of N_s on the stability of explanations, we generate 5 explanations for each of the 80 excerpts in the randomly sampled batches. We calculate the value of U_n in all the 5 explanations for each excerpt and plot the average result over 5 batches for a given value of N_s. Fig. 6(a) reports the results of the experiment. The result shows that U_n is inversely related to N_s, and thus the stability of the generated explanations is proportional to N_s. The result also shows that exhaustive search of the interpretable space $\mathcal{T}$ is not needed to generate stable explanations.

We also record the average time taken to generate one explanation for a given value of N_s. Results are generated by running SLIME on a computer with 1.6 GHz Intel core i5 processor and 8 GB memory and are reported in Fig. 6(b). Results show that T_s increases linearly with N_s, reaching to a maximum of around 5 mins for an explanation generated with N_s = 10k. The reported time includes the time taken for prediction by the CNN. These results suggest that selecting a suitable N_s depends on the trade-off between the stability of an explanation and the time-taken to generate it. In our experiment N_s = 1000 seems to be a good trade-off.

5. CONCLUSION

In this work we proposed SLIME, an algorithm that extends the applicability of LIME [17] to MCA systems. We proposed three versions of SLIME and demonstrated them with three types of singing voice detection systems to generate temporal and time-frequency explanations for the predictions of specific instances. We see that the temporal explanations generated by SLIME are helpful for revealing how the BDT is making decisions based on content that does not contain singing voice despite possessing high classification accuracy for the selected instances. Such issues cast doubt on the generalisability of the model. We also demonstrated that the analysis of time-frequency explanations is helpful to gain trust in the CNN based SVD system. We compared SLIME based explanations with saliency maps for the neural network model and the results suggest that model-agnostic SLIME based explanations agree in many cases with saliency maps.

In future we would like to apply SLIME to other MCA systems. We also plan to experiment with improved interpretable representations that will be created around audio "objects". We believe that the improved representations will assist in better understanding of the behaviour of the underlying machine learning model.

6. ACKNOWLEDGEMENTS

This work is supported in part by AHRC Grant Agreement no. AH/N504531/1. We would like to thank Jan Schlüter for sharing his implementation of the neural network based SVD system. We also thank the anonymous reviewers for their valuable comments and suggestions.

7. REFERENCES

[1] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. Muller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11:1803–1831, 2010.

[2] L. E. Boucheron and P. D. Leon. On the inversion of Mel-frequency cepstral coefficients for speech enhancement applications. In *Proc. of IEEE Intl. Conf. on Signals and Electronic Systems (ICSES)*, 2008.

[3] K. Choi, G. Fazekas, and M. B. Sandler. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444*, 2016.

[4] S. B. Davis and P. Mermelstein. Comparison of parametric representation for monosyllabic word recognition in continously spoken sentences. *IEEE Trans. on Acoustics, Speech and Signal Processing.*, 28(4):357–366, 1980.

[5] L. Deng and D. Yu. *Deep Learning: Methods and Applications*. Now Publisher, 2014.

[6] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *Proc. ICASSP*, 2014.

[7] M. Gevrey, I. Dimopoulos, and S. Lek. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 160(3):249–264, February 2003.

[8] A. Jannun, A. Maas, and A. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[9] J. H. Jensen, M. G. Christensen, D. P. W. Ellis, and S. H. Jensen. Quantitative analysis of a common audio similarity measure. *IEEE Trans. on Audio, Speech, and Language Processing*, 17(4):693–703, 2009.

[10] B. Lehner, R. Sonnleitner, and G. Widmer. Towards light-weight, real-time-capable singing voice detection. In *Proc. ISMIR*, 2013.

[11] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[12] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proc. ISMIR*, 2000.

[13] 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 *Proc. ISMIR*, 2011.

[14] 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 (SCIPY)*, 2015.

[15] G. Montavon, S. Bach, A. Binder, W. Samek, and K. R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[16] M. Ramona, G. Richard, and B. David. Vocal detection in music using support vector machines. In *Proc. ICASSP*, pages 1885–1888, 2008.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?": Explaining the predictions of any classifier. In *Proc. of the ACM Conf. on Knowledge Discovery and Data Mining (KDD)*, 2016.

[18] M. Rocamora and O. Herrera. Comparing audio descriptors for singing voice detection in music audio files. In *Brazilian Symposium on Computer Music*, volume 26, page 27, 2007.

[19] J. Schlüter and T. Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *Proc. ISMIR*, 2015.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Proc. ICLR Workshop*, 2014.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proc. of the Intl. Conf. on Learning Representations (ICLR) Workshop*, 2015.

[22] B. L. Sturm. A simple method to determine if a music information retrieval system is a "horse". *IEEE Trans. on Multimedia*, 16(6):1636–1644, 2014.

[23] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[24] F. Wang and C. Rudin. Falling rule lists. In *Proc. AISTATS*, 2015.

[25] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding Neural Networks Through Deep Visualization. In *Proc. of the Intl. Conf. on Machine Learning (ICML) Deep Learning Workshop*, 2015.

[26] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *Proc. of the European Conf. on Computer Vision*, 2014.

LYRIC JUMPER: A LYRICS-BASED MUSIC EXPLORATORY WEB SERVICE BY MODELING LYRICS GENERATIVE PROCESS

Kosetsu Tsukuda **Keisuke Ishida** **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
{k.tsukuda, ksuke-ishida, m.goto}@aist.go.jp

ABSTRACT

Each artist has their own taste for topics of lyrics such as "love" and "friendship." Considering such artist's taste brings new applications in music information retrieval: choosing an artist based on topics of lyrics and finding unfamiliar artists who have similar taste to a favorite artist. Although previous studies applied latent Dirichlet allocation (LDA) to lyrics to analyze topics, LDA was not able to capture the artist's taste. In this paper, we propose a topic model that can deal with the artist's taste for topics of lyrics. Our model assumes each artist has a topic distribution and a topic is assigned to each song according to the distribution. Our experimental results using a real-world dataset show that our model outperforms LDA in terms of the perplexity. By applying our model to estimate topics of 147,990 lyrics by 3,722 artists, we implement a web service called *Lyric Jumper* that enables users to explore lyrics based on the estimated topics. Lyric Jumper provides functions such as artist's topic taste visualization and topic-similarity-based artist recommendation. We also analyze operation logs obtained from 12,353 users on Lyric Jumper and show the usefulness of Lyric Jumper especially in recommending topic-related phrases in lyrics.

1. INTRODUCTION

Different artists have different tastes in lyrics. Some artists tend to sing about "love," while other artists tend to sing about "friendship." When listening to music, people choose artists according to not only musical audio content, such as music genre, mood, melody, vocal timbre, and rhythm, but also the topics of lyrics [2, 21]. However, the potential of using the topics of lyrics has not yet been fully exploited in the field of music information retrieval (MIR). For example, it is difficult to choose an artist based on the topics of their lyrics, find unfamiliar artists that are similar to the user's favorite artist in terms of the topics of the lyrics, and listen to a song that has the user's favorite topic of lyrics. The goal of this research is to achieve lyrics-based MIR that can leverage the topics of lyrics at both artist and song levels.

One approach for lyrics-based MIR is to directly use the words in lyrics. Users input some words as a query [5, 27] or can find the same phrase in the lyrics of another song while they are listening to music [9]. Another approach is to use a topic model because it can deal with the underlying meanings of lyrics. The topic is usually represented by a distribution over the vocabulary, and the meaning of topics (*e.g.*, "love" or "friendship") is determined based on the distribution. In lyrics-based MIR, it has been popular to use latent Dirichlet allocation (LDA) [1] as a topic model. LDA models each song as a mixture of topics and assigns a topic to each word in the song's lyrics. Since LDA does not take the set of songs of each artist into account, it is not able to capture the artist's taste for topics of lyrics.

In light of the above, we propose a topic model that considers the artist's taste for topics of lyrics. In the lyrics generative process of our model, each artist has a distribution over topics that reflects the artist's taste for topics in their lyrics. In addition, since it is common to decide the theme for a song before starting to write its lyrics [4, 36], our model assigns one topic to each song. That is, given a topic k assigned to a song, topic k is also assigned to the words in its lyrics. We also use the background word distribution because not all the words in lyrics are related to the topic.

By using our proposed model, we implemented a lyrics-based music exploratory web service, called *Lyric Jumper*[1] [2]. Lyric Jumper aims to enable users to explore lyrics and enjoy music in a more flexible way by considering songs' topics. Our proposed model automatically assigns 1 of 20 topics for each song, where the 20 topics are also automatically estimated by our model. Lyric Jumper provides several topic-based functions such as visualization of the topic tendency for a given artist, artist ranking based on topics, and artist recommendation based on the topic distribution similarity.

Our main contributions in this paper are as follows.

- To the best of our knowledge, this is the first study modeling a lyrics generative process by considering the artist's taste for topics of lyrics and assuming each song has one topic. (Section 3)
- We quantitatively evaluated our model by using a real-world song dataset provided by a lyrics distribution company. Our experimental results show that our proposed model outperformed the conventional LDA in terms of the perplexity. (Section 4)

[1] https://lyric-jumper.petitlyrics.com
[2] The demonstration video: https://youtu.be/5V9kHnelSAk

- By using our proposed model, we implemented a web service, called *Lyric Jumper*, that enables users to search for songs based on the topics of their lyrics. We also analyzed the search logs obtained from more than 12,000 users and showed the impact of Lyric Jumper on users' search behavior. (Section 5)

2. RELATED WORK

Previous studies have used lyrics for various objectives such as lyrics-to-audio alignment [7, 22, 35], analyzing lyrics characteristics [8, 13, 16, 29, 34], accurately finding lyrics [11, 19, 24], genre or mood classification [15, 25, 26, 38–40], songwriting support [30], and video generation [10]. This section describes more related studies in terms of (1) lyrics-based music retrieval/browsing systems and (2) topic-based lyrics analysis and applications.

2.1 Lyrics-Based Music Retrieval/Browsing Systems

Brochu and de Freitas [5] modeled music and text jointly so that users can search song databases using music and/or text as input. Müller *et al.* [27] automatically annotated audio recordings of a given song with its corresponding lyrics and realized a query-by-lyrics retrieval system. When a user selects a query result, the system can directly navigate to the corresponding matching positions within the audio. Detecting songs from the user's singing lyrics is also a popular research topic [14,37]. Fujihara *et al.* [9] proposed the concept of a "Music Web" where songs were hyperlinked to each other based on the phrases of lyrics. This enables users to jump to the same phrase in the lyrics of another song by clicking a linked phrase while they are listening to music. Visualization is also a useful approach to browse a music collection. *SongWords* [3], which is an application for tabletop computers, displays a music collection on a two-dimensional canvas based on self-organizing maps for lyrics and tags. *Lyricon* [23] is a system that automatically selects and displays icons that match the word sequences of lyrics so that users can intuitively understand the lyrics.

Although these studies directly use the words in lyrics, we consider topics that are automatically estimated from lyrics. Our approach has an advantage in that users can explore lyrics based on the underlying meanings of the lyrics.

2.2 Topic-Based Lyrics Analysis and Applications

Since a topic model can learn the underlying meanings of lyrics, it has been used in various studies, including lyrics analysis [17, 31, 33], lyrics retrieval applications [32], and a music player [28]. In terms of lyrics analysis, Sharma and Murty [33] analyzed the hidden sentimental structure behind lyrics by using LDA and revealed that some of the detected topics correspond to sentiments. Similarly, by applying LDA to rap lyrics, not only expected topics such as "street life" and "religion" but also unexpected ones such as "family/childhood" can be discovered [17]. Ren *et al.* [31] tackled the problem of predicting the popularity of a music track by considering lyrics topics and found that more than half of the popular tracks are related to the topic of "love." Regarding applications, *LyricsRadar* [32] is a lyrics retrieval system that visualizes the topic ratio for each song by using the topic radar chart and enables users to find their favorite lyrics interactively. Nakano and Goto [28] presented a music playback interface *LyricList-Player* that enables users to see word sequences of other songs similar to the sequence currently being played back, where the similarity is computed based on the topic.

In these studies, LDA is used as a topic model, where it is assumed that each song has a topic distribution and a topic is assigned to each word in the lyrics. We propose a new topic model that assumes each artist has a topic distribution and a topic is assigned to each song. Since our model outperforms LDA (see Section 4), there is the potential for improving previous studies on lyrics analysis and applications by using our model.

The study closest to ours is that of Kleedorfer *et al.* [18], who applied non-negative matrix factorization to lyrics for clustering them and manually labeled the cluster names. Our study differs from theirs in that we consider the artist's taste for topics of lyrics, and this enables users to find their favorite lyrics based on the relationships between artists and topics. Moreover, we not only propose a new model but also implement a web service so that everyone can explore lyrics with a real world dataset.

3. MODEL AND INFERENCE

In this section, after summarizing the notations used in our model in Section 3.1, we first describe LDA in Section 3.2 and then propose our model in Section 3.3.

3.1 Notations

Given a lyrics dataset, let A be the set of artists in the dataset. Let R_a be the number of songs of artist $a \in A$ in the dataset; then the set of a's songs is given by $\{S_{ar}\}_{r=1}^{R_a}$, where S_{ar} represents the rth song of a. Moreover, let V_{ar} be the number of words in the lyrics of S_{ar}; then S_{ar} can be represented by $S_{ar} = \{v_{arj}\}_{j=1}^{V_{ar}}$, where v_{arj} is the jth word in S_{ar}. Hence, the set of words of all artists' lyrics is given by $D = \{\{\{v_{arj}\}_{j=1}^{V_{ar}}\}_{r=1}^{R_a}\}_{a \in A}$.

3.2 LDA (Latent Dirichlet Allocation)

When LDA is used as a generative process of lyrics, it is assumed that (1) each song has a distribution over topics, (2) a topic is assigned to each word in the song's lyrics according to the distribution, and (3) a word is generated from the topic's distribution over words. Figure 1(a) shows the graphical model of LDA, where the shaded and unshaded circles represent the observed and unobserved variables, respectively. In the figure, K is the number of topics, θ is the song-topic distribution, and ϕ is the topic-word distribution. We assume that θ and ϕ have Dirichlet priors of α and β, respectively. The generative process of LDA is described in Algorithm 1.

3.3 Artist's Taste (AT) Model

Although previous studies reported the usefulness of applying LDA to lyrics [17, 28, 31–33], LDA does not take artist information into account in the generative process.

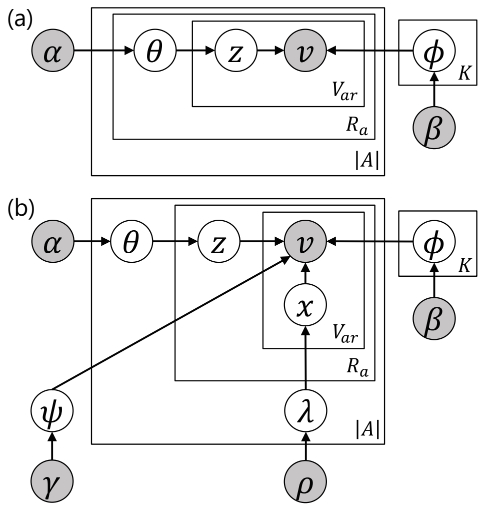


Figure 1. Graphical models of (a) baseline LDA and (b) proposed artist's taste (AT) model.

It is reasonable to assume that each artist has their own taste for topics of lyrics. For example, one artist may tend to sing lyrics related to the topic of "love," while another artist may tend to sing lyrics related to the topic of "life."

In light of the above, we propose a model that considers the *artist's taste* for topics. Figure 1(b) shows the graphical model of our proposed model. In our model, each artist has a distribution over topics (θ). When people write lyrics, the writer typically decides the theme (*i.e.*, the topic) before starting to write the lyrics [4, 36]. Hence, we assume each song has a topic z that is generated from θ. However, not all of the words in the lyrics are related to the topic. For example, although "thing" and "this" frequently appear in many lyrics, usually these words do not represent a specific topic. To solve this problem, we use the idea of background words [6]. In Figure 1(b), ψ represents the background word distribution, where words that are not related to any topic have high occurrence probabilities. Each artist has a Bernoulli distribution λ that controls the weights of influence for a song topic and background words. To be more specific, when artist a chooses a word in a song, we assume that the choice is influenced by the song topic with probability λ_{a0} $(x = 0)$ and by background words with probability λ_{a1} $(x = 1)$, where $\lambda_{a0} + \lambda_{a1} = 1$. When $x = 0$, a word is generated from the topic's distribution over words, while when $x = 1$, a word is generated from the background word distribution ψ. The generative process of the AT model is described in Algorithm 2.

3.4 Inference

To learn the parameters of our proposed model, we use collapsed Gibbs sampling [12] to obtain samples of hidden variable assignment. Since we use a Dirichlet prior for θ, ϕ, and ψ and a Beta prior for λ, we can analytically calculate the marginalization over the parameters. The marginalized joint distribution of D, latent variables $Z = \{\{z_{ar}\}_{r=1}^{R_a}\}_{a\in A}$, and latent variables $X = \{\{\{x_{arj}\}_{j=1}^{V_{ar}}\}_{r=1}^{R_a}\}_{a\in A}$ is computed as follows:

Algorithm 1 LDA generative process

```
for each topic $k \in \{1, \cdots, K\}$ do
  Draw $\phi_k \sim Dirichlet(\beta)$
end for
for each artist $a$ in $A$ do
  for each song $S_{ar}$ do
    Draw $\theta_{ar} \sim Dirichlet(\alpha)$
    for each word $v_{arj}$ in $S_{ar}$ do
      Draw a topic $z_{arj} \sim Multinomial(\theta_{ar})$
      Draw a word $v_{arj} \sim Multinomial(\phi_{z_{arj}})$
    end for
  end for
end for
```

Algorithm 2 AT model generative process

```
for each topic $k \in \{1, \cdots, K\}$ do
  Draw $\phi_k \sim Dirichlet(\beta)$
end for
Draw $\psi \sim Dirichlet(\gamma)$
for each artist $a$ in $A$ do
  Draw $\theta_a \sim Dirichlet(\alpha)$
  Draw $\lambda_a \sim Beta(\rho)$.
  for each song $S_{ar}$ do
    Draw a topic $z_{ar} \sim Multinomial(\theta_a)$
    for each word $v_{arj}$ in $S_{ar}$ do
      Draw switch $x \sim Bernoulli(\lambda_a)$
      if $x = 0$ then
        Draw a word $v_{arj} \sim Multinomial(\phi_{z_{ar}})$
      else if $x = 1$ then
        Draw a word $v_{arj} \sim Multinomial(\psi)$
      end if
    end for
  end for
end for
```

$$
\begin{aligned}
&P(D, Z, X|\alpha, \beta, \gamma, \rho) \\
&= \iiiint P(D, Z, X|\mathbf{\Theta}, \mathbf{\Phi}, \psi, \mathbf{\Lambda})P(\mathbf{\Theta}|\alpha) \\
&\quad \times P(\mathbf{\Phi}|\beta)P(\psi|\gamma)P(\mathbf{\Lambda}|\rho)d\mathbf{\Theta}d\mathbf{\Phi}d\psi d\mathbf{\Lambda}, \qquad (1)
\end{aligned}
$$

where $\mathbf{\Theta} = \{\theta_a\}_{a\in A}$, $\mathbf{\Phi} = \{\phi_k\}_{k=1}^{K}$, and $\mathbf{\Lambda} = \{\lambda_a\}_{a\in A}$. By integrating out those parameters, we can compute Equation (1) as follows:

$$
\begin{aligned}
&P(D, Z, X|\alpha, \beta, \gamma, \rho) \\
&\propto \prod_{a\in A} \frac{\Gamma(\rho + N_{a0})\Gamma(\rho + N_{a1})}{\Gamma(2\rho + N_a)} \frac{\prod_{v\in V} \Gamma(N_{1v} + \gamma)}{\Gamma(N_1 + \gamma|V|)} \\
&\times \prod_{k=1}^{K} \frac{\prod_{v\in V} \Gamma(N_{kv} + \beta)}{\Gamma(N_k + \beta|V|)} \prod_{a\in A} \frac{\prod_{k=1}^{K} \Gamma(R_{ak} + \alpha)}{\Gamma(R_a + \alpha K)}. \qquad (2)
\end{aligned}
$$

Here, N_{a0} and N_{a1} are the number of words in a's songs such that $x = 0$ and $x = 1$, respectively, and $N_a = N_{a0} + N_{a1}$. The term N_{1v} represents the number of times that word v was chosen under the condition of $x = 1$, and $N_1 = \sum_{v\in V} N_{1v}$ where V is the set of unique words in D. Furthermore, $N_k = \sum_{v\in V} N_{kv}$ where N_{kv} is the number of times word v is assigned to topic k under the condition of $x = 0$. Finally, R_{ak} is the number of times topic k is assigned to a's song, and $R_a = \sum_{k=1}^{K} R_{ak}$.

For the Gibbs sampler, given the current state of all but one variable z_{ar}, the new latent assignment of z_{ar} is sampled from the following probability:

$$\begin{aligned} &P(z_{ar} = k|D, X, Z_{\backslash ar}, \alpha, \beta, \gamma, \rho) \\ &\propto \frac{R_{ak\backslash ar} + \alpha}{R_a - 1 + \alpha K} \frac{\Gamma(N_{k\backslash ar} + \beta|V|)}{\Gamma(N_{k\backslash ar} + N_{ar} + \beta|V|)} \\ &\times \prod_{v \in V} \frac{\Gamma(N_{kv\backslash ar} + N_{arv} + \beta)}{\Gamma(N_{kv\backslash ar} + \beta)}, \end{aligned} \quad (3)$$

where $\backslash ar$ represents the procedure excluding the rth song of a. Moreover, N_{ar} and N_{arv} represent the number of words in the rth song of a and the number of times word v appears in the rth song of a, respectively.

In addition, given the current state of all but one variable x_{arj}, the probability at which $x_{arj} = 0$ is given by:

$$\begin{aligned} &P(x_{arj} = 0|D, X_{\backslash arj}, Z, \alpha, \beta, \gamma, \rho) \\ &\propto \frac{\rho + N_{a0\backslash arj}}{2\rho + N_a - 1} \frac{N_{z_{ar}v_{arj}\backslash arj} + \beta}{N_{z_{ar}\backslash arj} + \beta|V|}, \end{aligned} \quad (4)$$

where $\backslash arj$ represents the procedure excluding the jth word in the rth song of a. Similarly, the probability at which $x_{arj} = 1$ is computed as follows:

$$\begin{aligned} &P(x_{arj} = 1|D, X_{\backslash arj}, Z, \alpha, \beta, \gamma, \rho) \\ &\propto \frac{\rho + N_{a1\backslash arj}}{2\rho + N_a - 1} \frac{N_{1v_{arj}\backslash arj} + \gamma}{N_{1\backslash arj} + \gamma|V|}. \end{aligned} \quad (5)$$

Finally, we can make the point estimates of the integrated out parameters as follows:

$$\theta_{ak} = \frac{R_{ak} + \alpha}{R_a + \alpha K}, \quad \phi_{kv} = \frac{N_{kv} + \beta}{N_k + \beta|V|}, \quad \psi_v = \frac{N_{1v} + \gamma}{N_1 + \gamma|V|},$$
$$\lambda_{a0} = \frac{N_{a0} + \rho}{N_a + 2\rho}, \quad \lambda_{a1} = \frac{N_{a1} + \rho}{N_a + 2\rho}. \quad (6)$$

4. EVALUATION

In this section, we carry out a quantitative evaluation to answer the following research question: is adopting the artist's taste for topics effective to model the lyrics generative process?

[Dataset] We used the lyrics of commercially available popular music. Those lyrics with the song's title and artist name were provided by one of the largest companies for commercial lyrics distribution. We collected data on the top 1,000 artists in terms of the number of lyrics that are available as of the end of December 2016; this gave us 93,716 songs in total. We then extracted Japanese nouns from each song's lyrics by using MeCab [20], which is a Japanese morphological analyzer. Nouns that appeared in less than 10 lyrics were eliminated. Although our proposed model is language-independent, we used only Japanese words because of the understandability of the estimated topics for Japanese users of Lyric Jumper that we will describe in Section 5. From each of lyrics, we randomly sampled 80% of the nouns for training data and used the remaining 20 % for test data.

[Settings] In terms of hyperparameters, in line with other topic modeling work, we set $\alpha = \frac{1}{K}$ and $\beta = \frac{50}{|V|}$ in LDA and the artist's taste (AT) model. In addition, in the AT model, we set $\gamma = \frac{50}{|V|}$ and $\rho = 0.5$. To compare the performance of LDA and the AT model, we use the perplexities of the two models. Perplexity is widely used to compare the performance of statistical models [1], and the lower value represents the better performance. In terms of the number of topics, we compute the perplexity for $K = 2, 4, 6, 8, 10, 20, 30, 40$, and 50.

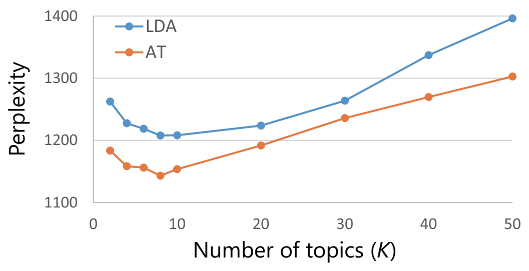


Figure 2. Perplexity for baseline LDA and proposed AT model (the lower, the better).

[Results] Figure 2 shows the perplexity. As can be seen, regardless of the number of topics, the AT model outperforms LDA. In both methods, the perplexity reaches a minimum when the number of topics is eight. The difference of perplexity between the two models becomes larger as the number of topics increases. From these results, we can conclude that the AT model is superior to LDA for modeling a lyrics generative process and confirmed the effectiveness of modeling a topic distribution for each artist.

5. LYRIC JUMPER

By using the AT model, we implemented a lyrics-based music exploratory web service called *Lyric Jumper* that anyone can use for free without registration. In this section, we describe the implementation and functions of Lyric Jumper followed by the log analysis based on the users' operation logs obtained from the web service.

5.1 Implementation

For Lyric Jumper, the lyrics are provided by the aforementioned company for commercial lyrics distribution. We used all the lyrics that are available as of the end of December 2016 and extracted Japanese nouns. To guarantee the topic quality, we eliminated artists who had < 10 songs and nouns that appeared in < 10 songs. This gave us 147,990 songs by 3,722 artists.

As for the number of topics K, if K is too small, users would soon get bored of using Lyric Jumper, while if K is too large, it would be difficult to understand the difference between topics since many similar topics are generated. Hence, after comparing the topic qualities for several K values, we set $K = 20$ for Lyric Jumper, although $K = 8$ achieved the best result in terms of the perplexity in Section 4. After automatically estimating the 20 topics by using the AT model, we manually labeled topic names so that users can easily understand the characteristics of each topic. Examples of topic names are "life," "sentimental," and "adolescence." Although five topics are related to "love," our model was able to distinguish between sub-

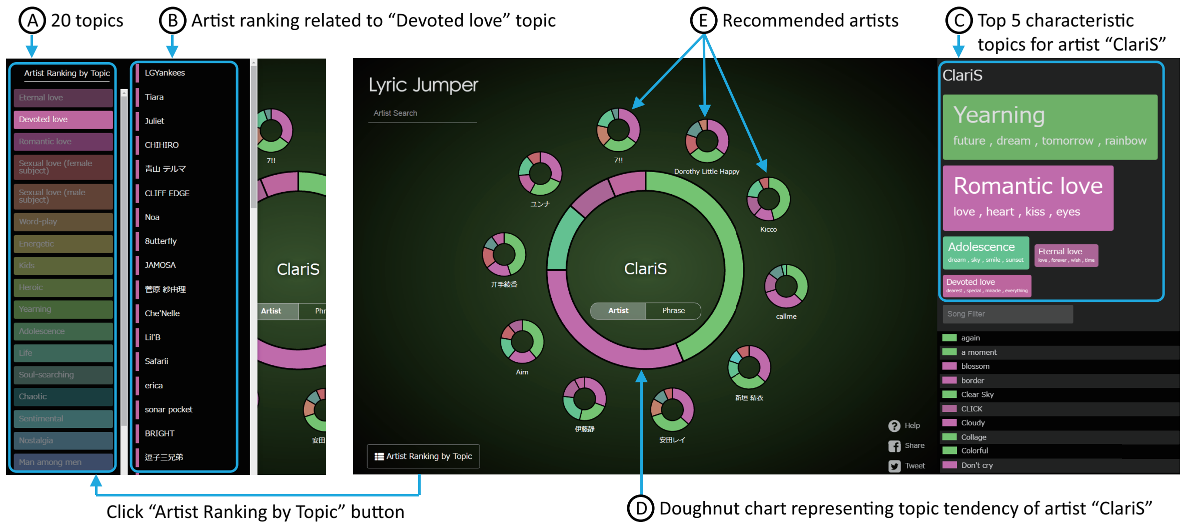


Figure 3. Overview of Lyric Jumper.

tle differences of love: "eternal love," "devoted love," "romantic love," "sexual love (female subject)," and "sexual love (male subject)."

5.2 Function

Lyric Jumper mainly provides six functions. The following sections describe the functions one by one.

5.2.1 *Artist Ranking*

Lyric Jumper displays 20 topic names as shown in Figure 3 Ⓐ. By clicking one of the 20 topics, Lyric Jumper shows up to 100 artists related to the topic (Figure 3 Ⓑ). This enables the user to see many artists related to the topic of his/her interest. The user can also find that unexpected artists are related to the topic.

Intuitively, given a topic k, artists are ranked based on both their topic ratio of k (θ_{ak}) and the number of songs assigned to k (R_{ak}) so that artists more closely related to the topic are ranked higher. To be more specific, we sort all the artists in A by using the rank of topic k in θ_a as the first key (the smaller the better) and using the number of songs assigned to k (R_{ak}) as the second key (the larger the better). Note that artists whose rank of k in θ_a is lower than five are not included in the ranking because Lyric Jumper shows the top five topics for each artist as we will describe in Section 5.2.2. Finally, we select the top 100 artists in the sorted list and show them to the users.

5.2.2 *Topic Tendency Visualization*

When a user clicks an artist, Lyric Jumper visualizes the topic tendency of the artist. In this function, given artist a, the top five topics in terms of the occurrence probability in θ_a are displayed in rectangles (Figure 3 Ⓒ). The size of a rectangle corresponds to the topic probability: the larger it is, the higher the probability is. With this function, a user can not only understand the topic tendency of the artist's lyrics but also find out that the artist sings songs with unexpected topics. We also manually selected four characteristic words for each topic and displayed them below the topic name so that users can more easily understand the meaning of the topic. In addition, Lyric Jumper visualizes the topic tendency using a doughnut chart where the circle is divided according to the ratio of the top five topics (Figure 3 Ⓓ).

5.2.3 *Artist Recommendation*

Since similar artists are one of the important information needs in MIR [21], Lyric Jumper provides a similar artists recommendation function. Lyric Jumper recommends 10 artists in terms of the topic similarity (Figure 3 Ⓔ). Among the 10 artists, eight artists are popular and two artists are minor. By displaying minor artists as well as popular ones, Lyric Jumper aims to encourage the user to listen to unfamiliar artists' songs that are related to his/her favorite artist by the topic similarity. By clicking a recommended artist's graph, the user can jump to the artist's search result.

Given a selected artist a, we compute the similarity between a and each artist $a' \in A \setminus \{a\}$ based on Jensen-Shannon divergence (JSD) between θ_a and $\theta_{a'}$. The smaller the JSD value is, the higher the similarity between artists is. After computing the similarities, we select the top eight similar artists who have $\geq m$ songs in the dataset (*i.e.*, popular artists) and the top two artists who have $< m$ songs (*i.e.*, minor artists) and show those 10 artists to the users. On Lyric Jumper, m is set to 100.

5.2.4 *Phrase Emphasized Lyrics Visualization*

When a user clicks a topic of the selected artist, Lyric Jumper shows the list of song titles of the artist that are assigned to the topic (the song ranking method will be described in Section 5.2.5). When the user clicks a title in the list, the song's lyrics are displayed (Figure 4). In the lyrics, lines[3] related to the topic are displayed with emphasis: the stronger the relation is, the larger the font size becomes and the darker the color becomes. This enables a user to easily understand the characteristics of the lyrics such as "the latter half of the lyrics is strongly related to the topic." Users can also watch the song's videos on Lyric Jumper by clicking the "YouTube Search" button. Lyric Jumper shows the search results obtained from YouTube[4]

[3] We use the terms "phrase" and "line" interchangeably.
[4] https://www.youtube.com/

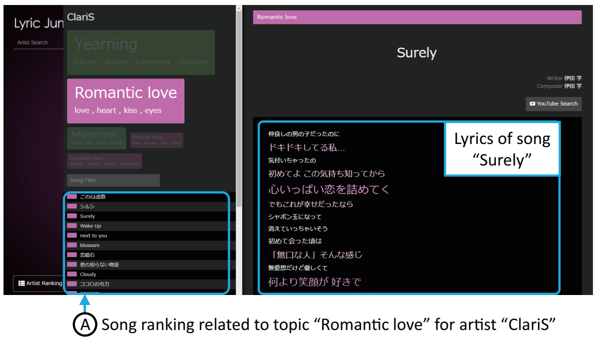


Figure 4. Phrase emphasized lyrics visualization.

where the query is the artist name and the song title.

The relevance score between a line in the lyrics and the topic is computed as follows. We first assign scores for the nouns in ϕ_k. Let $rank(k, v)$ be the occurrence probability rank of noun v in ϕ_k. The score of v is given by $w\_rel(k, v) = 101 - rank(k, v)$ if $rank(k, v) \leq 100$ and $w\_rel(k, v) = 0$ otherwise. Line l consists of $n \geq 0$ nouns and can be represented by $l = (v_1, \cdots, v_n)$. The relevance score of l with topic k is given by $l\_rel(k, l) = \sum_{i=1}^{n} w\_rel(k, v_i)$. After computing the scores of all lines in a song's lyrics, the scores are normalized to fit into the interval $[0, 1]$ by min-max normalization. The font size linearly changes from 16 pt for a score of 0 to 36 pt for a score of 1; the color density also linearly changes from #FFFFFF for a score of 0 to the topic color for a score of 1.

5.2.5 *Artist's Songs Ranking*

As mentioned in Section 5.2.4, when a user clicks a topic of the selected artist, Lyric Jumper returns the ranked list of songs in the topic (Figure 4 Ⓐ). Songs are sorted in descending order of relevance to the topic so that the user can easily access songs that are strongly related to the topic.

The relevance score between song s and topic k is given by $s\_rel(k, s) = \frac{1}{|L_s|} \sum_{l \in L_s} l\_rel(k, l)$, where L_s is the set of lines in s's lyrics. That is, we assume that the relatedness between s and k can be represented by the average relevance between k and each line in s's lyrics.

5.2.6 *Phrase Recommendation*

By clicking the "Phrase" button after selecting an artist's topic, Lyric Jumper recommends phrases related to the topic in the artist's songs (Figure 5). Moreover, every time the user clicks the "PUSH!" button, a new phrase is recommended. This function enables users to understand there are various expressions to deliver messages about the topic. When the user clicks a phrase, Lyric Jumper shows the corresponding lyrics in the same way as in Section 5.2.4.

Given artist a and topic k, the recommended phrases are selected as follows. In the ith round ($i = 1, 2, \cdots$), we pool lines that have the ith highest score of $l\_rel(k, l)$ from a's songs in order of decreasing $s\_rel(k, s)$. This round is repeated until the number of pooled lines is equal to 100. Lyric Jumper recommends phrases from the pooled list in random order so that users can see different phrases every time the user accesses Lyric Jumper.

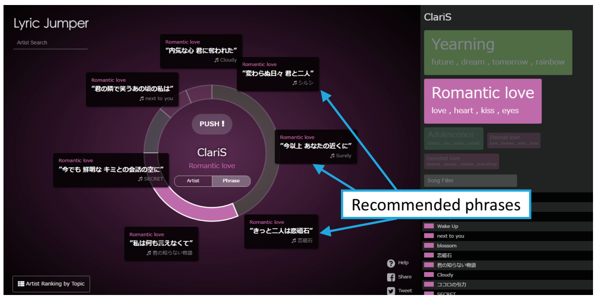


Figure 5. Phrase recommendation.

Function	PC	Smartphone
Artist ranking	2,092	30,295
Artist recommendation	1,706	4,016
Artist's songs ranking	5,399	14,665
Phrase recommendation	4,997	253,430

Table 1. Statistics of use frequency of each function.

5.3 Log Analysis

We released Lyric Jumper as a web service open to the public on 2/21/2017. To analyze users' exploratory behavior on Lyric Jumper, we obtained operation logs for 30 days (2/21 to 3/22). The numbers of unique PC users and smartphone users are 1,288 and 11,065, respectively. The use frequencies of each function are summarized in Table 1. We can see that the use frequencies of the artist ranking and artist's songs ranking are high. These results indicate that exploratory search for artists and songs based on topics can stimulate the user's interest. It can also be observed that for smartphone users in particular, the phrase recommendation function was used frequently: the push button was clicked as many as 253,430 times. This data shows the user's high information needs regarding finding lyrics using phrases related to a topic. Compared to these functions, the recommended artists were not clicked very often. To encourage the artist-similarity-based lyrics exploratory search, a more sophisticated interface for the recommendation deserves to be explored; we leave this as future work.

6. CONCLUSION

In this paper we proposed a topic model that incorporates the artist's taste for topics of lyrics. Our experimental results showed that our model outperformed the state-of-the-art LDA model regardless of the number of topics in terms of the perplexity. We also released a lyrics-based music exploratory web service called Lyric Jumper, where we applied our model to 147,990 lyrics by 3,722 artists. Our log analysis results show that the phrase recommendation function, which recommends phrases from the lyrics of the artist's songs related to the selected topic, achieved a particularly high use frequency.

For future work, since our model is language-independent, we plan to apply our model to English lyrics and implement an English version of Lyric Jumper. We are also interested in combining topics obtained by our model with other features such as audio content and tags. This would enable users to explore songs by adapting their search intent with increased flexibility.

7. ACKNOWLEDGMENTS

The authors would like to extend their appreciation to SyncPower Corporation for providing lyrics data. This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

8. REFERENCES

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

[2] 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*, ISMIR'03, pages 221–222, 2003.

[3] D. Baur, B. Steinmayr, and A. Butz. SongWords: Exploring music collections through lyrics. In *Proceedings of the 11th International Conference on Music Information Retrieval*, ISMIR'10, pages 531–536, 2010.

[4] M. Baxter. Voices of resistance, voices of transcendence: Musicians as models of the poetic - political imagination. *International Journal of Education & the Arts*, 11:1–24, 2010.

[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*, NIPS'02, pages 1505–1512. 2002.

[6] C. Chemudugunta, P. Smyth, and M. Steyvers. Modeling general and specific aspects of documents with a probabilistic topic model. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 241–248, 2006.

[7] G. Dzhambazov, 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 Conference on Music Information Retrieval*, ISMIR'16, pages 716–722, 2016.

[8] R. J. Ellis, Z. Xing, J. Fang, and Y. Wang. Quantifying lexical novelty in song lyrics. In *Proceedings of the 16th International Conference on Music Information Retrieval*, ISMIR'15, pages 694–700, 2015.

[9] 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*, ISMIR'08, pages 281–286, 2008.

[10] 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 Conference on Music Information Retrieval*, ISMIR'10, pages 63–68, 2010.

[11] G. Geleijnse and J. H. M. Korst. Efficient lyrics extraction from the web. In *Proceedings of the 7th International Conference on Music Information Retrieval*, ISMIR'06, pages 371–372, 2006.

[12] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, 2004.

[13] H. Hirjee and D. G. Brown. Automatic detection of internal and imperfect rhymes in rap lyrics. In *Proceedings of the 10th International Conference on Music Information Retrieval*, ISMIR'09, pages 711–716, 2009.

[14] 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*, ISMIR'05, pages 532–535, 2005.

[15] X. Hu and J. S. Downie. When lyrics outperform audio for music mood classification: A feature analysis. In *Proceedings of the 11th International Conference on Music Information Retrieval*, ISMIR'10, pages 619–624, 2010.

[16] X. Hu and B. Yu. Exploring the relationship between mood and creativity in rock lyrics. In *Proceedings of the 12th International Conference on Music Information Retrieval*, ISMIR'11, pages 789–794, 2011.

[17] C. Johnson-Roberson and M. Johnson-Roberson. Temporal and regional variation in rap lyrics. In *NIPS Workshop on Topic Models: Computation, Application and Evaluation*, NIPSW'13, 2013.

[18] F. Kleedorfer, P. Knees, and T. Pohle. Oh oh oh whoah! towards automatic topic detection in song lyrics. In *Proceedings of the 9th International Conference on Music Information Retrieval*, ISMIR'08, pages 287–292, 2008.

[19] 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*, ISMIR'05, pages 564–569, 2005.

[20] T. Kudo, K. Yamamoto, and Y. Matsumoto. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP'04, pages 230–237, 2004.

[21] 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*, ISMIR'04, pages 989–992, 2004.

[22] K. Lee and M. Cremer. Segmentation-based lyrics-audio alignment using dynamic programming. In *Proceedings of the 9th International Conference on Music Information Retrieval*, ISMIR'08, pages 395–400, 2008.

[23] W. Machida and T. Itoh. Lyricon: A visual music selection interface featuring multiple icons. In *Proceedings of the 15th International Conference on Information Visualisation*, IV'11, pages 145–150, 2011.

[24] R. Macrae and S. Dixon. Ranking lyrics for online search. In *Proceedings of the 13th International Conference on Music Information Retrieval*, ISMIR'12, pages 361–366, 2012.

[25] 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*, ISMIR'08, pages 337–342, 2008.

[26] R. Mayer and A. Rauber. Music genre classification by ensembles of audio and lyrics features. In *Proceedings of the 12th International Conference on Music Information Retrieval*, ISMIR'11, pages 675–680, 2011.

[27] 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*, ECDL'07, pages 112–123, 2007.

[28] 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*, SMC'16, pages 344–349, 2016.

[29] E. Nichols, D. Morris, S. Basu, and C. Raphael. Relationships between lyrics and melody in popular music. In *Proceedings of the 10th International Conference on Music Information Retrieval*, ISMIR'09, pages 471–476, 2009.

[30] 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 Conference on Music Information Retrieval*, ISMIR'12, pages 463–468, 2012.

[31] J. Ren, J. Shen, and R. J. Kauffman. What makes a music track popular in online social networks? In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW'16 Companion, pages 95–96, 2016.

[32] 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 Conference on Music Information Retrieval*, ISMIR'14, pages 585–590, 2014.

[33] G. Sharma and M. N. Murty. Mining sentiments from songs using latent Dirichlet allocation. In *Proceedings of the 10th International Conference on Advances in Intelligent Data Analysis X*, IDA'11, pages 328–339, 2011.

[34] A. Singhi and D. G. Brown. Are poetry and lyrics all that different? In *Proceedings of the 15th International Conference on Music Information Retrieval*, ISMIR'14, pages 471–476, 2014.

[35] V. Thomas, C. Fremerey, D. Damm, and M. Clausen. Slave: A score-lyrics-audio-video-explorer. In *Proceedings of the 10th International Conference on Music Information Retrieval*, ISMIR'09, pages 717–722, 2009.

[36] J. M. Toivanen, H. Toivonen, and A. Valitutti. Automatical composition of lyrical songs. In *Proceedings of the 4th International Conference on Computational Creativity*, ICCC'13, pages 87–91, 2013.

[37] 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 Conference on Music Information Retrieval*, ISMIR'10, pages 45–50, 2010.

[38] 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 Conference on Music Information Retrieval*, ISMIR'11, pages 765–770, 2011.

[39] B. Wei, C. Zhang, and M. Ogihara. Keyword generation for lyrics. In *Proceedings of the 8th International Conference on Music Information Retrieval*, ISMIR'07, pages 121–122, 2007.

[40] M. Zaanen and P. Kanters. Automatic mood classification using TF*IDF based on lyrics. In *Proceedings of the 11th International Conference on Music Information Retrieval*, ISMIR'10, pages 75–80, 2010.

MULTI-PITCH DETECTION AND VOICE ASSIGNMENT FOR *A CAPPELLA* RECORDINGS OF MULTIPLE SINGERS

Rodrigo Schramm[1,3*], **Andrew McLeod**[2*], **Mark Steedman**[2], **Emmanouil Benetos**[3]
[1] Department of Music, Universidade Federal do Rio Grande do Sul, Brazil
[2] School of Informatics, University of Edinburgh, UK
[3] Centre for Digital Music, Queen Mary University of London, UK
rschramm@ufrgs.br, A.McLeod-5@sms.ed.ac.uk, steedman@inf.ed.ac.uk,
emmanouil.benetos@qmul.ac.uk

ABSTRACT

This paper presents a multi-pitch detection and voice assignment method applied to audio recordings containing *a cappella* performances with multiple singers. A novel approach combining an acoustic model for multi-pitch detection and a music language model for voice separation and assignment is proposed. The acoustic model is a spectrogram factorization process based on Probabilistic Latent Component Analysis (PLCA), driven by a 6-dimensional dictionary with pre-learned spectral templates. The voice separation component is based on hidden Markov models that use musicological assumptions. By integrating the models, the system can detect multiple concurrent pitches in vocal music and assign each detected pitch to a specific voice corresponding to a voice type such as soprano, alto, tenor or bass (SATB). This work focuses on four-part compositions, and evaluations on recordings of Bach Chorales and Barbershop quartets show that our integrated approach achieves an F-measure of over 70% for frame-based multi-pitch detection and over 45% for four-voice assignment.

1. INTRODUCTION

Automatic music transcription is defined as the process of converting an acoustic music signal into some form of music notation [3]. In the past years, several signal processing and machine learning approaches have been proposed for automatic music transcription, with applications in music information retrieval, music education, computational musicology, and interactive music systems. A core problem of automatic transcription is multi-pitch detection, i.e. the detection of multiple concurrent pitches.

For multi-pitch detection, spectrogram factorization methods have been used extensively in the last decade [3]. However, despite promising results of template-based techniques [4, 11, 17], the considerable variation in the spectral shape of pitches produced by different sources can still affect generalization performance. Recent research on multi-pitch detection has also focused on deep learning approaches: in [13, 22], feedforward, recurrent and convolutional neural networks were evaluated towards the problem of automatic piano transcription.

On approaches for automatic transcription of vocal music, Bohak and Marolt [5] proposed a method for transcribing folk music containing both instruments and vocals, which explores the repetitions of melodic segments using a musicological model for note-based transcription. A less explored type of music is *a cappella*; in particular, vocal quartets constitute a traditional form of Western music, typically dividing a piece into multiple vocal parts such as soprano, alto, tenor, and bass (SATB). In [21], an acoustic model based on spectrogram factorisation was proposed for multi-pitch detection of such vocal quartets.

A small group of methods have attempted to go beyond multi-pitch detection, towards *instrument assignment* (also called timbre tracking) [1, 8, 11], where a system detects multiple pitches and assigns each pitch to a specific source that produced it. Bay et al. [1] tracked individual instruments in polyphonic instrumental music using a spectrogram factorisation approach with continuity constraints controlled by a hidden Markov model (HMM).

An emerging area of automatic music transcription attempts to combine *acoustic models* (i.e. based on audio information only) with *music language models*, which model sequences of notes and other music cues based on knowledge from music theory or from constraints automatically derived from symbolic music data. This is in direct analogy to automatic speech recognition systems, which typically combine an acoustic model with a spoken language model. An example of such an integrated system is the work by Sigtia et al. [22] which combined neural network-based acoustic and music language models for multi-pitch detection in piano music.

Combining instrument assignment with this idea of using a music language model, it is natural to look at the field of voice separation, which is the separation of pitches into monophonic streams of notes, called voices, mainly addressed in the context of symbolic music pro-

* Authors 1 and 2 contributed equally to this work.

cessing [6, 14, 16]. Several voice separation approaches are based on voice leading rules, which were investigated in [12,23,24] from a cognitive perspective. Among the numerous rules pointed out by these authors, common characteristics are that large melodic intervals between consecutive notes within a single voice should be avoided and that two voices should not cross in pitch. A third principle suggested by [12] is the idea that the stream of notes should be relatively continuous within a single voice, and not have too many gaps of silence, ensuring temporal continuity.

The overarching aim of this work is to create a system able to detect multiple pitches in polyphonic vocal music and assign each detected pitch to a single voice of a specific voice type (e.g. soprano, alto, tenor, bass). Thus, the proposed method is able to perform both multi-pitch detection and voice assignment. Our approach uses an acoustic model for multi-pitch detection based on probabilistic latent component analysis (PLCA), which is modified from the model proposed in [21], and a music language model for voice assignment based on the HMM proposed in [16]. Although previous work has integrated musicological information for note event modelling [5, 19, 22], to the authors' knowledge, this is the first attempt to incorporate an acoustic model with a music language model for the task of voice or instrument assignment from audio, as well as the first attempt to propose a system for voice assignment in polyphonic *a cappella* music. The approach described in this paper focuses on recordings of singing performances by vocal quartets without instrumental accompaniment; to that end we use two datasets containing *a capella* recordings of Bach Chorales and Barbershop quartets. The proposed system is evaluated both in terms of multi-pitch detection and voice assignment, where it reaches an F-measure of 70% and 45% for the two respective tasks.

2. PROPOSED METHOD

In this section, we present a system for multi-pitch detection and voice assignment from audio recordings of polyphonic vocal music where the number of voices is known a priori, that integrates an acoustic model with a music language model. First, we describe the acoustic model, a spectrogram factorization process based on probabilistic latent component analysis (PLCA). Then, we present the music language model, an HMM-based voice assignment model. Finally, a joint model is proposed for the integration of these two components. Figure 1 illustrates the proposed system pipeline.

2.1 Acoustic Model

The acoustic model is a variant of the spectrogram factorisation-based model proposed in [21]. The model uses a fixed dictionary of log-spectral templates and aims to decompose an input time-frequency representation into several components denoting the activations of pitches, voice types, tuning deviations, singer subjects, and vowels. As time-frequency representation we use a normalised variable-Q transform (VQT) spectrogram [20] with a hop

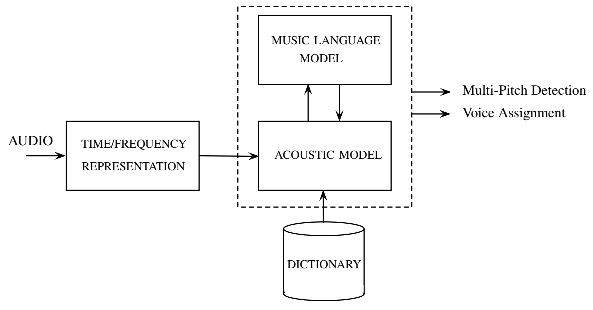


Figure 1: Proposed system diagram.

size of 20 msec and 20 cent resolution.

The input VQT spectrogram is denoted as $X_{\omega,t} \in \mathbb{R}^{\Omega\times T}$, where ω denotes log-frequency and t time. In the model, $X_{\omega,t}$ is approximated by a bivariate probability distribution $P(\omega,t)$, which is in turn decomposed as:

$$P(\omega,t) = \qquad (1)$$
$$P(t) \sum_{s,p,f,o,v} \Phi P_t(s|p)P_t(f|p)P_t(o|p)P(v)P_t(p|v)$$

where $P(t)$ is the spectrogram energy (known quantity). $\Phi = P(\omega|s,p,f,o,v)$ is the fixed pre-extracted spectral template dictionary (for details about the dictionary construction, refer to [21]). Variable $p \in \{21,...,108\}$ denotes pitch in MIDI scale, s denotes the singer index (out of the collection of singer subjects used to construct the input dictionary), o denotes the vowel type, v denotes the voice type (e.g. soprano, alto, tenor, bass), and f denotes tuning deviation from 12-tone equal temperament in 20 cent resolution ($f \in \{1,\dots,5\}$, with $f = 3$ denoting ideal tuning). Unlike in [21], this model decomposes the probabilities of pitch and voice type as $P(v)P_t(p|v)$. That is, $P_t(p|v)$ denotes the pitch activation for a specific voice type (eg. SATB) over time and $P(v)$ can be viewed as a mixture weight that denotes the overall contribution of each voice type to the whole input recording. The contribution of specific singer subjects from the training dictionary is modelled by $P_t(s|p)$, i.e. the singer contribution per pitch over time. $P_t(f|p)$ is the tuning deviation per pitch over time and finally $P_t(o|p)$ is the time-varying vowel contribution per pitch [1].

The factorization can be achieved by the expectation-maximization (EM) algorithm [7], where the unknown model parameters $P_t(s|p)$, $P_t(f|p)$, $P_t(o|p)$, $P_t(p|v)$, and $P(v)$ are iteratively estimated. In the *Expectation* step we compute the posterior as:

$$P_t(s,p,f,o,v|\omega) = \qquad (2)$$
$$\frac{\Phi P_t(s|p)P_t(f|p)P_t(o|p)P(v)P_t(p|v)}{\sum_{s,p,f,o,v} \Phi P_t(s|p)P_t(f|p)P_t(o|p)P(v)P_t(p|v)}$$

In the *Maximization* step, each unknown model parameter is then updated using the posterior from Eqn (2):

[1] Although $P_t(o|p)$ is not explicitly used in this proposed approach, it is kept to ensure consistency with the RWC audio dataset structure.

$$P_t(s|p) \propto \sum_{f,o,v,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \quad (3)$$

$$P_t(f|p) \propto \sum_{s,o,v,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \quad (4)$$

$$P_t(o|p) \propto \sum_{s,f,v,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \quad (5)$$

$$P_t(p|v) \propto \sum_{s,f,o,\omega} P_t(s,p,f,o,v|\omega)X_{\omega,t} \quad (6)$$

$$P(v) \propto \sum_{s,f,o,p,\omega,t} P_t(s,p,f,o,p|\omega)X_{\omega,t} \quad (7)$$

The model parameters are randomly initialised, and the EM algorithm iterates over Eqns (2)-(7). In our experiments we use 30 iterations.

The output of the acoustic model is a semitone-scale pitch activity tensor for each voice type and a pitch shifting tensor, given by $P(p,v,t) = P(t)P(v)P_t(p|v)$ and $P(f,p,v,t) = P(t)P(v)P_t(p|v)P_t(f|p)$ respectively. By stacking together slices of $P(f,p,v,t)$ for all values of p, we can create a 20 cent-resolution time-pitch representation for each voice type v:

$$P(f',t,v) = [P(f,21,v,t)...P(f,108,v,t)] \quad (8)$$

where $f' = 1,...,880$ denotes pitch in 20 cent resolution. The overall multi-pitch detection without voice assignment, is given by $P(p,t) = \sum_v P(p,v,t)$. Finally, the voice-specific pitch activation output $P(p,v,t)$ is binarized and post-processed through a refinement step described in [21], where each pitch is aligned with the nearest peak to it in the input log-frequency spectrum.

2.2 Music Language Model

The music language model attempts to assign each detected pitch to a single voice based on musicological constraints. It is a variant of the HMM-based approach proposed in [16], where the main change is to the emission function (here it is probabilistic, while in the previous work it was deterministic). The model separates sequential sets of multi-pitch activations into monophonic voices (of type SATB) based on three principles: (1) consecutive notes within a voice tend to occur on similar pitches; (2) there are minimal temporal gaps between them; and (3) voices are unlikely to cross.

The observed data for the HMM are notes generated from the acoustic model's binarised multi-pitch activations $P(p,t)$, where each generates a note n with pitch $\rho(n) = p$, onset time $\delta(n) = t$, and an offset time $\tau(n) = t+1$. O_t represents this observed data at frame t.

2.2.1 HMM: State Space

In the HMM, a state S_t at frame t contains a list of M monophonic voices V_i, $1 \leq i \leq M$. The initial state S_0 contains M empty voices, and at each frame, each voice is assigned either no note, or a note with pitch $\rho(n) \in \{21,...,108\}$. Each voice contains the entire history of the notes which have been assigned to it from frame 1 to t. The state space of our model blows up exponentially (though it is reduced significantly when the model is run discriminatively as we do),, so instead of precomputed transition and emission probabilities, we use transition and emission probability functions, presented in the following sections.

Conceptually, it is helpful to think of each state as simply a list of M voices, rather than to consider each voice to also be a list of notes. Thus, each state transition is calculated based on each voice in the previous state (though some of the probability calculations require knowledge of individual notes).

2.2.2 HMM: Transition Function

A state S_{t-1} has a transition to state S_t if and only if each voice $V_i \in S_{t-1}$ can be transformed into the corresponding $V_i \in S_t$ by assigning to it up to 1 note with onset time t.

This transition from S_{t-1} to S_t can be represented by the variable T_{S_{t-1},N_t,W_t}, where S_{t-1} is the original state, N_t is a list of those notes n contained by any voice in S_t where $\delta(n) = t$, and W_t is a list of integers, each representing the voice assignment index for a single note $n \in N_t$. For each index i, $1 \leq i \leq |N_t| = |W_t|$, note n_i is assigned to voice $V_{w_i} \in S_t$. Here, N_t only contains those observed notes which are assigned to a voice in S_t, not all observed notes. Since all of our voices are monophonic, no two elements in W_t may be equal.

We now define the HMM transition probability P($S_t|S_{t-1}$) as P(T_{S_{t-1},N_t,W_t}):

$$P(T_{S_{t-1},N_t,W_t}) = \Psi(W_t)\prod_{1\leq i\leq|N_t|}\Theta(S_{t-1},n_i,w_i)\Lambda(V_{w_i},n_i). \quad (9)$$

The first term in this product is defined as

$$\Psi(W) = \prod_{1\leq j\leq M}\begin{cases}\Upsilon & j \in W\\ 1-\Upsilon & j \notin W\end{cases} \quad (10)$$

where the parameter Υ represents the probability that a given voice contains any note in a frame.

$\Theta(S_{t-1},n,w)$ is a penalty function used to minimize the voice crossings. It returns by default 1, but its output is multiplied by a parameter θ—representing the probability of a voice being out of pitch order with an adjacent voice—for each of the following cases that applies:

1. $w > 1$ and $\chi(V_{w-1}) > \rho(n)$
2. $w < |M|$ and $\chi(V_{w+1}) < \rho(n)$

$\chi(V)$ represents the pitch of a voice, calculated as a weighted sum of the pitches of its most recent notes. Cases 1 and 2 apply when a note is out of pitch order with the preceding or succeeding voice in the state respectively.

$\Lambda(V,n)$ is used to calculate the probability of a note n being assigned to a voice V, and is the product of a pitch score Δ_p and a gap score Δ_g:

$$\Lambda(V,n) = \Delta_p(V,n)\Delta_g(V,n) \quad (11)$$

The pitch score, used to minimise melodic jumps within a voice, is computed as shown in Eqn (12), where $\mathcal{N}(\mu,\sigma)$

represents a normal distribution with mean μ and standard deviation σ, and σ_p is a parameter. The gap score is used to prefer temporal continuity within a voice, and is computed using Eqn (13), where $\tau(V)$ is the offset time of the most recent note in V and σ_g and g_{min} are parameters. Both Δ_p and Δ_g return 1 if V is empty.

$$\Delta_p(V, n) = \mathcal{N}(\rho(n) - \chi(V), \sigma_p) \tag{12}$$

$$\Delta_g(V, n) = \max\left(\ln\left(-\frac{\delta(n) - \tau(V)}{\sigma_g} + 1\right) + 1, g_{min}\right) \tag{13}$$

2.2.3 HMM: Emission Function

A state S_t emits a set of notes containing only those which have an onset at frame t, and a state containing a voice with a note at frame t must emit that note. The probability of a state S_t emitting the note set O_t is shown in Eqn (14), using the voice posterior $P_t(v|p)$ from the acoustic model.

$$\mathrm{P}(O_t|S_t) = \prod_{n \in O_t} \begin{cases} P_t(v = i|p = \rho(n)) & n \in V_i \in S_t \\ 1 & \text{otherwise} \end{cases} \tag{14}$$

A state is not penalised for emitting notes not assigned to any of its voices. This allows the model to better handle false positives from the multi-pitch detection. For example, if the acoustic model detects more than M pitches, we allow a state to emit the corresponding notes without penalty. We do, however, penalise a state for not assigning a voice any note during a frame, but this is handled by $\Psi(W)$ from Eqn (10).

2.2.4 HMM: Inference

To find the most likely final state given our observed note sets, we use the Viterbi algorithm [26] with beam search with beam size b. That is, after each iteration, we save only the $b = 50$ most likely states given the observed data to that point, in order to handle the complexity of the HMM.

2.3 Model Integration

In this section, we describe the integration of the acoustic model and the music language model into a single system which jointly performs multi-pitch detection and voice assignment from audio. This integration is done in two stages. First, using only the acoustic model from subsection 2.1, the EM algorithm is run for 15 iterations, when the multi-pitch detections converge. Next, the system runs for 15 more EM iterations, this time also using the music language model from subsection 2.2. In each iteration, the acoustic model is run first, and then the language model is run on the resulting multi-pitch detections. To intergrate the two models, we apply a fusion mechanism inspired by the one used in [9] to improve the acoustic model's pitch activations based on the resulting voice assignments.

The output of the language model is introduced into the acoustic model as a prior to $P_t(p|v)$. During the acoustic model's EM updates, Eqn (6) is modified as:

$$P_t^{new}(p|v) = \alpha P_t(p|v) + (1 - \alpha)\phi_t(p|v), \tag{15}$$

where α is a weight parameter controlling the effect of the acoustic and language model and ϕ is a hyperparameter defined as:

$$\phi_t(p|v) \propto P_t^a(p|v) P_t(p|v). \tag{16}$$

$P_t^a(p|v)$ is calculated from the most probable final HMM state $S_{t_{max}}$ using the pitch score $\Delta_p(V, n)$ from the HMM transition function of Eqn (12). For V, we use the voice $V_v \in S_{t_{max}}$ as it was at frame $t - 1$, and for n, we use a note at pitch p. The probability values are then normalised over all pitches per voice. The pitch score returns a value of 1 when the V is an empty voice (thus becoming a uniform distribution over all pitches). The hyperparameter of Eqn (16) acts as a soft mask, reweighing the pitch contribution of each voice regarding only the pitch neighbourhood previously detected by the model.

The final output of the integrated system is a list of the detected pitches at each time frame which are assigned to a voice in the most probable final HMM state $S_{t_{max}}$, along with the voice assignment for each. Figure 2 shows an example output of the integrated system.

3. EVALUATION

3.1 Datasets

We evaluate the proposed model on two datasets of *a capella* recordings [2] : one of 26 Bach Chorales and another of 22 Barbershop quartets, in total 104 minutes. These are the same datasets used in [21], allowing for a direct comparison between it and the acoustic model proposed in Section 2.1. Each file is in wave format with a sample rate of 22.05 kHz and 16 bits per sample. Each recording has four distinct vocal parts (SATB), with one part per channel. The recordings from the Barbershop dataset each contain four male voices, while the Bach Chorale recordings each contain a mixture of two male and two female voices. A frame-based pitch ground truth for each vocal part was extracted using a monophonic pitch tracking algorithm [15] on each individual monophonic track. Experiments are conducted using the mix down of each audio file (i.e. polyphonic content), not the individual tracks.

3.2 Evaluation Metrics

We evaluate the proposed system on both multi-pitch detection and voice assignment using the frame-based precision, recall and F-measure as defined in the MIREX multiple-F0 estimation evaluations [2], with a frame hop size of 20 ms. The F-measure obtained by the multi-pitch detection is denoted as F_{mp}, and for this, we combine the individual voice ground truths into a single ground truth for each recording. For voice assignment, we simply use the individual voice ground truths and define voice-specific F-measures of F_s, F_a, F_t, and F_b for each respective SATB vocal part. We also define an overall voice assignment F-measure F_{va} for a given recording as the arithmetic mean of its four voice-specific F-measures.

[2] Original recordings available at `http://www.pgmusic.com/{bachchorales.htm|barbershopquartet.htm}`.

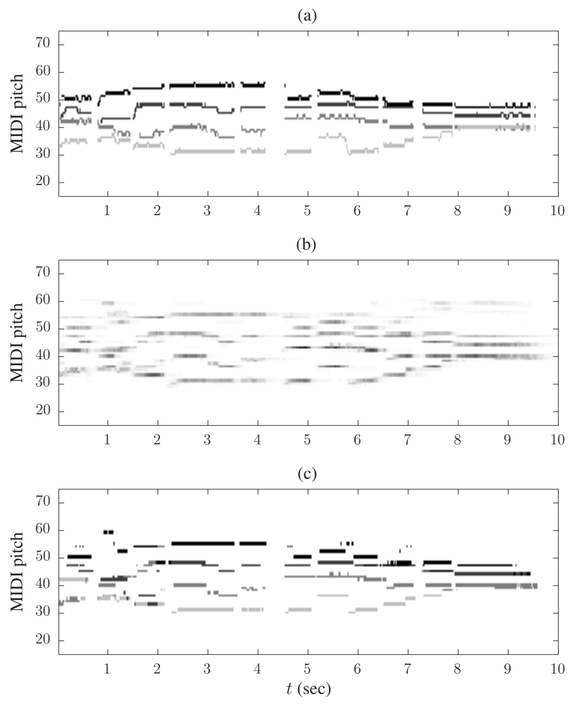


Figure 2: Multi-pitch detection and voice assignment for a 10-sec excerpt of "O Sacred Head Sore Wounded" from the Bach Chorales dataset. Each vocal part is shown in a distinct shade of grey. (a) Ground truth. (b) Pitch activation $P(p, t)$. (c) Output of the integrated system.

3.3 Training

To train the acoustic model, we use recordings from the RWC dataset [10] to generate the 6-dimensional dictionary of log-spectral templates specified in Section 2.1, following the procedure described in [21]. The recordings used to generate the dictionary contain sequences of notes following a chromatic scale, in five distinct English vowels (/a/, /æ/, /i/, /ɒ/, /u/). The dictionary contains templates generated from 15 distinct singers (9 male and 6 female, consisting of 3 human subjects for each voice type: bass, baritone, tenor, alto, soprano).

For all parameters in the music language model, we use the values reported in [16] that were used for voice separation in fugues. We also introduce two new parameters to the system: the voice order probability θ and the voice probability Υ. We use MIDI files of 50 Bach Chorales [3] (none of which appear in the test set), splitting the notes into 20 ms frames, and measure the proportion of frames in which a voice was out of pitch order with another voice, and the proportion of frames in which each voice contains a note. This results in values of $\theta = 0.006$ and $\Upsilon = 0.99$, which we use for testing.

To train the model integration weight α, we use a grid search on the range $[0.1, 0.9]$ with a step size of 0.1, maximising F_{va} for each dataset. This results in a value of 0.6 when trained on the Chorale recordings and 0.3 when trained on the Barbershop recordings. To avoid overfitting, we employ cross-validation, using the α value that maximises the Chorales' F_{va} when evaluating the Barbershop quartets, and vice versa.

[3] MIDI files available at `http://kern.ccarh.org/`.

3.4 Results

We compare our model's multi-pitch detection results with those of three baseline methods: VINC+ [25], which uses an adaptive spectral decomposition based on unsupervised NMF; PERT+ [18], which selects candidates among spectral peaks, validating candidates through additional audio descriptors; and MSINGERS†+ [21], a PLCA model for multi-pitch detection from multi-singers, similar to the acoustic model of our proposed system, although it also includes a binary classifier to estimate the final pitch detections from the pitch activations. To the authors' knowledge, there is no existing system for multi-pitch detection and voice assignment that can be used as a baseline for our model's voice assignment. However, for the sake of comparison, we include results from voice assignments derived from the model proposed in [21], which we call MSINGERS-VA, despite the fact that the original model was not designed for the task.

We evaluate the above systems against two versions of our proposed model: VOCAL4-MP, using only the acoustic model described in Section 2.1; and VOCAL4-VA, using the fully integrated model. From the multi-pitch detection results in Table 1, it can be seen that MSINGERS†+ achieves the highest F_{mp} on the Bach chorales, narrowly edging out VOCAL4-VA, but VOCAL4-VA achieves state-of-the-art results on the Barbershop quartets. In both datasets, VOCAL4-VA outperforms VOCAL4-MP substantially, indicating that the music language model is able to drive the acoustic model to a more meaningful factorisation. The voice assignment results are shown in Table 2, where it is clear that VOCAL4-VA outperforms the other models, suggesting that perhaps a language model is almost necessary for the task. Also interesting to note is that it performs significantly better on the bass voice than on the other voices. Overtones are a major source of errors in our model, and the bass voice avoids these since it is almost always the lowest voice.

A further investigation into our model's performance can be found in Figure 3, which shows all of the VOCAL4-VA model's F-measures, averaged across all songs in the corresponding dataset after each EM iteration. The first thing to notice is the large jump in performance at iteration 15, when the language model is first integrated into the process. This jump is most significant for voice assignment, but is also clear for multi-pitch detection. The main source of the improvement in multi-pitch detection is that the music language model helps to eliminate many false positive pitch detections using the integrated pitch prior. In fact, the multi-pitch detection performance continues to improve until it finally converges after iteration 30.

The voice assignment results, however, are less straightforward. After the significant improvement on the 15th iteration, the results either remain relatively stable (in the Barbershop quartets) or even drop slightly (in the Bach

Model	Bach Chorales	Barbershop Quartets
VINC+	53.58	51.04
PERT+	67.19	63.85
MSINGERS†+	**70.84**	71.03
VOCAL4-MP	63.05	59.09
VOCAL4-VA	69.66	**73.46**

Table 1: Multi-pitch detection results.

Model	Bach Chorales				
	F_{va}	F_s	F_a	F_t	F_b
MSINGERS-VA	18.02	15.37	17.59	26.32	12.81
VOCAL4-MP	21.84	12.99	10.27	22.72	41.37
VOCAL4-VA	45.31	26.07	37.63	49.61	67.94
Model	Barbershop Quartets				
	F_{va}	F_s	F_a	F_t	F_b
MSINGERS-VA	12.29	9.70	14.03	27.93	9.48
VOCAL4-MP	18.35	2.40	10.56	16.61	43.85
VOCAL4-VA	46.92	40.01	35.57	29.76	82.34

Table 2: Voice assignment results.

chorales) before convergence. This slight drop is due to the fact that the language model initially receives noisy multi-pitch detections that include false positives (mainly overtones). Incorporating these overtones into the voice assignment can cause the removal of correct pitch detections, which in turn reduces the voice assignment F-measures.

As mentioned earlier, the bass voice assignment outperforms all other voice assignments in almost all cases, since false positive pitch detections from the acoustic model often correspond with overtones from lower notes that occur in the same pitch range as the correct notes from higher voices. Another common source of errors (for both multi-pitch detection and voice assignment) is vibrato. The acoustic model can have trouble detecting vibrato, and the music language model prefers voices with constant pitch over voices alternating between two pitches, leading to many off-by-one errors in pitch detection. An example of both of these types of errors can be found in Figure 4.

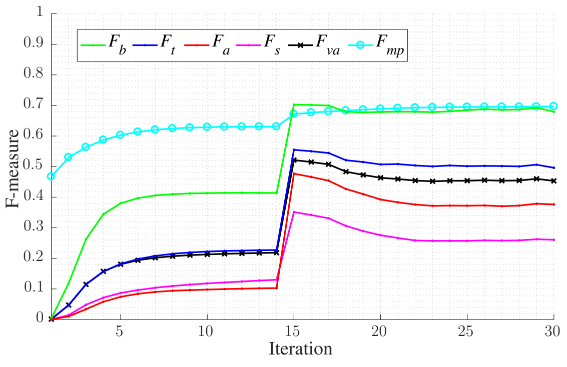
4. CONCLUSION

In this paper, we have presented a system for multi-pitch detection and voice assignment for *a cappella* recordings of multiple singers. It consists of two integrated components: a PLCA-based acoustic model and an HMM-based music language model. To our knowledge, ours is the first system to be designed for the task[4].

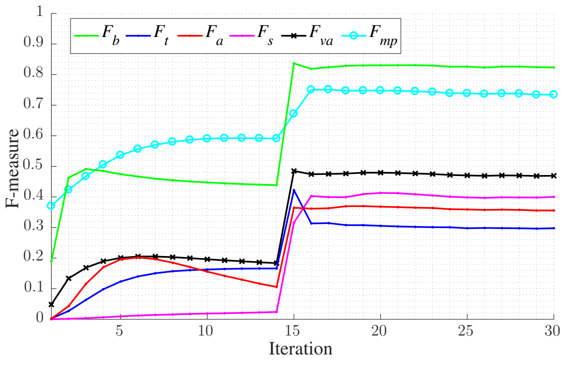
We have evaluated our system on both multi-pitch detection and voice assignment on two datasets: one of Bach chorales, and another of Barbershop quartets. Our model outperforms baseline multi-pitch detection systems on the Barbershop quartets, and achieves near state-of-the-art performance on the chorales. We have shown that integrating the music language model improves multi-pitch detection performance compared with a simpler version of our system with only the acoustic model. This suggests, as has been shown in previous work, that incorporating such music language models into other acoustic MIR tasks might also be of some benefit, since they can guide acoustic models using musicological principles.

We also presented results for voice assignment, and show that while our model performs well given the difficulty of the task, there is certainly room for improvement. Avenues for future work include a better handling of overtones in the acoustic model, and better recognition of vibrato in both the acoustic and the music language model. We will also investigate the use of timbral information for further improving voice assignment performance. Additionally, our model could be applied to different styles of music (e.g., instrumental, or those containing both instruments and vocals) by learning a new dictionary for the acoustic model and retraining the parameters of the music language model, and we intend to investigate the generality of our model in that context.

[4] Supporting material for this work is available at `http://inf.ufrgs.br/~rschramm/projects/msingers`



(a)



(b)

Figure 3: The VOCAL4-VA model's F-measures after each EM iteration, averaged across all songs in each dataset: (a) Bach Chorales. (b) Barbershop Quartets.

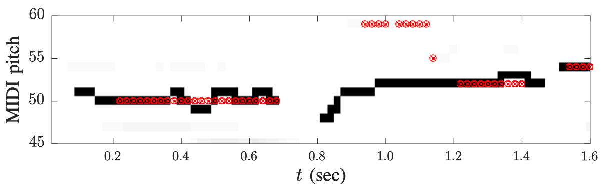


Figure 4: Pitch detections (red) and ground truth (black) for the soprano voice at the beginning of the excerpt from Figure 2, showing errors from both overtones and vibrato.

5. ACKNOWLEDGEMENT

RS is supported by a UK Newton Research Collaboration Programme Award (grant no. NRCP1617/5/46). AM and MS are supported by a gift from the 2017 Bloomberg Data Science Research Grant program and EU ERC H2020 Advanced Fellowship GA 742137 SEMANTAX. EB is supported by a UK Royal Academy of Engineering Research Fellowship (grant no. RF/128).

6. REFERENCES

[1] M. Bay, A. F. Ehmann, J. W. Beauchamp, P. Smaragdis, and J. Stephen Downie. Second fiddle is important too: Pitch tracking individual voices in polyphonic music. In *ISMIR*, pages 319–324, 2012.

[2] M. Bay, A. F. Ehmann, and J. S. Downie. Evaluation of multiple-F0 estimation and tracking systems. In *ISMIR*, pages 315–320, October 26-30 2009.

[3] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: challenges and future directions. *J. Intell. Inf. Syst.*, 41(3):407–434, 2013.

[4] E. Benetos and T. Weyde. An efficient temporally-constrained probabilistic model for multiple-instrument music transcription. In *ISMIR*, pages 701–707, 2015.

[5] C. Bohak and M. Marolt. Transcription of polyphonic vocal music with a repetitive melodic structure. *J. Audio Eng. Soc*, 64(9):664–672, 2016.

[6] E. Cambouropoulos. Voice and stream: Perceptual and computational modeling of voice separation. *Music Perception: An Interdisciplinary Journal*, 26(1):75–94, 2008.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society*, 39(1):1–38, 1977.

[8] Z. Duan, J. Han, and B. Pardo. Multi-pitch streaming of harmonic sound mixtures. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):138–150, Jan 2014.

[9] D. Giannoulis, E. Benetos, A. Klapuri, and M. D. Plumbley. Improving instrument recognition in polyphonic music through system integration. In *ICASSP*, pages 5222–5226, 2014.

[10] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Music genre database and musical instrument sound database. In *ISMIR*, pages 229–230, 2004.

[11] G. Grindlay and D. P. W. Ellis. Transcribing multi-instrument polyphonic music with hierarchical eigeninstruments. *IEEE J. Selected Topics in Signal Processing*, 5(6):1159–1169, October 2011.

[12] D. Huron. Tone and voice: A derivation of the rules of voice-leading from perceptual principles. *Music Perception*, 19(1):1–64, 2001.

[13] R. Kelz, M. Dorfer, F. Korzeniowski, S. Böck, A. Arzt, and G. Widmer. On the potential of simple framewise approaches to piano transcription. In *ISMIR*, pages 475–481, 2016.

[14] P. B. Kirlin and P. E. Utgoff. VOISE: learning to segregate voices in explicit and implicit polyphony. In *ISMIR*, pages 552–557, 2005.

[15] M. Mauch and S. Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *ICASSP*, pages 659–663, 2014.

[16] A. McLeod and M. Steedman. HMM-based voice separation of MIDI performance. *Journal of New Music Research*, 45(1):17–26, 2016.

[17] G. J. Mysore and P. Smaragdis. Relative pitch estimation of multiple instruments. In *ICASSP*, pages 313–316, 2009.

[18] A. Pertusa and J. M. Iñesta. Efficient methods for joint estimation of multiple fundamental frequencies in music signals. *EURASIP Journal on Advances in Signal Processing*, 2012.

[19] M. P. Ryynanen and A. Klapuri. Polyphonic music transcription using note event modeling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, pages 319–322, Oct 2005.

[20] C. Schörkhuber, A. Klapuri, N. Holighaus, and M. Dörfler. A Matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution. In *AES 53rd Conference on Semantic Audio*, January 2014.

[21] R. Schramm and E. Benetos. Automatic transcription of a cappella recordings from multiple singers. In *AES International Conference on Semantic Audio*, June 2017.

[22] S. Sigtia, E. Benetos, and S. Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, May 2016.

[23] D. Temperley. A probabilistic model of melody perception. *Cognitive Science*, 32(2):418–444, 2008.

[24] D. Tymoczko. Scale theory, serial theory and voice leading. *Music Analysis*, 27(1):1–49, 2008.

[25] E. Vincent, N. Bertin, and R. Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Trans. Audio, Speech, and Lang. Processing*, 18(3):528–537, March 2010.

[26] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

ONSET DETECTION IN COMPOSITION ITEMS OF CARNATIC MUSIC

Jilt Sebastian
Indian Institute of Technology, Madras
jiltsebastian@gmail.com

Hema A. Murthy
Indian Institute of Technology, Madras
hema@cse.itm.ac.in

ABSTRACT

Complex rhythmic patterns associated with Carnatic music are revealed from the *stroke* locations of percussion instruments. However, a comprehensive approach for the detection of these locations from composition items is lacking. This is a challenging problem since the melodic sounds (typically vocal and violin) generate soft-onset locations which result in a number of false alarms.

In this work, a separation-driven onset detection approach is proposed. Percussive separation is performed using a Deep Recurrent Neural Network (DRNN) in the first stage. A single model is used to separate the percussive vs the non-percussive sounds using discriminative training and time-frequency masking. This is then followed by an onset detection stage based on group delay (GD) processing on the separated percussive track. The proposed approach is evaluated on a large dataset of live Carnatic music concert recordings and compared against percussive separation and onset detection baselines. The separation performance is significantly better than that of Harmonic-Percussive Separation (HPS) algorithm and onset detection performance is better than the state-of-the-art Convolutional Neural Network (CNN) based algorithm. The proposed approach has an absolute improvement of 18.4% compared with the detection algorithm applied directly on the composition items.

1. INTRODUCTION

Detecting and characterizing musical events is an important task in Music Information Retrieval (MIR), especially in Carnatic music, which has a rich rhythm repertoire. There are seven different types of repeating rhythmic patterns known as *tāla*s, which when combined with 5 *jāti*s give rise to 35 combinations of rhythmic cycles of fixed intervals. By incorporating 5 further variations called *gati*/*nadai*, 175 rhythmic cycles are obtained [13]. A *tāla* cycle is made up of *mātrā*s, which in turn are made up of *akshara̅*s or strokes at the fundamental level. Another complexity in Carnatic music is that the start of the *tāla* cycle and of the composition need not be synchronous. Nevertheless, percussion keeps track of *rhythm*. The detection of percussive syllable locations aids higher level retrieval tasks such as *aksharā* transcription, *sama* (start of *tāla*) and *eḍuppu* (start of composition) detection and *tāla* tracking.

Various methods have been proposed for detecting onsets from music signals using a short-term signal, the linear prediction error signal, spectral magnitude or phase, energy and their combination [1, 3, 11, 14, 15]. In [2], various acoustic features are analyzed for this task and in [7], spectral methods are modified to enable onset detection. These and other algorithms are analyzed in detail in [5]. Recent efforts include the use of Recurrent (RNN) [17] and Convolutional Neural Networks (CNN) [19] for onset detection. All of the above techniques are primarily for the detection of monophonic musical onsets.

Every item in Carnatic music has, at its core, a composition. Every item in a concert is characterized by three sections. A lyrical composition section that is performed together by the lead performer, accompanying violinist and the percussion artist. This section is optionally preceded by a pure melody section (*ālāpana*) in which only the lead performer and the accompanying violinist perform. The composition section is optionally followed by a pure percussion section (*tani āvarthanam*). Onset detection and *aksharā* transcription in *tani āvarthanam*s are performed in [15], and [16] respectively. Percussive onset detection for an entire concert that is made up of 10-12 items, each associated with its own *tāla* cycle, is still challenging as the composition items are made up of ensembles of a lead vocal, violin/ensembles of the lead instrument(s) and percussion.

Onset detection in polyphonic music/ensemble of percussion either use audio features directly [4], or performs detection on the separated sources. Dictionary learning-based methods using templates are employed in the separation stage in certain music traditions [10, 22]. Harmonic/percussive separation (HPS) from the audio mixture is successfully attempted on Western music in [8] and [9]. Onset detection of notes is performed on polyphonic music in [4] for transcription. Efficient percussive onset detection on monaural music mixtures is still a challenging problem. The current approaches lead to a significant number of false positives, owing to the difficulty in detecting only the percussive syllables with varying amplitudes and the presence of melodic voices.

In a Carnatic music concert, the lead artist and all the accompanying instruments are tuned to the same base frequency called *tonic* frequency and it may vary for each concert. This leads to the overlapping of pitch trajectories. The bases do not vary over time in the case of dictionary-based separation methods, leading to a limited performance in Carnatic music renderings. HPS model [8] does not account for the melodic component and variation of *tonic* across the concerts. The state-of-the-art solo onset detection techniques, when applied to the polyphonic music, perform poorer ($\approx$ 20% absolute) than on the solo samples [22].

In this paper, a separation-driven approach for percussive onset detection is presented. A deep recurrent model (DRNN) is used to separate the percussion from the composition in the first stage. It is followed by the onset detection based on signal processing in the final stage. The proposed approach achieves significant improvement (18.4%) over the onset detection algorithm applied to the mixture and gracefully degrades (about 4.6% poorer) with respect to onset detection on solo percussion. The proposed approach has better separation and detection performance, when compared to that of the baseline algorithms.

2. DATASETS

Multi-track recordings of six live vocal concerts ($\simeq$ 14 hours) are considered for extracting the composition items. These items contain composition segments with vocal and/or violin segments in first track and percussive segments in the second track. To create the ground truth, onsets are marked (manually by the authors) in the percussive track. These onsets are verified by a professional artist [1] . Details of the datasets prepared from various concerts are given in Table 1. The composition items consist of recordings from both male and female artists sampled at 44.1 kHz. Some of the strokes in the mridangam are dependent on the tonic, while others are not. The concerts SS and KD also include *ghatam* and *khanjira*, which are secondary percussion instruments. Recordings are also affected by nearby sources, background applauses and the perpetual *drone*.

Concert	Total Length hh:mm:ss	Comp. Segments mm:ss (Number)	No. of Strokes
KK	2:15:50	1:52 (3)	541
SS	2:41:14	0:38(4)	123
MH	2:31:47	1:16 (3)	329
ND	1:15:20	1:51 (3)	330
MO	2:00:15	7:14 (3)	1698
KD	2:20:23	5:32 (3)	1088
Total	13:41:59	18:23 (19)	4109

Table 1: Details of the dataset

Training examples for the percussion separation stage are obtained from the *ālāpana* (vocal solo, violin solo) and mridangam *tani āvarthanam* segments. These are mixed to create the polyphonic mixture. A total of 12 musical clips are extracted from four out of six recordings, to obtain the training set (17min and 5s), and the validation set (4min and 10s). Hence, around 43% of the data is found to be sufficient for training. 10% of the dataset is used for the validation of neural network parameters and the rest for testing the separation performance. The concert segments KK and ND are only used for testing the proposed approach to check the generalizability of the approach across various concerts. The composition segments shown in Table 1 column 3 (with ground truth) are used as the test data. Onset detection is then performed on the separated percussive track.

[1] Thanks to musician Dr. Padmasundari for the verification

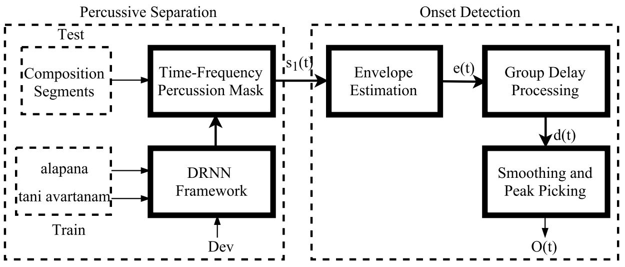


Figure 1: Block diagram of the proposed approach.

3. PROPOSED APPROACH

The proposed method consists of two stages: percussive separation stage and solo onset detection stage. Initially, the time-frequency masks specific to percussive voices (mainly mridangam) are learned using a DRNN framework. The separated percussion source is then used as input to the onset detection algorithm. Figure 1 shows the block diagram of the overall process which is explained subsequently in detail.

3.1 Percussive Separation Stage

A deep recurrent neural network framework originally proposed for singing voice separation [12] is adopted for separating the percussion from the other voices. *Ālāpana* segments are mixed with *tani āvarthanam* segments for learning the timbral patterns corresponding to each source. Figure 2 shows the time-frequency patterns of the composition mixture segment, melodic mixture and the percussive source in Carnatic music. The patterns associated with different voices are mixed in composition segments leading to a fairly complex magnitude spectrogram (Figure 2 *left*) which makes separation of percussion a nontrivial task. The DRNN architecture for percussive separation stage is shown in Figure 3. The network takes the feature vector corresponding to the composition items (x_t) and estimates the mask corresponding to the percussive ($y_t'^1$) and non-percussive ($y_t'^2$) sources. The normalized mask corresponding to the percussive source ($M_1(f)$) is used to filter the mixture spectrum and then combined with the mixture phase to obtain the complex-valued percussive spectrum:

$$\widehat{S_p}(f) = M_1(f)X_t(f) \tag{1}$$

$$S_p(t) = ISTFT(\widehat{S_p}\angle X_t) \tag{2}$$

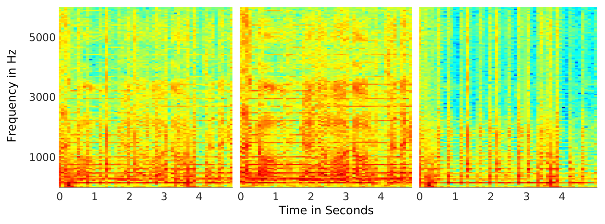


Figure 2: Spectrograms of a segment of composition ***(left)*** obtained from the mixture (KK dataset) containing melodic sources, vocal and violin ***(middle)*** and the percussive source ***(right)***.

where, ISTFT refers to inverse short-time Fourier transform, $\widehat{S_p}$ is the estimated percussive spectrum, $\angle(X_t)$ is the mixture phase at time t and, $S_p(t)$ is the percussive signal estimated for t^{th} time frame.

We use the short-time Fourier transform (STFT) feature as it performs better than conventional features in musical source separation tasks [21]. The regression problem of finding the source specific-magnitude spectrogram is formulated as a binary mask estimation problem where each time-frequency bin is classified as either percussive or non-percussive voice. The network is jointly optimized with the normalized masking function ($M_1(f)$) by adding an extra deterministic layer to the output layer. We use a single model to learn both these masks despite the fact that only percussive sound is required in the second stage. Thus, discriminative information is also used for the learning problem. The objective function (Mean Squared Error) that is minimized is given by:

$$||\widehat{y}_{1t} - y_{1t}||^2 + ||\widehat{y}_{2t} - y_{2t}||^2 - \gamma(||\widehat{y}_{1t} - y_{2t}||^2 + ||\widehat{y}_{2t} - y_{1t}||^2) \tag{3}$$

where $\widehat{y}_t$ and y_t are the estimated and original magnitude spectra respectively. The γ parameter is optimized such that more importance is given to minimizing the error for the percussive voices than maximizing the difference with respect to the other sources. This is primarily to ensure that the characteristics of percussive voice are not affected significantly by separation, as the percussive voice will be used later for onset detection. The recurrent connections are employed to capture the temporal dynamics of the percussive source which are not captured using the contextual windows. The network has a recurrent connection at the second hidden layer and is parametrically chosen based on the performance on development data. The second hidden layer output is calculated from the current input and output of the same hidden layer in the previous time-step as:

$$h^2(x_t) = f(W^2h^2(x_t) + b^2 + V^2h^2(x_{t-1})) \tag{4}$$

where, W and V are the weight matrices, V being the temporal weight matrix and the function $f(\cdot)$ is the ReLU activation [12].

A recurrent network trained with *Ālāpana* and *tani āvarthanam* separates the percussion from the voice by generating a time-frequency percussive mask. This mask

[2] Example redrawn from [12]

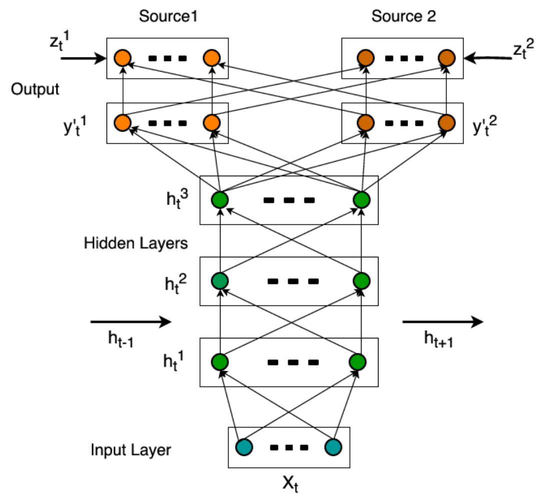


Figure 3: Percussive separation architecture [2]

is used to separate the percussive voice in the composition segment of a Carnatic music item. The separated signal is used for onset detection in the next stage (Figure 1).

3.2 Onset Detection Stage

The separated percussive voice is used as the source signal for the onset detection task. Note that this signal has other source interferences, artifacts and other distortions. The second block in Figure 1 corresponds to the onset detection stage. Onset detection consists of two steps. In the first step a detection function is derived from the percussive strokes which is then used in onset detection in the second step.

It is observed that the percussive strokes in Carnatic music can be modeled by an AM-FM signal based on amplitude and frequency variations in the vicinity of an onset [15]. An amplitude and frequency modulated signal ($x(t)$) is given by,

$$x(t) = m_1(t)cos(\omega_c t + k_f \int m_2(t)dt) \tag{5}$$

where, k_f is the frequency modulation factor, ω_c is the carrier frequency and, $m_1(t)$ and $m_2(t)$ are the message signals. The changes in the frequency are emphasized in the amplitude of the waveform by finding the differences of the time-limited discrete version of the signal, $x[n]$. The envelope function $e[n]$ is the amplitude part of $x'[n]$. The real-valued envelope signal can be represented by the corresponding analytic signal defined as:

$$e_a[n] = e[n] + ie_H[n] \tag{6}$$

$e_H[n]$ is the Hilbert transform of the envelope function. The magnitude of $e_a[n]$ is the detection function for the onsets. The high-energy positions of the envelope signal ($e[n]$) corresponds to the onset locations. However, these positions have a large dynamic range and the signal has a limited temporal resolution. It has been shown in [20] that minimum-phase group delay (GD) based smoothing

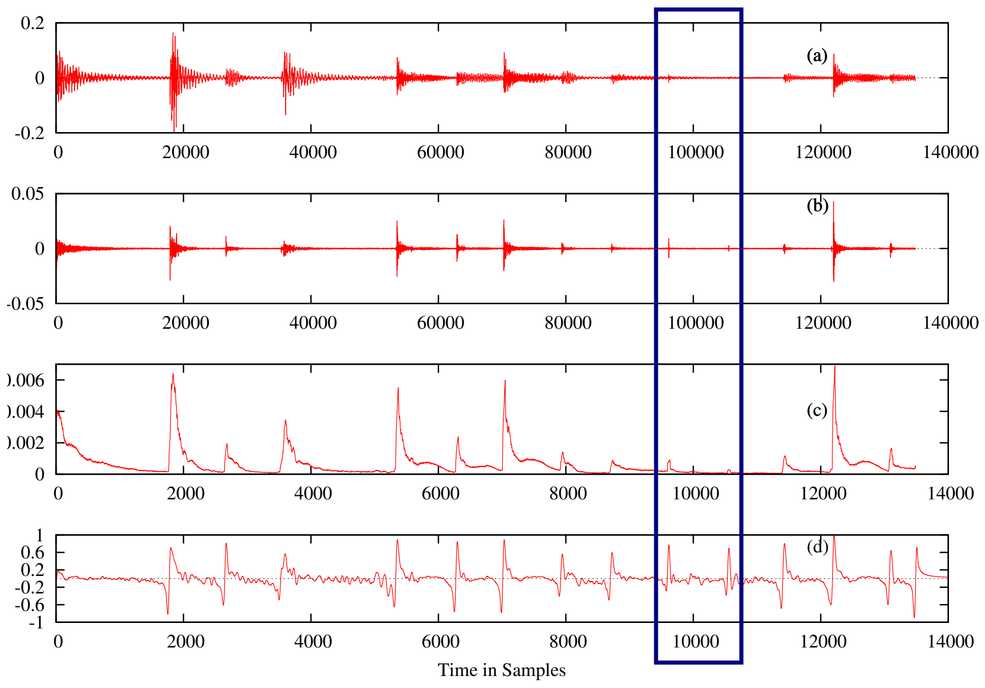


Figure 4: Solo onset detection algorithm. (a) Percussion signal (b) Derivative of (a). c) Envelope estimated on (b) using Hilbert transform. (d) Minimum phase group delay computed on (c).

leads to a better resolution for any positive signal that is characterized by peaks and valleys. The envelope is a non-minimum phase signal and it needs to be converted to a minimum phase equivalent to apply this processing.

It is possible to derive such an equivalent representation with a root cepstral representation. The causal portion of the inverse Fourier transform of the magnitude spectrum raised to a power of α is always minimum phase [18].

$$e'[k] = \{s[k] \mid_{k>0},\ s[k] = IFT((e(n) + e[-n])^{\alpha})\} \quad (7)$$

Note that $e'[k]$ is in root cepstral domain and k is the quefrency index. This minimum-phase equivalent envelope is then subjected to group delay processing.

The group delay is defined as negative frequency derivative of the unwrapped phase function. It can be computed directly from the cepstral domain input signal $e'[k]$ as:

$$\tau(\omega) = \frac{X_R(e^{j\omega})Y_R(e^{j\omega}) + X_I(e^{j\omega})Y_I(e^{j\omega})}{|X(e^{j\omega})|^2} \quad (8)$$

where, $X(e^{j\omega})$ and $Y(e^{j\omega})$ are the discrete Fourier transforms of $e'[k]$ and $ne'[k]$ respectively. Also, R and I denote the real and imaginary parts respectively. The high resolution property of the group delay domain emphasizes the onset locations. Onsets are reported as instants of significant rise, above a threshold.

Figure 4 illustrates the different steps in the algorithm using a mridangam excerpt taken from a *tani āvarthanam* segment. It is interesting to note that in the final step, the group delay function emphasizes all the strokes approximately to an equal amplitude, and even those onsets in which there is no noticeable change in amplitude are also obtained as peaks (highlighted area in Figure 4).

4. PERFORMANCE EVALUATION

The proposed percussive onset detection approach is developed specifically for rhythm analysis in Carnatic music composition items. However, it is instructive to compare the performance with other separation and onset detection algorithms. Also, it is important to note that the proposed approach could be applied to any music tradition with enough training musical excerpts to extract the onset locations from the polyphonic mixture. The dataset for these tasks is described in Section 2. The vocal-violin channel (*ālāpana*) and the percussion channel (*tani āvarthanam*) are mixed at 0 dB SNR. The STFT with a window length of 1024 samples and hop size of 512 samples is used as the feature for training a DRNN with 3 hidden layers (1000 units/layer) and temporal connection at the 2nd layer. This architecture shows a very good separation for the singing voice separation task [12]. The dataset consists of segments with varying tempo, loudness and number of sources at a given time. The challenge lies in detecting the onsets in the presence of the interference caused by other sources and the background voices.

4.1 Evaluation Metrics

Since the estimation of percussive onsets also depends on the quality of separation, it is necessary to evaluate the separated track. We measure this using three quantitative measures based on BSS-EVAL 3.0 metrics [23]: Source to Artifacts Ratio (SAR), Source to Interference Ratio (SIR) and Source to Distortion Ratio (SDR). The artifacts introduced in the separated track is measured by SAR. The suppression achieved for the interfering sources (vocal and violin) is represented in terms of SIR which is an indicator of the timbre differences between the vocal-violin mixture and percussive source. SDR gives the overall separation quality. The length-weighted means of these measures are used for representing the overall performance in terms of global measures (GSAR, GSIR and GSDR).

The conventional evaluation metric for the onset detection is F-measure, which is the harmonic mean of precision and recall. An onset is treated as correct (*True Positive*) if it is reported within a ±50ms threshold of the ground truth [6] as strokes inside this interval are usually unresolvable. Additionally, this margin accounts for the possible errors in the manual annotation. The F-measure is computed from sensitivity and precision. Since it is impossible to differentiate between simple and composite [3] strokes for mridangam, the closely spaced onsets (within 30 *ms*) are not merged together unlike in [5].

4.2 Comparison Methods

The performance of the separation stage is compared with a widely used Harmonic/Percussive Separation (HPS) algorithm [8] for musical mixtures. It is a signal processing-based algorithm in which median filtering is employed on the spectral features for separation. Other supervised percussive separation models were specific to the music traditions. We have not considered the Non negative Matrix Factorization (NMF)-based approaches since the separation performance was worse on Carnatic music, hinting the inability of a constant dictionary to capture the variability across the percussive sessions and instruments.

The onset detection performance is compared with the state-of-the-art CNN-based onset detection approach [19]. In this approach, a convolutional network is trained as a binary classifier to predict whether the given set of frames has an onset or not. It is trained using percussive and non percussive solo performances. We evaluate the performance of this algorithm on the separated percussive track and, on the mixture . The onset threshold amplitude is optimized with respect to the mixture and percussive solo channel for evaluating the performance on the separated and mixture tracks respectively for both of these algorithms.

5. RESULTS AND DISCUSSION

5.1 Percussive Separation

The results of percussive separation are compared with that of the HPS algorithm in Table 2. The large variability of the spectral structure with respect to the *tonic*, strokes and the percussive instruments (different types of mridangam as well) cause the HPS model to perform poorly with respect to the proposed approach. The DRNN separation benefits from the training whereas the presence of the melodic component with rich harmonic content adds to the interference in HPS method. This results in a poor separation of melodic mixture and percussive voice in HPS approach as indicated by an overall difference of 5.51 dB SDR with respect to DRNN approach. Although DRNN is not trained on the concerts KK and MD, separation measures are quite similar to other concerts. This is an indicator of the generalization capability of the network since each concert is of a unique *tonic* (base) frequency, and is recorded under a different environment. Separated sound examples are available online [4] .

[3] both left and right strokes co-occurring in the mridangam

	DRNN			HPS		
Concert	**GSDR**	**GSIR**	**GSAR**	**GSDR**	**GSIR**	**GSAR**
SS	7.00	13.70	8.61	3.39	6.73	7.93
ND	7.54	17.30	8.98	0.46	3.05	7.67
KK	7.37	13.93	8.93	0.66	2.04	10.09
MH	6.40	15.64	7.63	0.82	3.31	7.79
KR	7.37	13.93	8.93	1.32	2.43	9.09
MD	6.40	15.64	7.63	2.40	8.06	4.78
Average	**7.01**	15.02	8.45	**1.50**	4.27	7.89

Table 2: Percussive separation performance in terms of BSS evaluation metrics for the proposed approach and HPS algorithm

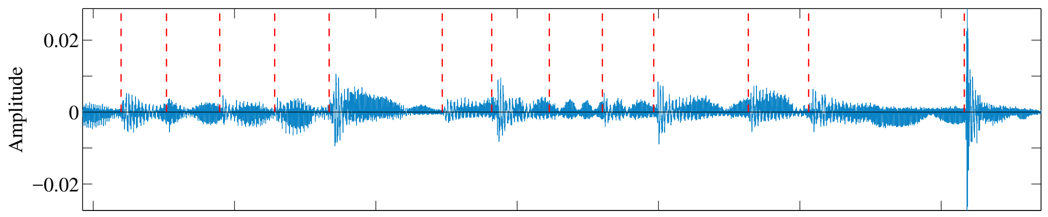
5.2 Onset Detection

Concert	Proposed	Direct	Solo	CNN	CNN Sep.
SS	0.747	0.448	0.864	0.685	0.656
ND	0.791	0.650	0.924	0.711	0.740
KK	0.891	0.748	0.972	0.587	0.636
MH	0.874	0.687	0.808	0.813	0.567
KR	0.891	0.748	0.972	0.859	0.848
MD	0.874	0.687	0.808	0.930	0.919
Average	**0.845**	**0.661**	0.891	0.764	**0.727**

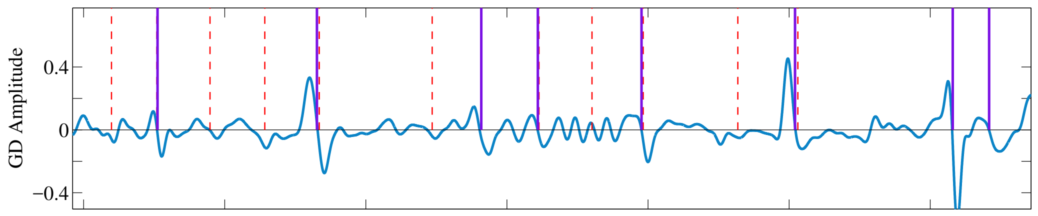
Table 3: Comparison of F-measures for the proposed approach, direct onset detection on the mixture, solo percussion channel, CNN on the mixture and on the separated percussive channel.

The accuracy of onset detection is evaluated using F-measure in Table 3. The performance varies with the dataset and the results with the maximum average F-measure is reported. The degradation in performance with respect to the solo source is only about 4.6%, while the improvement in performance compared to the direct onset detection on the composite source is 18.4%. The separation step plays a crucial role in onset detection of the composition items as the performance has improved for *all* the datasets upon separation. It should be noted that the algorithm performs really well for solo percussive source. This is reason for making comparisons with solo performances. For SS data (Table 1) with fast tempo (owing to multiple percussive voices) and significant loudness variation (Example online [4]), the direct onset method causes a large number of false positives resulting in lower precision whereas the proposed approach results in a reduced number of false positives. Figure 5 shows an example of a

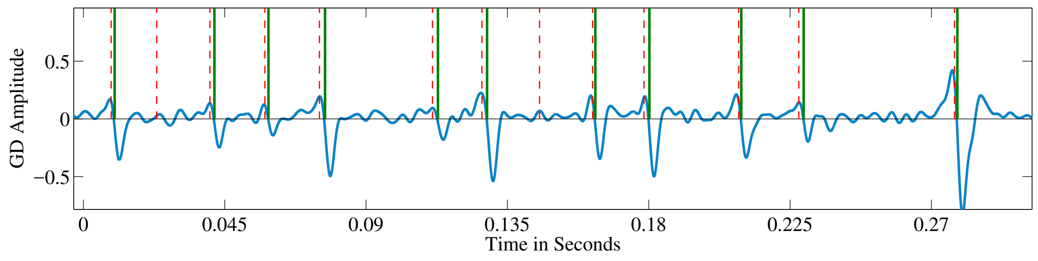
[4] `https://sites.google.com/site/percussiononsetdetection`



(a) A segment of composition item with the ground truth onsets



(b) Group delay representation for the mixture signal with the detected onsets



(c) Group delay representation for the separated signal with the detected onsets

Figure 5: An excerpt from SS dataset illustrating the performance of the proposed approach with respect to the direct onset detection method. Red dotted lines represent the ground truth onsets, violet (b) and green (c) lines represent the onsets detected on the mixture signal and the separated percussive signal respectively.

composition item taken from the SS dataset. It compares the performance of the proposed approach with that of the onset detection algorithm applied directly on the mixture. By adjusting the threshold of onset, the number of false positives can be reduced. However, it leads to false negatives as shown in Figure 5(b). The proposed approach is able to detect almost all of the actual onset locations (5(c)).

The proposed approach is then compared with the CNN algorithm. The optimum threshold of the solo algorithm for the Carnatic dataset [15] is used to evaluate the performance. The proposed method performs better than the CNN algorithm applied on the mixture (Table 3). This is because the CNN method is primarily for solo onset detection. The performance of the baseline on the separated channel is also compared with the group delay-based method. The threshold is optimized with respect to the performance of the baseline algorithm on the mixture track. The average F-measure of the proposed approach is 11.8% better than that of the CNN-based algorithm. This is because CNN-based onset detection requires different thresholds for different concert segments. This suggests that the GD based approach generalizes better in the separated voice track and is able to tolerate the inter-segment variability. A consistently better F-measure is obtained by the GD based method across all recordings. This separation-driven algorithm can be extended to any music tradition with sharp percussive onsets and having enough number of polyphonic musical ensembles for the training. These onset locations can be used to extract the strokes of percussion instruments and perform *tāla* analysis.

6. CONCLUSION AND FUTURE WORK

A separation-driven approach for percussive onset detection in monaural music mixture is presented in this paper with a focus on Carnatic music. Owing to its tonic dependency and improvisational nature, conventional dictionary-based learning methods perform poorly on percussion separation in Carnatic music ensembles. Vocal and violin segments from the *ālāpana* and mridangam phrases from the *tani āvarthanam* of concert recordings are used to train a DRNN for the percussive separation stage. The separated percussive source is then subjected to onset detection. The performance of the proposed approach is comparable to that of the onset detection applied on the solo percussion channel and achieves 18.4% absolute improvement over its direct application to the mixture. It compares favourably with the separation and onset detection baselines on the solo and separated channels. The onset locations can be used for analyzing the percussive strokes. Using repeating percussion patterns, the *tāla* cycle can be ascertained. This opens up a plethora of future tasks in Carnatic MIR. Moreover, the proposed approach is generalizable to other music traditions which include percussive instruments.

7. ACKNOWLEDGEMENTS

This research is partly funded by the European Research Council under the European Unions Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583). Authors would like to thank the members of Speech and Music Technology Lab for their valuable suggestions.

8. REFERENCES

[1] Juan P Bello, Chris Duxbury, Mike Davies, and Mark Sandler. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letters*, 11(6):553–556, 2004.

[2] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[3] Juan Pablo Bello and Mark Sandler. Phase-based note onset detection for music signals. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages V–441. IEEE, 2003.

[4] Emmanouil Benetos and Simon Dixon. Polyphonic music transcription using note onset and offset detection. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 37–40. IEEE, 2011.

[5] Sebastian Böck, Florian Krebs, and Markus Schedl. Evaluating the online capabilities of onset detection methods. In *Proc. of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, pages 49–54, 2012.

[6] Sebastian Böck and Gerhard Widmer. Local group delay based vibrato and tremolo suppression for onset detection. In *Proc. of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, pages 361–366, Curitiba, Brazil, November 2013.

[7] Simon Dixon. Onset detection revisited. In *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx)*, pages 133–137, 2006.

[8] Derry Fitzgerald. Harmonic/percussive separation using median filtering. In *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx)*, pages 15–19, 2010.

[9] Derry Fitzgerald, Antoine Liukus, Zafar Rafii, Bryan Pardo, and Laurent Daudet. Harmonic/percussive separation using kernel additive modelling. In *Proc. of the 25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communications Technologies (ISSC 2014/CIICT 2014)*, pages 35–40, 2014.

[10] Masataka Goto and Yoichi Muraoka. A sound source separation system for percussion instruments. *Transactions of the Institute of Electronics, Information and Communication Engineers (IEICE)*, 77:901–911, 1994.

[11] Masataka Goto and Yoichi Muraoka. Beat tracking based on multiple-agent architecture a real-time beat tracking system for audio signals. In *Proc. of 2nd International Conference on Multiagent Systems*, pages 103–110, 1996.

[12] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. *In Proc. of International Society for Music Information Retrieval (ISMIR)*, pages 477–482, 2014.

[13] M Humble. The development of rhythmic organization in indian classical music. *MA dissertation, School of Oriental and African Studies, University of London.*, pages 27–35, 2002.

[14] Anssi Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 3089–3092. IEEE, 1999.

[15] Manoj Kumar, Jilt Sebastian, and Hema A Murthy. Musical onset detection on carnatic percussion instruments. In *Proc. of 21st National Conference on Communications (NCC)*, pages 1–6. IEEE, 2015.

[16] Jom Kuriakose, J Chaitanya Kumar, Padi Sarala, Hema A Murthy, and Umayalpuram K Sivaraman. Akshara transcription of mrudangam strokes in carnatic music. In *Proc. of the 21st National Conference on Communications (NCC)*, pages 1–6. IEEE, 2015.

[17] Erik Marchi, Giacomo Ferroni, Florian Eyben, Stefano Squartini, and Bjorn Schuller. Audio onset detection: A wavelet packet based approach with recurrent neural networks. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, pages 3585–3591, July 2014.

[18] T Nagarajan, V K Prasad, and Hema A Murthy. The minimum phase signal derived from the magnitude spectrum and its applications to speech segmentation. In *Speech Communications*, pages 95–101, July 2001.

[19] Jan Schlüter and Sebastian Böck. Improved Musical Onset Detection with Convolutional Neural Networks. In *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 6979–6983, Florence, Italy, May 2014.

[20] Jilt Sebastian, Manoj Kumar, and Hema A Murthy. An analysis of the high resolution property of group delay function with applications to audio signal processing. *Speech Communications*, pages 42–53, 2016.

[21] Jilt Sebastian and Hema A Murthy. Group delay based music source separation using deep recurrent neural networks. In *Proc. of International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5. IEEE, 2016.

[22] Mi Tian, Ajay Srinivasamurthy, Mark Sandler, and Xavier Serra. A study of instrument-wise onset detection in beijing opera percussion ensembles. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2159–2163, 2014.

[23] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1462–1469, 2006.

SONG2GUITAR: A DIFFICULTY-AWARE ARRANGEMENT SYSTEM FOR GENERATING GUITAR SOLO COVERS FROM POLYPHONIC AUDIO OF POPULAR MUSIC

Shunya Ariga
The University of Tokyo
`ariga@iis-lab.org`

Satoru Fukayama
AIST
`s.fukayama@aist.go.jp`

Masataka Goto
AIST
`m.goto@aist.go.jp`

ABSTRACT

This paper describes Song2Guitar which automatically generates difficulty-aware guitar solo cover of popular music from its acoustic signals. Previous research has utilized hidden Markov models (HMMs) to generate playable guitar piece from music scores. Our Song2Guitar extends the framework by leveraging MIR technologies so that it can handle beats, chords and melodies extracted from polyphonic audio. Furthermore, since it is important to generate a guitar piece to meet the skill of a player, Song2Guitar generates guitar solo covers in consideration of playing difficulty. We conducted a data-driven investigation to find what factor makes a guitar piece difficult to play, and restricted Song2Guitar to use certain hand forms adaptively so that the player can play the piece without experiencing too much difficulty. The user interface of Song2Guitar is also implemented and is used to conduct user tests. The results indicated that Song2Guitar succeeded in generating guitar solo covers from polyphonic audio with various playing difficulties.

1. INTRODUCTION

A guitar solo cover version of an original song adds new pleasure to the music experience of the song. Various musical elements such as beats, melodies, and harmonies in an original song are represented in a uniform but expressive timbre of a guitar. However, a guitar solo cover of one's favorite song is not always available, and creating guitar arrangements requires advanced skills and knowledge and takes a lot of time. If such a guitar solo cover of any song can be generated from music audio signals, music listeners can enjoy their favorite songs in a different way, and guitarists who do not have skills for playing by ear can also enjoy performing any songs on their guitars.

The goal of this research is to develop a system that can automatically generate a guitar solo cover version from audio signals. By leveraging Music Information Retrieval (MIR) technologies, we propose a guitar arrangement system, *Song2Guitar*, that generates guitar solo covers from polyphonic audio signals of popular music, which contain sounds of various instruments. We also aim at creating difficult-aware guitar arrangements — i.e., generating guitar tablatures having different levels of playing difficulty for guitarists. There are three issues that should be considered:

(1) Generate from polyphonic audio of popular music

(2) Difficulty-aware arrangement

(3) Interface to perform the arrangement result

An overview of our solutions to address these issues is shown in Fig. 1. As for issue (1), even if we use the state-of-the-art MIR technologies, we cannot obtain completely-transcribed musical scores from such complex audio signals. We therefore directly extract important musical elements, such as melody lines represented as F0 (fundamental frequency) contour, beats, and chords, from polyphonic audio. We then reflect the extracted elements in generating guitar solo covers by using a novel extension of a hidden Markov model. As for issue (2), we conducted a data-driven survey to find what factors make a guitar tablature difficult to play. Based on the survey, Song2Guitar controls the movement of an index finger and the number of fingers to press the strings. Finally, as for issue (3), we designed and implemented an interface that enables a guitarist to change the degree of difficulty to perform the result. In this paper, we will also discuss a desirable interface for generating various arrangement results and providing training materials for guitarist. The design of the interface and the results generated by our system are available on the web [1].

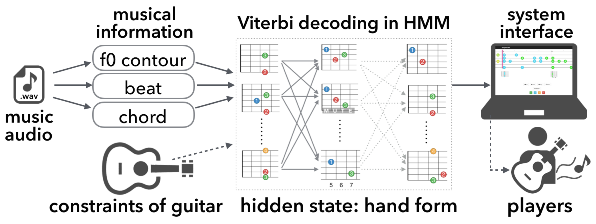


Figure 1: Overview of the Song2Guitar system.

[1] https://youtu.be/fN4-ibh7ZDI

2. RELATED WORK

2.1 Creative MIR

Our research is addressed in a Creative MIR approach. MIR researchers have recently explored creativity-oriented music technologies by applying technologies developed in the MIR community. This emerging field is named *Creative MIR* [11] where music analysis and transformation technologies are used in various creative applications. For example, AutoMashUpper [4] is an interactive system that creates music mashups by automatically selecting and mixing chosen songs. They achieved automatic mashup by estimating *mashability*, which is calculated by using MIR technologies to estimate various musical elements such as beats, downbeats, and chromagram. Song2Quartet [18] generates a cover song in the style of string quartet by combining probabilistic models estimated from a corpus of symbolic classical music with the target audio file of a song.

2.2 Generating playable guitar Solo

In order to generate a playable guitar covers, Hori *et al.* [7–9] used a hidden Markov model (HMM) to generate guitar arrangements from a symbolic musical score while considering natural fingerings. Audio signals, however, were not used as the input. By taking audio signals of an individual separated guitar part as the input, Yazawa *et al.* [25] developed an automatic transcription system specialized for a guitar performance and generated a guitar tablature by using multi-pitch analysis and playability constraints. Yazawa *et al.* [24] then extended their previous work to transcribe a guitar tablature while considering acoustical reproducibility and fingering easiness. Even though guitarist's proficiency was considered, creating guitar arrangements from polyphonic music including multiple instruments was not tackled so far.

Research of automatic fingering decision can also be regarded as related work of ours. This is because fingering decision is a sub-problem to generate playable guitar solo, and the existence of a fingering for a song is a necessary condition for the song to be playable. Radicioni discussed in his thesis how to computationally model the fingering in music performance [19]. The fingering is often determined by searching the fingering sequence as an optimal path search problem [20, 21].

Fingerings are represented in a tablature score or tabs, and they are often utilized to analyze and generate playable scores. A method to analyze and search valuable information in the tablature database has been proposed [14]. AutoGuitarTab [15–17] generates guitar music according to different styles of various guitarists by training individual probabilistic model using a tablature database. Genetic algorithms have been used to search the fingering sequence efficiently to generate an guitar solo arrangement [22].

Tablature transcription from music audio are also the related work. MIR technologies such as multi-pitch analysis and chord recognition have been used to capture notes and chords in the audio signal. Given those music elements, dynamic programming or Viterbi decoding with HMM has been leveraged to output reasonable fingerings [1, 10, 12].

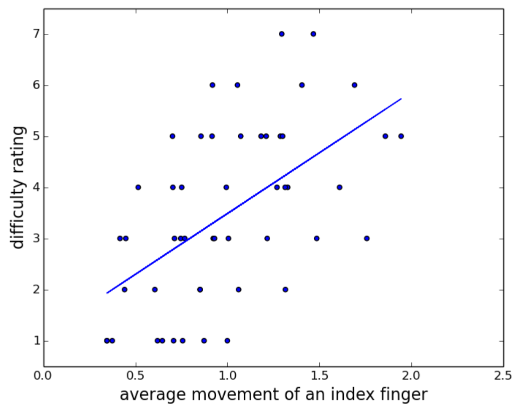


Figure 2: Average movement of an index finger and difficulty rating of 50 tablatures. The correlation coefficient was 0.55. The line indicates the linear regression result and the R-squared value was 0.30.

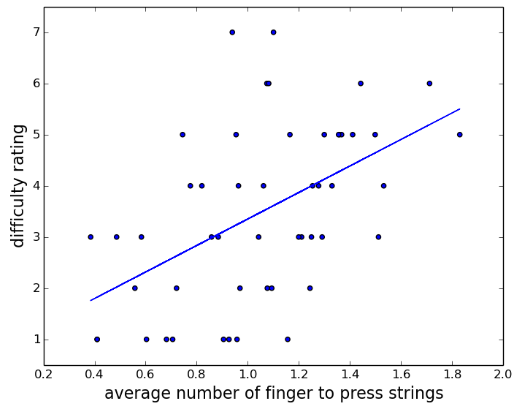


Figure 3: Average number of finger pressuring strings and difficulty rating of 50 tablatures. The correlation coefficient was 0.51. The line indicates the linear regression result and the R-squared value was 0.26.

3. PLAYING DIFFICULTY OF A GUITAR SOLO

3.1 Analysis of guitar tablatures

We first investigated what are the factors that affect the playing difficulty. We collected tablatures from a web site distributing classical guitar music [2]. These tabs were written in a plain text format and did not have uniformity in data structure. We therefore implemented a parser to retrieve structured tab information. Since Song2Guitar assumes only the standard tuning (E-A-D-G-B-E), we excluded tabs that were instructed to play in another tuning. Furthermore, we also excluded scores for guitar duo as we focus on guitar solo covers.

[2] http://www.classtab.org

For each tablature, we considered and calculated two factors: the average movements of the index finger of a hand to hold the guitar, and the average number of fingers to press the strings. The position of index finger is determined as follows: if the fingering contains a barre, we use the fret position of the barre, otherwise the position of the index finger is set to be the minimum fret number among the frets being pressed. We hypothesized that these two factors affect the playing difficulty of a guitar solo.

3.2 Subjective test to evaluate the playing difficulty

We verified our hypothesis by asking proficient guitarists to rate the difficulty of the tabs. Five independent raters subjectively evaluated the difficulty with a 7-point Likert scale (1: easiest – 7: most difficult). The raters were instructed to consider only the complexity of the fingerings of the left hand. As they respectively rated randomly-selected 10 tabs, we consequently obtained 50 ratings. Fig. 2 and Fig. 3 show the plots of our hypothesized features (average movement of an index finger, average number of fingers to press strings) and the result of difficulty ratings. The correlation coefficients calculated with these two features and the ratings were 0.55 and 0.51, respectively. We also conducted a linear regression on data. The regression results are also shown in Fig. 2 and Fig. 3. R-squared values for these two regressions were 0.30 and 0.26, respectively. These results indicated that the tabs were evaluated to be more difficult when the values of both features get larger.

4. CREATING DIFFICULTY-AWARE GUITAR ARRANGEMENTS FROM MUSIC SIGNALS

4.1 Our problem setting

In solving the problem of generating guitar solo covers from audio signals of popular music, we want to maintain and reproduce major characteristics of an original song in a generated guitar solo cover. The followings are the major characteristics that most songs in popular music have in common.

- It contains a clear melody line that is performed by a vocal part.
- It contains a bass line corresponding to a chord sequence.
- It gives a rhythmic groove emerged from sounds of rhythmic instruments.

Although previous work of generating guitar arrangements from symbolic musical scores [7] formulates the problem by using HMM, it have not tested with audio input. To generate from polyphonic music audios of popular music, we formulate the problem by a novel extension of HMM. We also propose how we can generate various results with different levels of playing difficulty for guitarists.

4.2 Guitar Arrangement by using HMM

We start from reviewing how HMM can be applied to the fingering decision problem. Suppose we have a collection of guitar music scores, and we want to model this collection statistically. This means that we need to obtain a function that returns high probability if the music seems to be included in the guitar music collection, and low probability when the music is obviously not a guitar music. Designing a generative model is one method to achieve this.

The generative process of a guitar music is apparently the process of performing a guitar instrument. When the guitar is played, one hand holds the neck and its fingers press strings on the frets. Fingers of the other hand pluck the strings, and eventually a sound is generated. We can see that the output sound is determined when the states of both hands are determined.

In terms of the hand to press the strings, it is less likely to observe a drastic change of the hand form in a very short duration because of physical constraints of the human body. It is also unlikely to observe a long distance move of position of a hand to hold the neck of a guitar. Since these two aspects are relationships between the current and the previous state of holding the neck, we can model them by the first-order Markov chain. Let X_t be the fingering at time t. We can define a probability for observing fingering X_{t+1} as $P\left(X_{t+1}|X_t\right)$. X_t contains four components each of which corresponds the state of each finger of the left hand. Each component has two values: one indicates the string index of a guitar to put pressure on, and the other indicates the fret number to put the finger on. Fret number 0 indicates that the finger does not touch any string.

The output sound is audible when strings are plucked. The sounding notes are biased by the fingering. Let Y_t be the set of notes played at time t, such as set consisting of C3, E4 and G4. Y_t follows a probability distribution $P\left(Y_t|X_t\right)$ which models the playing notes biased from the fingering.

A guitar performance can be realized as a time sequence of both the fingering ($X_1^T = X_1 \cdots X_T$) and the plucking of strings at each fingering ($Y_1^T = Y_1 \cdots Y_T$). Note that T indicates the length of a sequence, not indicating transposition. The probability of generating notes from the given fingering is calculated by the product of these probabilities as:

$$P\left(Y_1^T|X_1^T\right) = \prod_{t=1}^{T} P\left(Y_t|X_t\right) P\left(X_t|X_{t-1}\right). \qquad (1)$$

Since the fingering cannot be observed from the guitar music afterwards, X_1^T is hidden and therefore this probabilistic model is called as *hidden* Markov model. $P\left(Y_t|X_t\right)$ is called as emission probability, and $P\left(X_t|X_{t-1}\right)$ is called as transition probability. By using Viterbi decoding, we can efficiently estimate the most likely fingerings which maximize the likelihood in terms of X_1^T [23].

Now we can extend the generative model discussed above to let the model generate music that is not necessary to be guitar music, but *could be arranged* into guitar music. In particular, the emission probability is revised so that the model can output notes which are octave higher or lower than the notes of the actually played pitches by plucking the string. As Hori *et al.* formulated in their works [7–9], the emission probability is set to allow the number of notes more than the guitar can perform simultaneously.

By executing the Viterbi algorithm with this extension, we can obtain a sequence of fingering from not only guitar music but also from any music which is not originally composed for a guitar. Since the existence of proper fingering is the necessary condition for a guitar arrangement, we can generate a guitar solo cover by the above extension of fingering decision formulation.

4.3 Creative MIR Approach to Guitar Arrangement

4.3.1 Leveraging MIR technologies

One of the novelties of this research is the further extension of the generative model so that it can generate guitar solo covers from polyphonic music audios by leveraging MIR technologies. Since the melody, beats, and chords are main elements that can be reflected in a guitar solo arrangement, it is not necessary to try to obtain all notes by using multi-pitch analysis. We therefore use methods that can estimate the melody (F0 contour), beats, and chords in polyphonic audio including drums. We show that these methods developed in the MIR community largely contribute to generating a guitar solo cover.

The melody estimation here assumes that the melody is sung by a singer. We first extract a singing voice track by using an existing singing voice extraction method. We then applied a melody estimation method proposed by Goto [5] to obtain the F0 contour. We also smooth the F0 contour by using an FIR low-pass filter with 5 Hz cutoff frequency in order to remove the vibrato. To discretize the F0 contour into musical notes, we used beat estimation results to approximately obtain what musical note is played as the melody line at every 16th note.

Chord estimation provides chord labels (chord names). Since the label contains "on-chords" such as "C/E" or "C on E", we literally use the bass note described in the chord label. We used a chord estimation method developed by Korezeniowski *et al.* [13], which is available in Madmom or an audio signal processing library written in python [2].

Finally, the beat estimation plays an important role in generating a guitar solo cover. The beat estimation results give us a set of segments corresponding to quarter notes. By dividing every quarter note into four parts, we obtain a finer set of segments with the resolution of 16th notes. These 16th-note segments can be used to quantize the F0 contour of the melody line as explained above, and also quantize chord estimation results. In other words, all note lengths extracted from the audio are quantized into integer multiples of the 16th-note duration. We used a beat estimation method proposed by Böck *et al.* [3] which is also available in Madmom [2].

4.3.2 Emission probability

Based on the results of these estimation methods, we set the emission probability as follows so that the HMM can handle music audio to generate a guitar solo cover:

$$\begin{aligned} P\left(Y_t|X_t\right) &\propto P_{\text{chord}}\left(C_t|X_t\right) + P_{\text{melody}}\left(M_t|X_t\right) \\ &+ P_{\text{bass}}\left(B_t|X_t\right) \end{aligned} \tag{2}$$

The subscript t denotes the index of the onset. Since the onsets are not apparent in audio signals, we regard the timing of a sudden increase of power in singing voice and the timing of every chord change as onset timings. The onset timings are discretized by using the beat estimation result. Y denotes the audio segment with 16th-note duration. C, M, and B are the chord label, melody pitch, and bass pitch, respectively. X denotes the fingering to press the fret, which is a set of the pressing position of each finger including open strings. Open strings are represented as pressing the imaginary 0^{th} position of the fret.

Probability P_{chord} is set based on how the current fingering achieves the chord observed at the time. For example, when the fingering is given to play "C, E, G", the probability for observing "C maj7" would be high, but the probability for observing "F# maj" would be low. This can be measured by the number of elements in intersection between the set of notes derived from the fingering and the chord label. In this example, the set of notes for "C maj7" is "C, E, G, B", and the set of notes for "F# maj" is "F#, A#, C#". The probability for observing "C maj7" is higher since the intersection has "C, E, G" (3 elements) whereas "F# maj" has no elements as intersection. We implemented this as follows:

$$P_{\text{chord}}\left(C_t|X_t\right) \propto \exp\left(-\alpha \cdot \#\left(\mathbf{N}\left(C_t\right) \cap \mathbf{N}\left(X_t\right)\right)\right) \tag{3}$$

where $\mathbf{N}\left(\cdot\right)$ denotes the set of consisting notes of chord label or guitar fingering. We adjusted the parameter to be $\alpha = 3.0$ in our experiments.

Probability P_{melody} is designed by considering how the highest note of the playing notes with the fingering is relevant to the melody pitch observed in the acoustic signal. Let $M\left(X_t\right)$ be the highest note could be played from the current fingering. The probability can be designed as:

$$P_{\text{melody}}\left(M_t|X_t\right) \propto \begin{cases} 1.0 & \left(M_t = M\left(X_t\right)\right) \\ \varepsilon_1 & \left(M_t = M\left(X_t\right) + 12n\left(n \neq 0\right)\right) \\ \varepsilon_2 & \text{otherwise} \end{cases} \tag{4}$$

where the parameters are set as $\varepsilon_1 = 0.3$ and $\varepsilon_2 = 10^{-5}$. These parameters were set heuristically by iteratively generating and subjectively evaluating the results.

Finally, probability P_{bass} is set similarly to P_{melody}. Let $B\left(X_t\right)$ be the lowest note that could be played from the current fingering X_t. The probability is designed as:

$$P_{\text{bass}}\left(B_t|X_t\right) \propto \begin{cases} 1.0 & \left(B_t = B\left(X_t\right)\right) \\ \varepsilon_1 & \left(B_t = B\left(X_t\right) + 12n\left(n \neq 0\right)\right) \\ \varepsilon_3 & \text{otherwise} \end{cases} \tag{5}$$

where the parameter ε_1 shared the same value as in P_{melody}, and ε_3 was set as $\varepsilon_3 = 0.0027$ in our experiment.

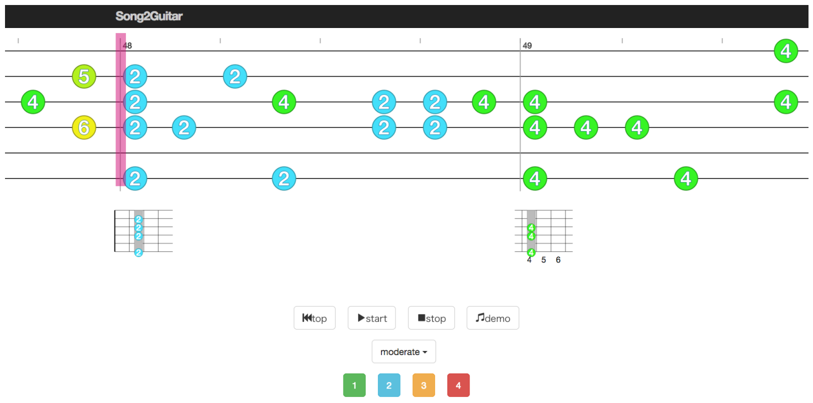


Figure 4: User interface of the Song2Guitar system.

4.3.3 Transition probability

For setting transition probability $P(X_{t+1}|X_t)$, we basically followed the formulation by Hori *et al.* [7]. We defined the transition probability given the time interval d_t between onsets as:

$$
\begin{aligned}
P(X_{t+1}|X_t) \propto \frac{1}{2d_t} \exp\left(-\lambda_m \frac{|I(X_{t+1}) - I(X_t)|}{d_t}\right) \\
\times \frac{1}{1 + I(X_{t+1})} \times \frac{1}{1 + W(X_{t+1})} \\
\times \frac{1}{1 + N(X_{t+1})}
\end{aligned} \tag{6}
$$

where $I(X)$ denotes the position of an index finger when holding the fret of a guitar with a fingering X. $W(X)$ denotes the length between the leftmost fret used and the rightmost fret used under the fingering X. $N(X)$ denotes the number of fingers used to achieve the fingering X.

4.4 Controlling the Degree of Difficulty

Based on the survey described in section 3, we determined the following two parameters to control the playing difficulty for generating a guitar solo cover: the average movement of the index finger of a hand to hold the guitar, which is denoted as a_{move}, and the average number of fingers to press the strings, which is denoted as a_{string}.

Song2Guitar supports three different levels of playing difficulty: EASY, NORMAL, and HARD. To create these levels by changing a_{move} and a_{string}, we adaptively restricted the use of fingering X according to the following constraints:

$$
\begin{aligned}
\text{EASY}: a_{\text{move}} \leq 2.0 \ \ \&\& \ \ a_{\text{string}} \leq 2.0 \\
\text{NORMAL}: a_{\text{move}} \leq 4.0 \ \ \&\& \ \ a_{\text{string}} \leq 3.0 \\
\text{HARD}: \text{use all available fingerings.}
\end{aligned} \tag{7}
$$

5. INTERFACE DESIGN OF SONG2GUITAR

Song2Guitar aims at not only generating a guitar solo cover automatically but also enabling a guitarist to easily practice and perform the generated result. Fig. 4 shows the main interface of Song2Guitar. The design of the interface and the results generated by our system are available on the web [3].

Because the tablature form is more intuitive than the music score, Song2Guitar visualizes the tablature score of the generated cover song. This tablature score scrolls automatically while playing since a guitarist uses both hands to perform the guitar and no hands are left to control the system.

We also implemented an interface to control the playing difficulty of the results. When we aim at creating playable arrangements for human guitarists, it is important to control how difficult the generated score is. Guitarists would be discouraged if the score is too difficult or too easy for them.

The tablature shown in the interface contains additional notations to make the practice and performance easier. Numbers in colored circles on the strings indicate the fret that the guitarist should press on the string. The indicator with a purple vertical line (in the left of Fig. 4) shows the timing to pluck the string.

The interface of Song2Guitar also supports non-proficient guitarists to find the position to press the indicated frets. Usually a guitarist needs to prepare the hand form to press the fret in advance of plucking the strings. Even though the tablature score is shown, a novice guitarist often gets stuck in keeping finding where to put their left hand to hold the neck of a guitar. This is because the tablature usually indicates only the fingerings on the frets, but does not indicate the position of the left hand to hold the neck of the guitar. Therefore we implemented to show small diagrams (shown below of the tablature in Fig. 4) representing how to place the fingers in a similar fashion to a guitar fingerboard. The diagram is shown when there is a position change in a hand to hold the frets.

The Song2Guitar interface also supports a demonstration mode which playbacks the generated tablature by using synthesized guitar sounds so that music listeners can simply enjoy the system output.

[3] https://youtu.be/fN4-ibh7ZDI

6. EVALUATION

To evaluate how the performing difficulty varied among the generated guitar solo covers, we conducted an experiment in a qualitative evaluation approach.

6.1 Experimental setting

We asked a guitarist who is proficient in playing the classical guitar to participate in the evaluation. The guitarist was male and 24 years old, and had an experience in playing the acoustic guitar (both steel and nylon strings) for around six years.

We used RWC-MDB-P-2001 No.7 from the RWC Music Database [6] to generate a guitar solo cover. We generated three different covers with different levels of difficulty (EASY, NORMAL, and HARD) by using Song2Guitar. To focus on evaluating varying difficulty, we manually corrected estimated beats and chords before generating them.

The guitarist was first asked to practice each score for 15 minutes. Since the duration of generated pieces were about five minutes long and it was too long to practice the entire song, we asked the guitarist to practice only the intro, the first verse, and the chorus section. After the practice, we asked the guitarist to play all designated sections of each cover. Finally, we conducted a short interview to obtain comments on Song2Guitar. The obtained comments were originally in Japanese, and they were translated into English as shown in this paper.

6.2 Evaluation results

We obtained a comment indicating that the participant enjoyed using the system:

I think this is a really great app.. I can play a song endlessly, and it was like some kind of a game.

We also found a comment to indicate that our system generated covers in three different levels of playing difficulty (EASY, NORMAL, and HARD):

Well, playing difficulties were appropriate, difference between NORMAL and HARD makes sense.

Although we intended to make three covers as getting gradually difficult, the participant commented that the playing difficulty of EASY and NORMAL were reversed:

EASY score was not easy, it was more difficult than NORMAL one, for me. The HARD score was like in the middle of NORMAL and EASY.

The participant reported why “EASY score was not easy” as follows:

I guess that it’s easier when it consists of chords (multiple notes) moderately than full of simple notes. Chords are the basic form, and I can figure out how to do fingering in my mind. When only two notes appear in the tab, of course, I can figure out the fingerings, however, it didn’t go well [...]

This comment indicated that smaller number of notes are not always easy to play. The fingering of chords provides a basic form, and a guitarist is more familiar with it than the other irregular fingerings for fewer notes.

The participant also pointed out the playing difficulty comes from the note value of the generated results.

The difficulty is, I think it’s easy if all notes were eighth note. Sixteenth note is difficult to figure out the timing.

He also indicated the issue in the interface design:

It’s hard to understand beats and timings of notes with the interface. I appreciate if every half beat were highlighted, somehow.

7. DISCUSSION

We confirmed that Song2Guitar was able to generate guitar solo covers from polyphonic audio of popular music by leveraging MIR technologies. We found that the HMM formulation to generate guitar solo combined with estimation of melody (F0), beats, and chords was effective even from music audio which multi-pitch analysis cannot be sufficiently applied to.

We also found that Song2Guitar was able to generate output with different playing difficulties. We introduced two parameters: the average movement of an index finger and the average number of fingers to press the strings, to control the playing difficulty. The evaluation results, however, suggested that there would be more factors that affect the playing difficulty. One possible factor for determining the difficulty is the familiarity of particular fingerings such as chords.

The interface of Song2Guitar enabled the player to practice and perform the generated result. The comments obtained in the experiment revealed that the rhythms of the generated covers were sometimes hard to recognize. The interface did not visualize the timing except for showing the indicator bar. Highlighting half beats would help the player recognize the rhythm much easier.

The future work of this research is to enable Song2Guitar to generate cover songs in real time considering the player’s proficiency. Conducting an objective evaluation is also included in future work. Since the generative model is designed as a probabilistic model, we can verify the fingering model by calculating cross-entropy.

8. CONCLUSION

We proposed Song2Guitar that generates guitar solo covers from polyphonic music signals of popular songs. The formulation using HMM was combined with MIR technologies so that it can generate covers considering the melody, bass and rhythm of the songs. Furthermore, Song2Guitar generated covers with different levels of playing difficulty. The interface was implemented and a guitarist succeeded in playing different guitar solo covers. In the future, cover song generation from music audio signals will be further improved by leveraging other MIR technologies.

9. ACKNOWLEDGEMENT

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

10. REFERENCES

[1] Ana M. Barbancho, Anssi Klapuri, Lorenzo J. Tardon, and Isabel Barbancho. Automatic transcription of guitar chords and fingering from audio. *IEEE/ACM TASLP*, 20(3):915–921, 2011.

[2] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proceedings of the 24th ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 10 2016.

[3] Sebastian Böck, Florian Krebs, and Gerhard Widmer. Joint beat and downbeat tracking with recurrent neural networks. In *Proc. ISMIR*, ISMIR '16, 2016.

[4] Matthew E. P. Davies, Philippe Hamel, Kazuyoshi Yoshii, and Masataka Goto. Automashupper: Automatic creation of multi-song music mashups. *IEEE/ACM TASLP*, 22(12):1726–1737, Dec 2014.

[5] Masataka Goto. A real-time music-scene-description system: predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4):311 – 329, 2004. Special Issue on the Recognition and Organization of Real-World Sound.

[6] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical and jazz music databases. In *Proc. ISMIR*, volume 2 of *ISMIR '02*, pages 287–288, 2002.

[7] Gen Hori, Hirokazu Kameoka, and Shigeki Sagayama. Input-output HMM applied to automatic arrangement for guitars. *Journal of Information Processing*, 21(2):264–271, 2013.

[8] Gen Hori and Shigeki Sagayama. HMM-based automatic arrangement for guitars with transposition and its implementation. In *Proc. ICMC/SMC*, 2014.

[9] Gen Hori and Shigeki Sagayama. Minimax viterbi algorithm for HMM-based guitar fingering decision. In *Proc. ISMIR*, 2016.

[10] Eric J. Humphrey and Juan P. Bello. From music audio to chord tablature: Teaching deep convolutional networkds to play guitar. In *Proc. IEEE ICASSP*, pages 7024–7028, 2014.

[11] Eric J. Humphrey, Douglas Turnbull, and Tom Collins. A brief review of Creative MIR. *Proc. ISMIR (Late Breaking/Demo Session)*, 2013.

[12] Christian Kehling, Jakob Abesser, Chirstian Dittmar, and Gerald Schuller. Automatic tablature transcription of electric guitar recordings by estimation of score- and instrument-related parameters. In *Proc. DAFx*, 2014.

[13] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: the deep chroma extractor. In *Proc. ISMIR*, ISMIR '16, pages 37–43, 2016.

[14] Robert Macrae and Simon Dixon. Guitar tab mining, analysis and ranking. In *Proc. ISMIR*, pages 453–458, 2011.

[15] Matt McVicar, Satoru Fukayama, and Masataka Goto. Autoleadguitar: Automatic generation of guitar solo phrases in the tablature space. In *Proc. IEEE ICSP*, pages 599–604, Oct 2014.

[16] Matt McVicar, Satoru Fukayama, and Masataka Goto. Autorhythmguitar: Computer-aided composition for rhythm guitar in the tab space. In *Proc. ICMC/SMC*, 2014.

[17] Matt McVicar, Satoru Fukayama, and Masataka Goto. Autoguitartab: Computer-aided composition of rhythm and lead guitar parts in the tablature space. *IEEE/ACM TASLP*, 23(7):1105–1117, 2015.

[18] Graham Percival, Satoru Fukayama, and Masataka Goto. Song2quartet: A system for generating string quartet cover songs from polyphonic audio of popular music. In *Proc. ISMIR*, pages 114–120. Citeseer, 2015.

[19] D.P. Radicioni. *Computational Modeling of Fingering in Music Performance*. PhD thesis, Università di Torino, Centro di Scienza Cognitiva, 2005.

[20] Aleksander Radisavljevic and Peter Driessen. Path difference learning for guitar fingering problem. In *Proc. ICMC*, volume 28, 2004.

[21] Samir I. Sayegh. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal*, 13(3):76–84, 1989.

[22] Daniel R. Tuohy and W. D. Potter. GA-based music arranging for guitar. In *Proc. IEEE Congress on Evolutional Computation*, pages 1065–1070, 2006.

[23] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theor.*, 13(2):260–269, September 2006.

[24] Kazuaki Yazawa, Katsutoshi Itoyama, and Hiroshi G. Okuno. Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency. In *Proc. IEEE ICASSP*, pages 3146–3150, 2014.

[25] Kazuaki Yazawa, Daichi Sakaue, Kohei Nagira, Katsutoshi Itoyama, and Hiroshi G. Okuno. Audio-based guitar tablature transcription using multipitch analysis and playability constraints. In *Proc. IEEE ICASSP*, pages 196–200, 2013.

THE SEILS DATASET: SYMBOLICALLY ENCODED SCORES IN MODERN-EARLY NOTATION FOR COMPUTATIONAL MUSICOLOGY

Emilia Parada-Cabaleiro^{1,2} **Anton Batliner**^{1,2} **Alice Baird**^{1,2} **Björn W. Schuller**^{1,2,3}
^{1} Chair of Complex and Intelligent Systems, University of Passau, Germany
^{2} Chair of Embedded Intelligence for Health Care and Wellbeing, Augsburg University, Germany
^{3} GLAM – Group on Language, Audio & Music, Imperial College London, U.K.
emilia.paradacabaleiro@informatik.uni-augsburg.de

ABSTRACT

The automatic analysis of notated Renaissance music is restricted by a shortfall in codified repertoire. Thousands of scores have been digitised by music libraries across the world, but the absence of symbolically codified information makes these inaccessible for computational evaluation. Optical Music Recognition (OMR) made great progress in addressing this issue, however, early notation is still an on-going challenge for OMR. To this end, we present the Symbolically Encoded *Il Lauro Secco* (SEILS) dataset, a new dataset of codified scores for use within computational musicology. We focus on a collection of *Italian madrigals* from the 16th century, a polyphonic secular a cappella composition characterised by strong musical-linguistic synergies. Thirty madrigals for five unaccompanied voices are presented in modern and early notation, considering a variety of digital formats: Lilypond, Music XML, MIDI, and Finale (a total of 150 symbolically codified scores). Given the musical and poetic value of the chosen repertoire, we aim to promote synergies between computational musicology and linguistics.

1. INTRODUCTION

Since scores are the only remaining source of Renaissance music, they are essential for replication and analysis of this repertoire. Through the analysis of an early score it is possible to identify musical similarities between composers [24], as well as correlations between poetry and music [32]. Due to this, libraries and museums spend great effort in the digitisation of early documents. This practice allows for easier dissemination of the repertoire and preserves it from the inevitable degradation.

Nevertheless, since this mass of scores have been scanned manually, no symbolically codifiable information is available, which makes them meaningless for computational procedures (e. g., automatic analysis). Furthermore, in digital libraries of symbolically encoded scores, transcriptions in modern notation of early musical repertoire are restricted, and early notation is almost completely ignored.

To resolve this issue, Optical Music Recognition (OMR) has been applied to early music [6, 10, 26, 28]. However, the degraded conditions of early documents (some times unreadable), and the linguistic inconsistencies between the different voices (common in vocal polyphonic music), make expert intervention essential, in some cases. Therefore, despite obtaining promising results, early notated music is still an open challenge for OMR [3]. With this in mind, we present the Symbolically Encoded *Il Lauro Secco* (SEILS) dataset[1]. The SEILS dataset is a corpus of scores encoded in a variety of digital formats (Lilypond [22][2], Music XML, MIDI and Finale[3]) and musical notation styles (*white mensural* notation [2] and modern Western notation) deliberately selected to maximise computational possibilities. Furthermore, considering the strong synergies between poetry and music typical of the chosen repertoire, the presented dataset aims to promote, from a musicological, linguistic and historic perspective, further understanding of the artistic manifestations of the 'Humanism Renaissance'.

In particular, the SEILS dataset is suitable for musical-linguistic pattern recognition, given the prominent relationship between music and lyrics that characterise the *Il Lauro Secco* anthology. Furthermore, since each *madrigal* (piece) of the considered repertoire is composed by a different composer, the SEILS dataset will also allow for automatic identification of composers' similarities. In addition, by presenting a codified version in *white mensural* notation (ground truth), OMR technology will be able to evaluate its performance. In section 2 we will evaluate previous studies related to the presented issue. In section 3 the considered repertoire will be described. An overview of the criteria for symbolic codification and an evaluation of the considered digital formats will be given in sections 4 and 5. Finally, the conclusions in Section 6.

[1] https://github.com/SEILSdataset/SEILSdataset
[2] http://lilypond.org/
[3] http://www.finalemusic.com

2. RELATED WORK

Even though scores are a great source of knowledge, codified symbolic information is missing for many. Some attempts have been made to improve this, mainly through OMR systems [4, 23, 31]. OMR has also been applied for processing early music by several initiatives: SIMSSA [10] [4], ARUSPIX [26] [5], and GAMERA [6]. OMR, when used with early notation, examines tablature and mensural notation [25], as well as primitive notation [15] and lyric recognition [3]. Nevertheless, the degraded conditions of the original source and the inconsistencies in the lyrics for vocal polyphonic music make human intervention crucial in many cases.

The score collections available online consider an increasing variety of digital formats. The most commonly found formats are Music XML and MIDI; however, other digital formats are becoming more popular: e. g., the **kern format [17] (available in the ELVIS database [1] [6], music21 [5] [7], and the kernscores database [29] [8]); Lilypond files [22] [9] (available in the Petrucci Music Library – IMSLP [10] and the MUTOPIA project database [11]); or files encoded through the professional music notation software Finale [12] (available in IMSLP). Nevertheless, despite rare exceptions like *Tasso in Music Project* [27] [13], *The Marenzio Online Digital Edition* – MODE [14], *Josquin Research Project* [15], or the *Liber Usualis* [16] encoded in MEI [16], early music in such a variety of formats is still limited.

3. THE SEILS DATASET REPERTOIRE

The musical repertoire considered for the presented dataset is the *Italian madrigal* of the 16th century, a secular polyphonic vocal composition in the Italian language, commonly for five to six unaccompanied voices. This kind of madrigal is characterised by meticulous musicalisation of poetic texts, a strategy known as *madrigalism* [14]. Towards the end of the 16th century, this composition technique was refined and flourished into a rich and virtuous music [7], characterised by its use of lyrics from great poets of the time [21]. The synergy between poetry and music, prominent within these madrigals, makes them an icon of the 'Humanism Renaissance' [13]. Given the relevance of this intellectual movement to Western Europe, the considered repertoire has great importance not only to Italian heritage [9], but also to musicological, linguistic, and historical research.

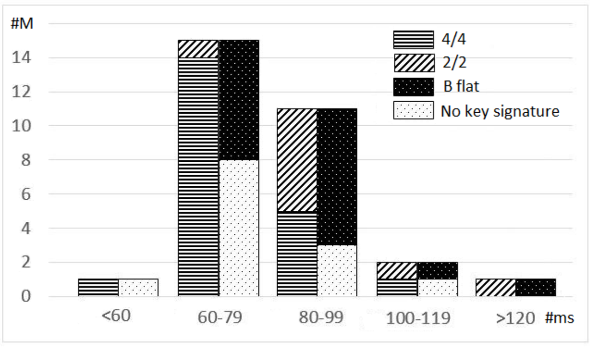


Figure 1: Distribution of the 30 madrigals utilised, considering: number of madrigals (#M), measure length (#ms), time signature (4/4 and 2/2), and key signature (B flat and no key signature).

3.1 *Il Lauro Secco* Anthology

The presented dataset is a codified version of the madrigal anthology *Il lauro Secco* (The dry laurel) [18], a collection of 31 Italian madrigals written by a variety of highly reputable composers from the end of the 16th century. For consistency, only 30 of these madrigals (for five a cappella voices, each written by a different composer), are available in the presented dataset. The 31st (and last) madrigal in the anthology has been excluded from the dataset, as it is starkly different from the others (for ten voices, and composed by one of the previously considered composers).

The presented anthology has been chosen for its high level of musical–linguistic consistency, i. e., composed with both music and lyrics expressively written for the anthology [20]. Such content is unique, as a standard for anthologies was to be created from pre-existing compositions, without musical or linguistic relationships. This homogeneity across the anthology allows for an interscore musical-linguistic analysis, which will enable for a deeper understanding of composer similarities via automatic recognition methods.

Both the music and lyrics of *Il lauro Secco* have been written by some of the most important Italian figures of this period. Several composers belong to the *Compagnia Romana*, also known as *Eccellenti Musici di Roma* (Excellent Musicians of Rome) [24], a congregation of composers famous for their proficiency. Furthermore, even though the authorship of the lyrics is not declared in the anthology, many have attributed this to the great Italian poet, Torquato Tasso [8, 12, 30].

3.2 The SEILS Dataset Statistic Evaluation

Considering the modern notated transcriptions, the presented madrigals display a mean average length of 79.5 measures (with a standard deviation of 15.7). Of the 30 madrigals, 21 are in 4/4 time signature and 9 in 2/2; 13 have a B flat in the key signature and 17 do not have alterations declared. In Figure 1, an overview of the distribution of madrigals is given, considering number of measures as well as key and time signature.

Although there is a high level of musical-linguistic consistency, the considered anthology is prominently characterised by its varying rhythms that differ between madri-

[4] https://simssa.ca/
[5] http://www.aruspix.net/
[6] https://database.elvisproject.ca/
[7] http://web.mit.edu/music21/
[8] http://kern.ccarh.org/
[9] http://lilypond.org/
[10] http://imslp.org/
[11] http://www.mutopiaproject.org/
[12] http://www.finalemusic.com
[13] http://www.tassomusic.org/
[14] https://d2q4nobwyhnvov.cloudfront.net/86940d50-206f-4db3-9b88-754dddb3486f/92KX7friyUw0WA/index.html
[15] http://josquin.stanford.edu/
[16] http://music-encoding.org/

	16th	*8th*	*4th.*	*2nd*	*breve*	*acc*
Belli	0	42	17	251	4	**63**
Eremita	0	127	61	**167**	7	34
Fiorino	**8**	62	19	295	0	25
Luzzaschi	0	65	15	**348**	5	11
Macque	0	**265**	48	170	1	34
Massaino	0	173	40	248	**12**	36
Perue	2	**35**	11	168	0	21
Spontone	2	73	28	269	2	**7**
Strigio	0	252	**85**	271	2	29
Zoilo	0	27	**2**	187	2	25
30M	60	3222	958	7399	117	817
mean (30M)	2	107.4	31.9	246.6	3.0	27.3
sd (30M)	2.4	66.8	19.7	52.8	3.6	14.4

Table 1: Occurrence of sixteenth- (16th), eighth- (8th), quarter dotted- (4th.), half- (2nd), and double whole- (breve) notes, as well as accidentals (acc) within the madrigals (identified by composer name). Max. and min. values, for occurrences across the dataset, are highlighted in bold. Mean and standard deviation (sd), are given considering all madrigals together (30M).

gals. Some madrigals are virtuosic, i. e., showing more 'syncopation' (rhythms off-beat generally represented in music by dotted-notes), and fast notes (sixteenth- and eighth-notes). Others are more 'sustained', i. e., showing more long notes (double whole-notes), or are 'harmonically' more unstable, i. e., showing more 'accidentals' (notes of a pitch that do not belong to the scale declared in the key signature). To illustrate the distributions of notes and accidentals, in Table 1 statistics for specific madrigals are given which include extreme occurrences (maximum and minimum values), as well as across all madrigals in the data set.

The considered anthology presents a balanced distribution of voice types: 15 of the 30 madrigals are composed for 'medium' vocal range (range from *baritone* to *mezzo-soprano*); the other 15 are composed for 'extreme' vocal range, i. e., 7 for 'high' range (*tenor* to *soprano*), and 8 for 'low' range (*bass* to *contralto*). Two of the 15 madrigals for 'medium voices' (those composed by Marenzio and Luzzaschi), display the maximum 'extension' (i. e., vocal range considering all five voices) of the anthology (between G2 – 97.9 Hz, and G5 – 783.9 Hz). The highest note performed is A5 – 880 Hz, being present only in Massaino's madrigal; whereas the lowest is E2 – 82.4 Hz, performed in the madrigal composed by Spontone.

4. SYMBOLIC SCORE CODIFICATION

4.1 Original Notation and Modern Transcription

The original notation in which the madrigals of *Il Lauro Secco* have been written in the 16^{th} century is the *white mensural* notation (cf. Figure 2) [2]. Two editions of this musical source in early notation are available [18], both digitised and freely available online. The first was printed in 1582 by Vittorio Baldini in Ferrara (Italy) and is available from the Music Library of Bologna [17] as well as from

[17] http://www.bibliotecamusica.it/cmbm/scripts/gaspari/scheda.asp?id=7156

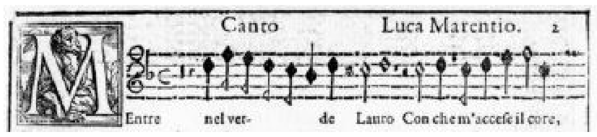


Figure 2: First staff of Marenzio's madrigal of the first edition (1582) of *Il Lauro Secco* written in *white mensural* notation.

IMSLP [18]. The second, printed in 1596 by Angelo Gardano in Venice, is available from the Gallica Digital Library [19]. Both editions have been used in the codification of the symbolic scores, collecting missing information of the first from the second when necessary and vice versa.

Based on the original source, two transcriptions have been made: one in *white mensural* notation (early notation), and another in modern notation. Both types have been chosen for their inherent advantages, and are available in Lilypond format. Since proficiency in early notation requires a level of musicological expertise, rare even in subjects from the musical field, symbolically codified transcriptions in modern notation are essential, offering a more understandable version of the repertoire.

On the other hand, the codified transcriptions in early notation, having the same notation as in the original source, provide the ground truth necessary to evaluate the performance of OMR systems (cf. Figure 3). Furthermore, since early notation do not split the notated music in 'measures' (segments within the 'staff' delimited by bar lines), 'ties' (the symbol used to link notes with the same pitch across different measures), are not required. This means that the symbolic representation of rhythm is always exactly the same, and never made of different note symbols, something typical of modern notation (cf. Figure 4). Since in modern notation, the codification of a given rhythm within a measure is different from the one across two measures, scores encoded in early notation would be more suitable for musical pattern recognition.

4.2 Musical Criteria

Even though in the original scores the individual vocal lines are written on different sheets, when engraving visually the proposed codified versions in Lilypond format (for both modern and early notation), the five voices are placed vertically superimposed (cf. Figure 3); the same holds for the modern transcription encoded in Finale. Computationally this does not make any difference, but we chose this arrangement because, from the musicological and linguistic point of view, vertical alignment is essential for effective analysis.

As early notation does not present 'bar lines', these are not considered in the scores encoded in Lilypond format, neither for early nor for modern notation (to allow for a visual comparison between both). Nevertheless, since bar lines are typical (if not mandatory) for modern notation, dashed bar lines have been considered incorporated in the

[18] http://imslp.org/wiki/Il_Lauro_secco_(Various)

[19] http://gallica.bnf.fr/ark:/12148/btv1b8449068j

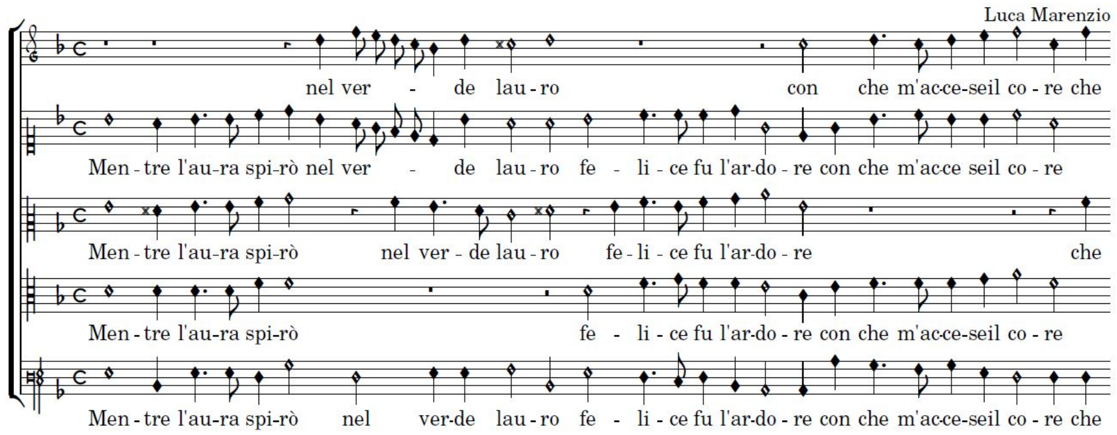

Luca Marenzio
nel ver - de lau-ro con che m'acce-seil co-re che
Men-tre l'au-ra spi-rò nel ver - de lau-ro fe - li-ce fu l'ar-do-re con che m'acce seil co-re
Men-tre l'au-ra spi-rò nel ver-de lau-ro fe-li-ce fu l'ar-do-re che
Men-tre l'au-ra spi-rò fe - li-ce fu l'ar-do-re con che m'acce-seil co-re
Men-tre l'au-ra spi-rò nel ver-de lau-ro fe - li-ce fu l'ar-do-re con che m'acce-seil co-re che

Figure 3: Visual representation of the transcription in *white mensural* notation (early notation) of the first staff of Marenzio's madrigal encoded by Lilypond. Unlike the original source, the voices are visually superimposed.

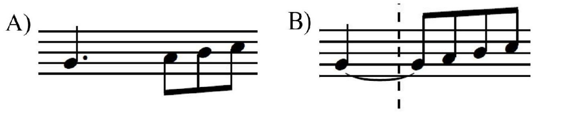


Figure 4: Two symbolic representations of the same rhythm. A) within a measure (encoded in Lilypond by: g4. a8 b8 c8); B) across two measures (in Lilypond: g4 ∼ g8 a8 b8 c8).

modern notated scores encoded in Finale (as commonly applied for modern transcription of early repertoire).

In early notation, accidentals are not always clearly indicated. Due to this, in critical editions of early repertoire, a 'cautionary accidental' (accidental placed above the note), is usually given by the musicologist as a suggestion. However, even these suggestions are given based on musical rules, such as consonances and dissonances created by the vertical collisions between notes, many times there is no full agreement between musicologists. Indeed, 'cautionary accidentals' can be displayed even by the musicologists themselves in two different ways, i. e., enclosed in parentheses above the note (when suggested), or without parentheses (when strongly suggested).

Based on these considerations, in the scores encoded in Lilypond, Music XML, and MIDI format, only the accidentals shown in the original source will be taken into account; whereas in the scores encoded in Finale, cautionary accidentals (both enclosed within parentheses or not), have been included to assist musicological analysis and potential musical performance. Furthermore, the symbols given for the accidentals in the original source (sharps and flats), had been respected in the early notated transcriptions, whereas these have been changed into naturals, when necessary, in the modern notated transcriptions (according to modern notation rules).

In mensural notation, 'ligatures' are groups of notes encoded with a unique graphical symbols. The interpretation of ligatures is made according to specific rules, and the notes involved are at least semibreve, i. e., only 'long' notes are involved [2]. Ligatures are relatively rare in the presented repertoire, being only 18 in the 30 madrigals (consider that each madrigal has at least 550 note symbols). Moreover, ligatures are never involved in musical-linguistic patterns (since these are made up of shorter notes). For these reasons, we encoded ligatures as single notes instead of a unique graphical symbol. In the scores encoded in Finale, a square bracket has been used to indicate the notes originally involved in the ligature (as is usual in modern transcriptions of early repertoire).

Finally, long rests (e. g., maxima rest), have been codified differently lasting a maximum of the whole rest, i. e., whole measure. This is the normal practice in modern notation, but not in early notation, where values are not determined by measure length. However, in order to save encoding time, and given that neither long rests nor graphical appearance have a role for musical analysis purposes, this practice has been adopted for the encoding in both early and modern notation.

4.3 Linguistic Criteria

In the original source, lyrics are placed in two locations of the score: under the notated music (for each one of the five voices), and in a poem format at the left of each music-sheet. Differences between these lyrics are typical of this repertoire, e. g., random use of abbreviations, missing accents and punctuation, or different spelling of the same word (cf. Figure 5). These inconsistencies create a challenge for OMR, and make automatic analysis an extensive task (since no musical-linguistic patterns can be identified in a non-unified text). For this reason, to encode this repertoire according to a uniform version, considering musical-linguistic criteria is essential.

Differences between the first edition of the source (1582) and the second (1596) have been found as well, the reprinted version being characterised by the use of more 'textual contractions' (e. g., *verd'io* instead of *verde io*, or *sott'ai* instead of *sotto ai*). Evaluating this, in the presented dataset, the standardisation of the lyrics has been made according to the first edition of the anthology (1582), and the lyrics have been presented only under the musical notation. The following linguistic criteria have been considered [11]:

I. Linguistic aspects faithful to the Italian language of the 16^{th} century:

A) The etymological 'h' that does not produce pronunciation changes (e. g., in 'hor'), has been conserved;

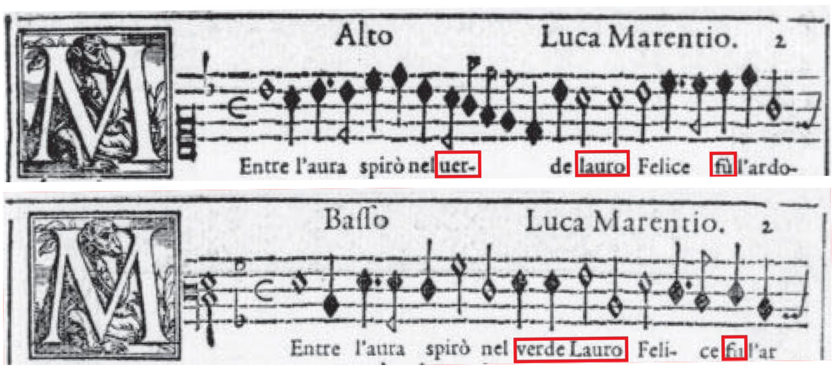

Alto
Luca Marentio.
2
Entre l'aura spirò nel uer- de lauro Felice fù l'ardo-
Baſſo
Luca Marentio.
2
Entre l'aura spirò nel verde Lauro Feli- ce fu l'ar

Figure 5: First staff of the Marenzio's madrigal for *Alto* (shown above) and *Basso* (shown below) voices. The inconsistencies of the lyrics between both voices are highlighted: *uerde* vs *verde*, *lauro* vs *Lauro*, and *fù* vs *fu*, between *Alto* and *Basso*, respectively.

B) The graphical symbol 'ti', that must be pronounced 'zi' according the modern Italian phonetic rule, has been conserved;

C) The *tironian* symbol '&' has been transcribed as 'et', according the Italian orthographic rule of the 16th century;

D) In the cases where contracted and not contracted textual versions have been presented (e. g., *altrov'adopra* and *altrove adopra*), the not contracted version has been considered. Nevertheless, in the musical performance, the synalepha, i. e., to merge two syllables into one, has to be made.

II. Linguistic aspects faithful to the modern Italian language:

A) The diacritic mark has been normalised according to the modern rule (e. g., *'più'* instead of *'piu'*, and 'a' instead of 'à', cf. Figure 5);

B) The arbitrary use of 'u' and 'v' in the different voices has been normalised according to the modern rule (e. g., *verde* instead of *uerde*, cf. Figure 5);

C) The abbreviation of 'n' and 'm' as superscripts on vowels with ∼ has been normalised by the complete spelling (e. g., *hanno* instead of *hãno*);

D) The abbreviation of *'per'* through *'ꝑ'* has been normalised by the complete spelling (e. g., *perché* instead of *ꝑche*);

E) The abbreviation *'ij'*, referring to the repetition of sentences or words, has been substituted by the complete form;

F) Separated words have been normalised according to the modern rule (e. g., *invano* instead of *in vano*, or *poiché* instead of *poi che*).

III. Linguistic aspects considered in order to allow automatic musical-linguistic pattern recognition:

A) The punctuation has been standardised in all the voices, considering the prosody of the text but at the same time encouraging its simplification in order to allow musical-linguistic pattern recognition (where normalised punctuation between the different voices is essential);

B) The use of capital- and minor-letters has been standardised in all the voices, considering capital–case at the beginning of each verse and personification (cf. Figure 5). In order to prioritise the coherence between the different voices, vertical collisions between musical-linguistic patterns have been considered. According to this, the starting word of the repetitions of verses has been also capitalised.

Finally, melismatic prosody between syllables of the same word (i. e., a single syllable of text is sung through several different notes), has been graphically identified by dashes for both early and modern notation, as in the original source. However, when the melisma is placed at the end of the word, no graphical indication has been given in the early notated scores, following the original source. On the contrary, for the transcription in modern notation (both encoded in Lilypond and Finale), the length of the melisma has been indicated by an underscore.

5. DIGITAL FORMATS EVALUATION

As mentioned, the 30 madrigals have been encoded in four digital formats: Lilypond, Music XML, MIDI, and Finale. Early and modern notation are available in Lilypond format (a total of 60 files), whereas the Finale format has been considered only to encode modern notated transcriptions (30 files), and from these, Music XML and MIDI files have been automatically created (30 for each).

Each format has differing pros and cons for computational musicology. For example, Music XML files show clear links between linguistic information and associated notes, which helps for the automatic identification of musical-linguistic connections. In the following, we show the Music XML code (Code 1), used to indicate the first note of the *Alto* voice in the transcription in modern notation of Marenzio's madrigal (the original early notated version of this is shown in the top staff of Figure 5).

Code 1: Music XML syntax

```
<note default-x="121">
  <pitch>
    <step>B</step>
    <alter>-1</alter>
    <octave>4</octave>
  </pitch>
  <duration>8</duration>
  <voice>1</voice>
```

```
9    <type>whole</type>
10     <lyric default-y="-80" number="1">
11       <syllabic>begin</syllabic>
12         <text>Men</text>
13       </lyric>
14   </note>
```

As we can see, each line of the code indicates a specific musical parameter (e. g., line 3 the pitch, line 9 the note, line 12 the syllable). Nevertheless, this disposition breaks up the continuity of the musical patterns, complicating the performance of automatic analysis.

In contrast, Lilypond files have a clearer distribution of the musical patterns over the code lines, according to each measure (indicated in the following Lilypond code, i. e., Code 2, by "| %"). This facilitates computational operations such as automatic identification of rhythmic-melodic patterns, especially in scores encoded in early notation (where a given rhythm never indicates different shapes). In the following, we show the Lilypond code (Code 2) used to indicate the first staff of the *Alto* voice in the transcription in modern notation of Marenzio's madrigal (the original version of this, is shown in the top staff of Figure 5).

Code 2: Lilypond syntax

```
1    \key f \major
2    \time 4/4
3    \autoBeamOff
4      bes'1        | % 1
5      a4 bes4. bes8 c4        | % 2
6      d bes a8 g f e        | % 3
7      d4 bes' a2        | % 4
8      a bes        | % 5
9      c4. c8 c4 d        | % 6
```

As we can see, in each line of Lilypond, a whole measure is encoded, giving a more compact and meaningful distribution of the music. Indeed, whereas in 14 lines of Music XML, only one note is encoded, in the 9 lines of Lilypond, 20 notes are encoded, i. e., the whole first staff. In these 9 lines, not only the note length is encoded but also the pitch, accidentals, and octave (e. g., "4" means quarter-note, "bes" means B flat, and " ' " indicates the 4th octave), as well as additional graphical information (e. g., "\autoBeamOff" indicates not to link the eighth-notes by a beam, typical of modern notation).

However, in Lilypond format, the lyrics are described in a different section of the code respectively to the notes, and without measure wise alignment. The link between notes and syllables is given by a single space to indicate that the following syllable is aligned to the following note and does not belong to the same word. To link syllables of the same word that are aligned to different notes the command "−−" is used (rests are not considered). In a melismatic passage, to indicate that an extra note must be skipped, the command "\skip4" is used. Following this, the first verse sung by the *Alto* voice in the Marenzio's madrigal is encoded in Lilypond as follows (the original early notated version of this, is shown in the top staff of Figure 5):

Men −− tre l'au −− ra spi −− rò nel ver −− \skip4 \skip4 \skip4 \skip4 de lau −− ro

MIDI is probably the most common digital format for music dissemination in the web, being also used in computational approaches as pattern identification on polyphonic music [19]. Nevertheless, early music is almost completely overlooked in the repertoire presented in this digital format. As well as MIDI files, Finale files are also a standard format always more common in digital music libraries. However, again this format is popular in sharing codified scores from other 'classical' musical periods but not for Renaissance music. With this in mind, we included in the presented dataset MIDI and Finale files.

Beyond the symbolically codified files, a total of 180 pdf files have also been included. From these, 30 pdf are the modern notated transcriptions of the Finale encoded madrigals (to gain an easier evaluation of the repertoire). The other 150 pdf are scanned copies of the first edition of the original source (5 pdf files for each madrigal, one for each voice). In total, the SEILS dataset encompasses 330 files: 180 of them are pdf files; whereas the remaining 150 are symbolic files digitally encoded in different formats. Of these 150 symbolic files: 60 are encoded by Lilypond (.ly), 30 in each of the considered notations (early and modern); 30 are encoded by Music XML (.xml); 30 by MIDI (.mid); and 30 by Finale (.musx).

6. CONCLUSIONS

The presented dataset of codified scores aims to encourage automatic musical analysis in Renaissance music. Considering the strong connections between music and poetry of the chosen repertoire, the presented dataset is specifically suitable for creating synergies between musicology and linguistics. We present symbolically encoded scores of the *Il Lauro Secco* anthology considering the original *white mensural* early notation, which will allow for the evaluation of OMR performance.

Since each digital format has some advantages and disadvantages, it is our belief that through this combination, each limitation found in the formats can be overcome (e. g., by combining Lilypond and Music XML files, it is possible to clearly identify lyrics with musical patterns). With this in mind, the SEILS dataset makes available a variety of digital formats: Lilypond, Music XML, MIDI, and Finale.

Our next priority is to complete the analytic annotation of the presented dataset in **kern format, through the identification of different types of madrigalisms (e. g., based on contrapuntal and homorhythmic textures, or in consonant and dissonant vertical sonorities, among others), within each madrigal.

7. ACKNOWLEDGEMENT

This work was supported by the European Unionn's Seventh Framework and Horizon 2020 Programmes under grant agreements No. 338164 (ERC StG iHEARu) and No. 688835 (RIA DE-ENIGMA).

8. REFERENCES

[1] C. Antila and J. Cumming. The vis framework: Analyzing counterpoint in large datasets. In *Proc. of ISMIR*, pages 71–76, Taipei, Taiwan, 2014.

[2] W. Apel. *The notation of polyphonic music, 900-1600*. Medieval Academy of America, Cambridge, UK, 1961.

[3] J. A. Burgoyne, Y. Ouyang, T. Himmelman, J. Devaney, L. Pugin, and I. Fujinaga. Lyric extraction and recognition on digital images of early music sources. In *Proc. of ISMIR*, pages 723–727, Kobe, Japan, 2009.

[4] L. Chen, E. Stolterman, and C. Raphael. Human-interactive optical music recognition. In *Proc. of ISMIR*, pages 647–653, New York, NY, USA, 2016.

[5] M. S. Cuthbert and C. Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proc. of ISMIR*, pages 637–642, Utrecht, Netherlands, 2010.

[6] M. Droettboom, I. Fujinaga, K. MacMillan, G. S. Chouhury, T. DiLauro, M. Patton, and T. Anderson. Using the gamera framework for the recognition of cultural heritage materials. In *Proc. of the 2nd ACM/IEEE-CS*, pages 11–17, Portland, OR, USA, 2002.

[7] E. Durante and A. Martellotti. *Madrigali segreti per le dame di Ferrara: Il manoscritto musicale F. 1358 della Biblioteca Estense di Modena*. Studio per edizioni scelte, Firenze, Italia, 2000.

[8] E. Durante and A. Martellotti. *Giovinetta peregrina: La vera storia di Laura Peperara e Torquato Tasso*. LS Olschki, Firenze, Italia, 2010.

[9] A. Einstein. *The Italian Madrigal*. Princeton University Press, Princeton, NJ, USA, 1971.

[10] I. Fujinaga and A. Hankinson. Simssa: Single isnterface for music score searching and analysis. *Journal of the Japanese Society for Sonic Arts*, 6(3):25–30, 2005.

[11] G. Gialdroni. *Di Giovanni Battista Moscaglia. Il secondo Libro de' Madrigali a Quattro Voci*. Fondazione Pierluigi da Palestrina, Palestrina, Italy, 2007.

[12] M. Giuliani. *I lieti amanti: Madrigali di venti musicisti ferraresi e non*. Leo S. Olschki, Firenze, Italia, 1990.

[13] A. Goodman and A. MacKay. *The impact of humanism on Western Europe*. Taylor and Francis, London, UK, 2013.

[14] D. J. Grout and C. V. Palisca. *A history of western music*, volume 1. Norton, New York, NY, USA, 2001.

[15] A. Hankinson, J. A. Burgoyne, G. Vigliensoni, and I. Fujinaga. Creating a large-scale searchable digital collection from printed music materials. In *Proc. of the 21st Int. Conf. on World Wide Web*, pages 903–908, Lyon, France, 2012.

[16] A. Hankinson, J. A. Burgoyne, G. Vigliensoni, A. Porter, J. Thompson, W. Liu, R. Chiu, and I. Fujinaga. Digital document image retrieval using optical music recognition. In *Proc. of ISMIR*, pages 577–582, Porto, Portugal, 2012.

[17] D. Huron. Music information processing using the humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[18] F. Lesure. *Recueils imprimes XVIe-XVIIe siecles*. Henle, Munich, Germany, 1960.

[19] B. Meudic and E. St-James. Automatic extraction of approximate repetitions in polyphonic midi files based on perceptive criteria. In *Int. Symp. on Computer Music Modeling and Retrieval*, pages 124–142, Montpellier, France, 2003.

[20] A. Newcomb. The three anthologies for laura peverara, 1580–1583. *Rivista Italiana di Musicologia*, 10:329–345, 1975.

[21] A. Newcomb. *The Madrigal at Ferrara: 1579-1597*. Princeton University Press, Princeton, NJ, USA, 1980.

[22] H.-W. Nienhuys and J. Nieuwenhuizen. Lilypond, a system for automated music engraving. In *Proc. of the 14th Colloquium on Musical Informatics*, volume 1, pages 167–172, Firenze, Italia, 2003.

[23] V. Padilla, A. McLean, A. Marsden, and K. Ng. Improving optical music recognition by comining outputs from multiple sources. In *Proc. of ISMIR*, pages 517–523, Málaga, Spain, 2015.

[24] N. Pirrotta. *Dolci affetti: I Musici di Roma e il madrigale*. L. S. Olschki, Firenze, Italia, 1985.

[25] L. Pugin and T. Crawford. Evaluating OMR on the early music online collection. In *Proc. of ISMIR*, pages 439–444, Curitiba, Brazil, 2013.

[26] L. Pugin, J. Hockman, J. A. Burgoyne, and I. Fujinaga. Gamera versus Aruspix – two optical music recognition approaches. In *Proc. of ISMIR*, pages 419–424, Philadelphia, PA, USA, 2008.

[27] E. Ricciardi. The tasso in music project. *Early Music*, 43(4):667–671, 2015.

[28] P. Roland, A. Hankinson, and L. Pugin. Early music and the music encoding initiative. *Early Music*, pages 605–611, 2014.

[29] C. S. Sapp. Online database of scores in the humdrum file format. In *Proc. of ISMIR*, pages 664–665, London, UK, 2005.

[30] A. Vassalli. Il tasso in musica e la trasmissione dei testi: alcuni esempi. *Tasso, la Musica, i Musicisti*, pages 45–90, 1988.

[31] V. Viro. Peachnote: Music score search and analysis platform. In *Proc. of ISMIR*, pages 359–362, Miami, FL, USA, 2011.

[32] J. A. Winn. *Unsuspected eloquence: A history of the relations between poetry and music*. Yale University Press, New Haven, CT, USA, 1981.

Poster Session 3

“I’M AT #OSHEAGA!”: LISTENING TO THE BACKCHANNEL OF A MUSIC FESTIVAL ON TWITTER

Audrey Laplante
Université de Montréal
audrey.laplante@umontreal.ca

Timothy D. Bowman
Wayne State University
timothy.d.bowman@wayne.edu

Nawel Aamar
Université de Montréal
nawel.aammar@umontreal.ca

ABSTRACT

It has become common practice for audience members to use social media to connect, share, and communicate with each other during events (e.g., sport events, elections, award ceremonies). But how is this backchannel used during a musical event and what does it say about how people engage with the music and the artists performing it? In this paper, we present the result of a study of a dataset composed of 31,140 tweets posted during and around the 10th edition of Osheaga, an important music festival held annually in Montreal. A combination of statistics and qualitative content analysis is used to examine the postings. This allows us to describe the content of these postings (i.e., topics, shared media), the type of message being shared (i.e., opinion, expression, information), and who the authors of these tweets are.

1. INTRODUCTION

Music tastes are for many an important dimension of their sense of self, particularly during adolescence and young adulthood. People often use their music tastes as a ‘social badge’ of their identity [1, 2], which tells others who they are or who they aspire to be. Disclosing our music preferences is therefore an exercise of taste and discrimination. This is certainly one of the reasons music recordings have not displaced live musical performances; attending a music show is one of the strongest ways of showing others our love of music and/or of a particular music artist [3]. It is also the occasion to buy T-shirts, posters, or other mementos to testify that we were there. Music tastes also play an important role in the construction of group identity. Concert going, as a social outing, is therefore also an opportunity to share an experience that could reinforce a friendship or a romantic relationship.

Social media have further amplified the role of music tastes in identity formation. By providing tools that allow their users to share their cultural preferences in various ways, users can now display their ‘social badge’ to a broader audience composed of friends, relatives, coworkers, acquaintances, or even unknown people. By doing so, they make a ‘taste statement’ that is used for ‘taste performance’, as an expression of prestige [4]. Furthermore, social media afford users a means of connecting with other concert-goers and potentially even with the performing music artists. It has become common for the organizers of important events to provide an official hashtag so that audience members can connect and participate in a shared conversation about the event. But how do people use these affordances?

Music appears to be a common topic on Twitter; the hashtag #nowplaying, used to indicate the music a user is currently listening to, was the 6th most popular hashtag from 26 March to 25 April 2017[1]. Several musical events, from televised music award ceremonies and contests to music festivals now propose their own official hashtag. However, very few studies have examined the content of these tweets.

This study focuses on the use of Twitter during an important musical event, the 10th edition of Montreal’s Osheaga festival. Using both quantitative and qualitative methods, we analyzed 31,140 tweets with the aim of exploring the following research questions:

RQ1. Who tweeted during the event and who were they speaking to?

RQ2. What is the content of these messages (i.e., topic, media)?

RQ3. Are these messages objective or subjective?

RQ4. Which events, shows, or artists during the festival generated the most tweets?

Garnering more information about the content and the authors of these tweets could provide some insights into how people engage with music and what they have to say about it, about the artists performing it, and about the fans. Since our reception of music depends not only on the inherent characteristics of the music itself but also on its social and cultural context, it seems relevant to examine what type of information user-generated content related to music could provide and how it could help us better understand how music tastes are shaped. Moreover, according to surveys conducted by The Nielsen Company [5, 6], large music festivals have been gaining in popularity in Canada and in the United States. American music festival-goers were 98% more likely than the average American to discover new music on Spotify, the music streaming service, and nearly half of them shared photos and/or texted friends while attending a concert. This suggests that having a better understanding of the music con-

[1]*Hashtagify*. http://hashtagify.me

sumption and perception of this growing user group could be particularly relevant for the design of music recommender systems.

2. RELATED WORK

2.1 Twitter As a Backchannel

Social media provide a fertile ground for research. These online contexts offer a public or semi-public space where people can connect and share a conversation in real time. Therefore, despite the recency of these platforms, there are already numerous studies that have focused on the use of Twitter for various purposes, including its use for connecting with other audience members during major events. Researchers have looked at Twitter use during televised events [7-9]. Wohn and Na [8] examined the messages posted on Twitter during two televised events, a talent show and a political speech. They manually coded the postings into four categories: emotion, attention, information, and opinion. Their analysis revealed that the most popular category was Opinion, in which more than 30% of the postings were coded for both programs. Also using a qualitative approach to content analysis, Giglietto and Selva [7] looked at Twitter activity during a full season of a political talk show. Again, opinion expression was the most important tweet category: it accounted for 59% of the postings. Their study also revealed that Twitter could be useful in identifying the most engaging moments of such shows. Bruns and Stieglitz [9] employed statistical methods to examine the audience activity on Twitter, minute per minute, during the television broadcast of the British Royal Wedding. This allowed them to determine that there was a strong correlation between Twitter activity and key moments during the ceremony.

These studies suggest that, thanks to the affordances of Twitter and other social media platforms, the audience has taken a more active role. Twitter serves as a backchannel—or as a ‘second screen’ [7]—that complements the broadcasting media and allows the broadcasters to receive audience feedback in real time [10].

2.2 Music-Related Twitter Studies

In the same line of research, Highfield et al. [11] used a combination of quantitative and qualitative methods to examine Twitter activity during a major musical event that is broadcasted internationally: the Eurovision Song Contest. They found that broadcasters encouraged the use of Twitter, for example by promoting an official hashtag for the event and, in some cases, by selectively showing user tweets on screen. This indicates that there is a real interest from broadcasters and the organizers to receive live feedback from the audience and to facilitate audience engagement. Their study also showed the potential of Twitter for establishing and supporting fan communities.

A few studies have also been conducted within the MIR community. Hauger et al. [12] presented the ‘Million Musical Tweets Dataset’ (MMTD), a dataset composed of tweets collected using music-related hashtags. Since all tweets have geo-location data, the researchers used the dataset to geographically represent listening preferences. The MMTD has been used other researchers. Moore et al. [13] employed probabilistic embedding methods to uncover geographic and cultural patterns in it, and Farrahi et al. [14] explored the potential of Twitter data for improving the collaborative filtering approaches used by music recommender systems. Zangerle et al. [15] presented another dataset, the ‘#nowplaying Music Dataset’. Kim et al. [16] used this dataset to examine the relationship between the Billboard rank and play counts extracted from Twitter postings. A strong correlation between the two was found. Finally, Iren et al. [17] released the ‘Top 2000 Dataset’ composed of tweets posted in connection with the Top 2000, a yearly event broadcasted on the radio in the Netherlands for which the public is invited to vote for the greatest 2000 songs of all times.

The interest the MIR community has already demonstrated for Twitter data is an indication of the potential it has in helping us better understand users’ music behaviour and music tastes, with the objective of improving music recommender systems.

3. OSHEAGA

Created in 2006 by Evenko, the Festival Musique et Arts Osheaga is one of the most important music festival in Canada. Held annually in Montreal during the summer, the festival hosts more than 100 music artists across three days each year. While it focused on local underground music artists in the beginning, Osheaga has been hosting international artists for several years now. The festival offers a varied programme that covers different music genres, including rap, indie, and electronic music. In addition to the concerts, the festival offers on-site activities as well as visual art installations. Gaining in popularity, Osheaga attracts visitors from all over the world each year, most of whom are between 20 and 25 years old. In 2016, 65% of the 135,000 festival-goers came from outside Quebec [18, 19].

4. METHODS

4.1 The Dataset

To examine how people used Twitter during and around the Osheaga music festival, we collected the tweets related to the 2015 edition of the festival, which was held from July 31 to August 2, 2015. Although the festival itself did not promote the use of any official hashtag on its website, the hashtag #Osheaga2015 was included in many postings made by the festival on Twitter. People also used the more generic #Osheaga hashtag. Therefore, from July 24, 2015 to August 13, 2015, we collected the tweets that contained at least one of these two hashtags, as well as tweets that contained the Twitter handle of the festival, @osheaga (i.e., the username of the official account of Osheaga on Twitter). The final dataset was composed of 31,140 tweets.

4.2 Data Analysis

A mixed-methods approach was used to analyse the data. With our research questions in mind, we calculated descriptive statistics, to which we added the activity, visibility, and temporal metrics defined in [9].

To capture the richness of the postings, we employed a grounded theory approach to content analysis, which means that we let the categories emerge from the data, without imposing any preconceived model on it [20]. For this part, we focused on the tweets posted during the festival (from July 31 to August 2). We also limited our analysis to original tweets, which means that retweets were excluded. These will be analysed separately but, due to the limited length of this paper, this analysis is not included here. Since manual coding is time consuming, we chose to focus on a random sample stratified by date. More specifically, we randomly selected 5% of the postings published on each of the three days of the festival. In total, 712 postings were manually coded (see Table 1).

Posting date	No. of original tweets	No of postings analyzed
July 31	3,377	169
Aug. 01	4,778	239
Aug. 02	6,084	304
Total:	14,239	712

Table 1. Description of the dataset that was manually coded.

Qualitative content analysis is a multi-step and iterative process. The first step consisted in developing the codebook, which was done by coding 100 postings that were not included in the final sample [21]. In the next step, two researchers used the codebook to independently code the first 100 postings of the sample in order to test it. The analyses of the two coders were then compared and discussed. This led to a revised and final version of the codebook, which was composed of 66 categories. The coding of the first 100 postings was revised and the 612 remaining postings were coded. Coding each posting took time. For each posting, the coder accessed the user profile to determine what type of user it was (e.g., individual, broadcaster, promoter). If the tweet contained URLs, the coder had to follow them to see where they led. Moreover, the coder had to make sense of the content of the text. Multiple codes could be applied to one message.

5. RESULTS

5.1 Who Participates in the Conversation?

As mentioned before, Twitter affordances invite users to connect and converse with other people attending the concert, with people who could not or did not want to be there, and even with the performing artists. But in reality, who participates in this shared conversation?

Visibility. Our dataset was composed of 31,140 tweets. These tweets were posted by 12,294 distinct users, for an average of 2.5 tweets by user. However, a closer look shows an uneven distribution: a very small number of users accounted for a large proportion of the postings. More specifically, the top 1% of most active users accounted for 17.5% of the tweets, and the top 10%, for 44.8%. Conversely, we find a long tail of users with little activity. Indeed, 7,202 (58.6%) of users had posted only one message during the festival.

Categories of users. The coding process for content analysis included accessing the Twitter account of the author of each message in order to categorize it (see Table 2). Individuals accounted for 74.2% of the postings. The next two most important categories were reporters, bloggers, TV/radio hosts, and photographers, who authored 10.8% of the tweets, and magazines, newspapers, blogs, and TV/radio stations, who posted 4.9% of the tweets. Different types of societies (e.g., restaurants, clothing companies) posted some tweets, usually for promotional purposes. The festival itself posted 2.5% of the messages of our sample.

Category of users	No. of tweets	% (*n*=712)
Individuals	530	74.4%
Reporters, bloggers, TV/radio hosts, and photographers	77	10.8%
Magazines, newspapers, blogs, and TV/radio stations	35	4.9%
Societies	23	3.2%
Osheaga	18	2.5%
Music artists (performing during the festival or not)	13	1.8%
Promoters	9	1.3%
Music producers and labels	7	1.0%
Total	712	100.0%

Table 2. Tweets by user category.

5.2 Who Are They Speaking To?

Mentions. In the language of Twitter, a mention is a reference to a user in a tweet using his or her Twitter handle (e.g., @osheaga). Of the 31,140 tweets in our dataset, 16,773 (53.9%) included at least one mention. There was a total of 25,746 mentions. Postings included between 0 and 9 mentions, for an average of 0.83 mention and a median of 1 mention per posting.

Mentions were used in different ways, sometimes for addressing a tweet to a specific user:

```
Hey @b### are you at #OSHEAGA2015 this week-
end?
```

to tag someone in a photo or in a posting:

```
#day1 @osheaga with my main girl @L######
#stayhydratedfolks #osheaga #ootd @ Parc
Jean-Drapeau [followed by a link to a photo
of the two friends]
```

or to tag the performing artists of the concerts they are attending:

```
#OSHEAGA2015 Day 3 Wrap-up @GaryClarkJr
@Bobmosesmusic @SylvanEsso @sanferminband
@charli_xcx @TheWarOnDrugs @Hot_Chip @alt_J
@theblackkeys
```

The number of mentions a user receives is an indication of his or her visibility. In our dataset, 3,023 distinct users received at least one mention. A few users received a high number of mentions. With 6,360 mentions (24.7%), the Twitter account of the festival received the most mentions, which is not surprising considering that this was one of the criteria for collecting the tweets. If we put the festival mentions aside and examine the remaining 19,286 mentions, we notice that the top 1% of most mentioned users accounted for 32.7% of the tweets. They had received between 109 and 701 mentions each. Half of these top users were artists who performed during the festival (e.g., Kendrick Lamar, James Bay, Of Monsters and Men). Among these users were also online magazines and blogs (e.g., Sidewalk Hustle, Much), and music streaming services (e.g., Stingray Music, Spotify Canada). We also find two celebrities who attended the festival but had no official role to play in it: a local pop signer (i.e., Marie-Mai) and an international top model (i.e., Cara Delevingne) whose agency had posted several photos of her at the festival on Twitter.

Artists. But what role did the performing artists take in the conversation? Users regularly mentioned the names of the artists that were performing in their tweets. Among the postings that were manually coded, 231 or 32.4% contained a reference to a performing artist. However, the users did not always used the Twitter handle of the artist to do so. More precisely, they used the Twitter handle 40.7% of the time. This suggests that most of the time, the user was not expecting any reaction from the artist. But even when a Twitter handle was used, the user did not always explicitly address his/her message to the artist. Indeed, if some users talked directly to the artist, as in this message:

```
@MarinasDiamonds we loved your set at
#OSHEAGA2015 and we would love to hug you
and wish you well :) you're the best
```

most talked about the artist at the 3rd person, as in this message:

```
@flo_tweet has the most amazing voice. I'm
in awe of this woman #OSHEAGA2015 [followed
by a link to a photo]
```

Among the 123 singers and groups who performed during the festival, two had no Twitter account. According to our sample, a large majority (81 or 66.9%) of those who were Twitter users had not posted any tweets about Osheaga during the data collection period; they may have tweeted about the festival, but they did not use the hashtags or mentions we queried in the collection process. The remaining 40 artists (33.1%) had posted between 1 and 14 tweets each, for an average of 3.2 and a median of 2 postings per artist. In total, our dataset included 129 postings from performing artists. These tweets were all manually coded. Eighty-five (65.9%) of these messages were retweets. As we can expect, the original tweet often consisted of a positive review of the artist's performance. The tweets came mostly from blogs, radio/TV stations, newspapers, music services, reporters/bloggers, or the festival itself, but there were a few cases (12) where the artists retweeted a fan's posting.

Among the 44 postings that were not retweets were 13 thank you notes to the festival or the fans in general, such as:

```
That was one of our favourite shows this
summer @osheaga Thank you! #OSHEAGA2015
```

Some (10) used Twitter to announce that they were performing at the festival or doing their sound check. One music group shared a photo of its set list for the concert. There were only three postings that showed a direct interaction between an artist and a fan. For example, a fan had asked a music group (using its Twitter handle) to play a specific song, a request to which the band drily replied:

```
Not gonna happen
```

In another case, the tweet was a personal thank you to a fan. And in the final case, the singer shared a fan's video showing a blooper from his show and commented on it:

```
Hahaha that was such a fail [followed by the
link to the fan's tweet with the video]
```

This particular tweet was then retweeted 128 times by other users.

The low number of tweets that show a direct interaction between the artists and their fans should however be interpreted with caution: our dataset was composed of tweets containing two specific hashtags and one Twitter handle. It is possible that some artists replied to their fans without including those in their reply.

5.3 What Do They Tweet About?

Topics. The qualitative content analysis allowed us to closely look at the content of the messages that were posted on Twitter. The main topics are presented in Table 3. By far, the most common message was to announce that one was going to Osheaga. However, we must stress that many of these messages were not posted at the initiative of their author. The festival was encouraging festival-goers to register their bracelet online in order to win prizes and be able to take part in some activities on site. They could create a new account to sign up, or they could use Facebook or Twitter. Using Twitter apparently resulted in the application posting the following message on Twitter:

```
I'm at #Osheaga2015 Day 1 - powered by Sam-
sung Galaxy S6
```

Some changed it slightly. It could apparently also be done on site since many added to the message a photo taken in a dedicated space. These messages accounted for 29.6% of the dataset and 31.1% of the sample used for content analysis. Although these messages may appear to be spam, the fact that many users added a photo and/or did not make the effort to create a new account for the festival suggest that perhaps they wanted to share these tweets. Moreover, these messages were part of the conversation about the festival on Twitter: people reacted to and commented on these tweets, and they certainly created a 'hype' on Twitter considering the volume. For this reason, we decided to keep them for the analysis.

Registering the bracelet online was not the only incentive for sharing that one was attending the festival. Many did that on their own initiative, oftentimes adding a photo of their bracelet:

```
Off to osheaga #Osheaga #OSH15 [followed by
a link to a photo of the Osheaga bracelet]
```

A few users (25) also explicitly announced attending a concert:

```
I'm so excited today I'm gonna see two of my
favorite artists live! @MarinasDiamonds and
@twentyonepilots <3 #OSHEAGA2015
```

Questions, comments, and complaints were addressed to the festival, who would then reply to the users. Some people also shared personal experiences during the festival, like hurting themselves or stumbling upon a singer or musician:

```
Casually met the band of Florence & The Ma-
chine in the lobby of my hotel tonight
#OSHEAGA2015
```

In 12 cases, people commented on or complained about other festival-goers, about their appearance or their behaviour, as in:

```
Festival etiquette breach number one.
#osheaga #get #down #now @ parc jean drapeau
[accompanied by a photo of a person sitting
on someone else's shoulders]
```

```
all I see at #Osheaga is fake Kylie-Jenner-
styled people
```

Other topics, such as fashion, food, weather, and even books were also occasionally discussed, sometimes in combination:

```
Tacos in the rain? Why not! #osheaga #tacos
#festival #food [accompanied by a photo of
the tacos]
```

Topics	No. of tweets	% (*n*=712)
Presence at festival	271	38.1%
Performing artists and their music	231	32.4%
Festival (e.g., schedule, logistic, transport)	50	7.0%
Promotion of work, products, or services	46	6.5%
Presence at concert	25	3.5%
Fashion	17	2.4%
Personal experience	17	2.4%
Other festival-goers' behaviour	12	1.7%
Food	11	1.5%
Weather	10	1.4%
On-site activities	8	1.1%

Table 3. Main topics discussed in tweets posted during the festival

As seen in Section 5.1, festival-goers were not the only ones to take part in the conversation. Various societies used Twitter to promote their work, products, or services. For instance, some on-site restaurants and shops used Twitter as an advertising venue:

```
We are at #Osheaga! Come and see us
@C####### near the Scène des Arbres [accom-
panied by a photo of the food truck] (trans-
lated from French)
```

Some other retailers who were not on site, such as clothing companies, tailored their promotional message for the Osheaga festival-goers:

```
Dress it up or dress it down! This look is
easy to take from day to night. #ootd #toms
#friday #osheaga [accompanied by a photo of
an outfit from the clothing company]
```

Reporters, bloggers, photographers, and radio and TV hosts promoted their work differently, some by directly sharing the link to the result of their work—be it an newspaper article, a blog post, or a photo—others by announcing that they were covering the festival, such as in the following tweet posted by a TV reporter:

```
#C##### backstage at #osheaga with #patrick-
watson and string quartet #mommasontherun
[accompanied by a photo of the members of
the string quartet]
```

Media. Close to half (42.7%) of the 712 tweets that were manually coded contained or pointed to a non-textual resource (i.e., photo, video) (see Table 4). By far, the most often shared media type was the photo: 34% of the postings analyzed contained a photo taken by the author. Amongst the main categories of photos shared by users, whether their own or someone else's, were the following: photos of concert (36.4% of photos), selfies with others (26.3%), photos of festival site (12.3%), other festival-goers (7.2%), and selfies alone (5.5%). The vast majority of the videos shared were videos of a live performance taken during the festival.

Type of media shared	No. of tweets	% (*n*=712)
Personal photo	242	34.0%
Personal video	41	5.8%
Someone else's with photo	15	2.1%
Shares someone else's video	6	0.8%
Tweets with media in total:	**304**	**42.7%**

Table 4. Types of media shared in tweets posted during the festival

5.4 Are the Messages Objective or Subjective?

As mentioned in the introduction, research shows that people used their music tastes as a social badge that tells other people who they are, a phenomenon that has been exacerbated by social media who provide the sounding-box for such messages. Therefore, it seems reasonable to expect a large number of people using Twitter to express an opinion about the music they are listening to.

Of the 712 messages that were manually coded, 153 (21.5%) were explicit expression of an opinion, which is quite high considering the large proportion of tweets that were automatically generated when participants registered their bracelet online (see Section 5.3). Moreover, when people used Twitter to announce publically that they were attending a concert, although they were not *explicitly* expressing an opinion about the artist and his/her music, it seems very likely that for many, this was a form of implicit expression of their love for the artist. However, since it was impossible to know with certainty what the user had in mind while posting these tweets, they were not included in the Opinion Expression category.

In addition to expressing their opinion through their tweets, subjectivity also took the form of emotion expression. A small proportion of the tweets (45 or 6.3%) fell in that category. Most of the time, the emotion conveyed was excitement:

```
So excited to see @runjewels today at
#Osheaga today, it's gonna be hype!
```

```
Mikey & brian of #Weezer. #VIP I feel like a
16 year old Asian girl #OSHEAGA [accompanied
by a photo of self with friends]
```

Sometimes, the emotion was not named but it transpired from the interjections, the emojis, or the repetition of some letters in a word:

```
KENDRICKKKKKKKK #WEGONBEALRIGHT #OSHEAGA2015
[accompanied by a photo of Kendrick Lamar on
stage]
```

```
@youngthegiant 50 mins untill you guys
play!!! @youngthegiant #osheaga!
```

Sharing pure information, as in the tweet below, was not common.

```
New adult 'play' area complete with jumping
castles and swings #osheaga2015 #cbcmtl [ac-
companied by a photo of the area]
```

More often, information and opinion or emotion were combined in one tweet:

```
The charming George Ezra is playing on the
Mountain Stage🙊🙊#OSHEAGA2015 [accompanied
by a photo of the singer on stage]
```

	No. of tweets	% (n=712)
Opinion expression (all)	**153**	**21.5%**
About concerts or artists	96	13.5%
About festival	37	5.2%
About on-site activities	3	0.4%
Other	17	2.4%
Emotion expression	**45**	**6.3%**
Subjective tweets in total:	**207**	**29.1%**

Table 5. Opinion and emotion expression in tweets posted during the festival

5.5 How Do the Festival Events Influence Twitter Activity?

To identify how the activity on Twitter relates to the festival events, we looked at the number of tweets per hour during the three days of the festival, from 7 AM to 11 PM. Figure 1 shows the distribution of the tweets per hour for these days.

The program started at 1 PM to finish at 11 PM. We notice a first peak on each day at 12 PM, which certainly corresponds to the time at which people would arrive on site. Two other peaks are noted on July 31, which coincide with the beginning of shows by headliners artists of the festival. The first peak occurred at 3 PM, the time at which the Run The Jewels show started, and the second occurred at 8 PM, the time at which the show of two important artists simultaneously started: FKA Twigs and Of Monsters and Men.

On August 2, two clear bursts of activity on Twitter are observed, one at 4 PM, during the Father John Misty show, and another at 6 PM. This last peak is harder to explain. It might be due to the fact that it corresponds with the end of the The War On Drugs show and the beginning of the Hot Chip's. The activity on Twitter is more stable on August 1st, which is surprising since this was the day the most awaited show—Kendrick Lamar's—was scheduled. This concert, which started at 9:20 PM, only triggered a modest burst. A closer look at the tweets posted during this show could help better explain why it did not led to more activity on Twitter.

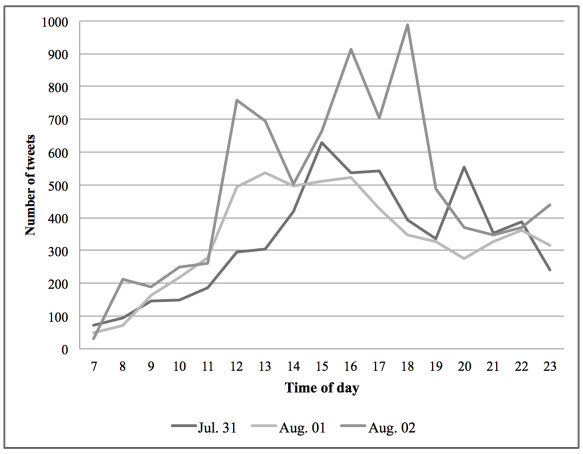


Figure 1. Number of tweets per hour during the festival

6. CONCLUSION

In this paper, we presented a study on the tweets posted during and around a major music festival, Osheaga 2015. A combination of quantitative and qualitative methods allowed us to better understand how Twitter was used by festival-goers, broadcasters, other societies, and performing artists. The analysis confirmed the results of previous studies, which revealed that Twitter [22] and other social media platforms [4] are used for taste performance or for what Papacharissi calls 'performances of the self'. Indeed, the high proportion of opinion expression tweets and the even higher number of tweets users wrote to announce that they were going to the festival or attending a specific concert suggest a desire to perform in this semi-public space. The content analysis also indicated that some users wanted the music artists they loved to take part in the conversation. Many users included the Twitter handle of the artists they were talking about in their tweets; some even spoke directly to them, even though we found little evidence that such interactions were common. This echoes the work of Litt and Hargittai [23] on the 'imagined audience' of Twitter users. In addition to the personal, communal, and professional ties people envision as their audience when posting a tweet, some people imagine 'phantasmal ties', which represent the famous people they hope to reach with their tweets and with whom they have an 'illusionary relationship'.

This study shows how rich the backchannel conversation of a music festival can be on Twitter. This conversation could provide interesting avenues for the refinement of music recommender systems. Since people use Twitter to express their opinion about music artists, this channel could be used to better understand the temporal dynamics of individuals' music tastes. Also, since Twitter allows us to follow the music reception of festival-goers in real-time, music recommender systems could potentially use hashtags of musical events to retrieve tweets that could allow them to identify music trends in a specific location.

7. REFERENCES

[1] S. Frith: *Sound Effects: Youth, Leisure, and the Politics of Rock 'n' Roll*, Pantheon Books, New York, 1981.

[2] A. C. North, and D. J. Hargreaves: "Music and Adolescent Identity," *Music Education Research,* Vol. 1, No. 1, pp. 75-92, 1999.

[3] S. Frith: "Live Music Matters," *Scottish Music Review,* Vol. 1, No. 1, 2007.

[4] H. Liu: "Social Network Profiles as Taste Performances," *Journal of Computer-Mediated Communication,* vol. 13, no. 1, pp. 252-275, 2007.

[5] The Nielsen Company: *Audience Insights Report on Music Festivals*, 2015.

[6] The Nielsen Company: *That's the Ticket: Music Fest Goers Are Ready for Summer in Canada*, 2015.

[7] F. Giglietto, and D. Selva: "Second Screen and Participation: A Content Analysis on a Full Season Dataset of Tweets," *Journal of Communication,* Vol. 64, No. 2, pp. 260-277, 2014.

[8] D. Y. Wohn, and E.-K. Na: "Tweeting about TV: Sharing Television Viewing Experiences Via Social Media Message Streams," *First Monday,* Vol. 16, No. 3, 2011.

[9] A. Bruns, and S. Stieglitz: "Towards More Systematic Twitter Analysis: Metrics for Tweeting Activities," *International Journal of Social Research Methodology,* Vol. 16, no. 2, pp. 91-108, 2013.

[10] S. Harrington, T. Highfield, and A. Bruns: "More than a Backchannel: Twitter and television," *Participations: Journal of Audience and Reception Studies,* Vol. 10, No. 1, pp. 405-409, 2013.

[11] T. Highfield, S. Harrington, and A. Bruns: "Twitter as a Technology for Audiencing and Fandom" *Information, Communication & Society,* Vol. 16, No. 3, pp. 315-339, 2013.

[12] D. Hauger: "The Million Musical Tweets Dataset: What Can We Learn from Microblogs," *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 2013.

[13] J. L. Moore, T. Joachims, and D. Turnbull: "Taste Space Versus the World: An Embedding Analysis of Listening Habits and Geography," *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pp. 439-444, 2014.

[14] K. Farrahi, M. Schedl, A. Vall, D. Hauger, and M. Tkalcic: "Impact of Listening Behavior on Music Recommendation," *15th International Society for Music Information Retrieval Conference,* pp. 484-486, 2014.

[15] E. Zangerle, M. Pichl, W. Gassler, and G. Specht: "#nowplaying Music Dataset: Extracting Listening Behavior from Twitter," *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*, pp. 21-26, 2014.

[16] Y. Kim, B. Suh, and K. Lee: "#nowplaying the Future Billboard: Mining Music Listening Behaviors of Twitter Users for Hit Song Prediction," *Proceedings of the first international workshop on Social media retrieval and analysis*, pp. 51-56, 2014.

[17] D. Iren, C. C. S. Liem, J. Yang, and A. Bozzon: "Using Social Media to Reveal Social and Collective Perspectives on Music," *Proceedings of the 8th ACM Conference on Web Science*, pp. 296-300, 2016.

[18] S. Chartier: "Dix ans dans les oreilles.," *Le Devoir*, 2015.

[19] É. Côté: "Bilan positif pour Osheaga," *La Presse*, 2016.

[20] H.-F. Hsieh, and S. E. Shannon: "Three Approaches to Qualitative Content Analysis," *Qualitative Health Research,* Vol. 15, No. 9, pp. 1277-1288, 2005.

[21] A. Marwick: "Ethnographic and Qualitative Research on Twitter," *Twitter and Society*, K. Weller, A. Bruns, C. Puschmann, J. Burgess and M. Mahrt, eds., pp. 109-122, Peter Lang, New York, 2013.

[22] Z. Papacharissi: "Without You, I'm nothing: Performances of the Self on Twitter," *International Journal of Communication,* Vol. 6, No. 2, 2012.

[23] E. Litt, and E. Hargittai: "The Imagined Audience on Social Network Sites," *Social Media + Society,* January-March, pp. 1-12, 2016.

A DATABASE LINKING PIANO AND ORCHESTRAL *MIDI* SCORES WITH APPLICATION TO AUTOMATIC PROJECTIVE ORCHESTRATION

Léopold Crestel[1] **Philippe Esling**[1]
Lena Heng[2] **Stephen McAdams**[2]
[1] Music Representations, IRCAM, Paris, France
[2] Schulich School of Music, McGill University, Montréal, Canada
leopold.crestel@ircam.fr

ABSTRACT

This article introduces the Projective Orchestral Database (*POD*), a collection of *MIDI* scores composed of pairs linking piano scores to their corresponding orchestrations. To the best of our knowledge, this is the first database of its kind, which performs piano or orchestral prediction, but more importantly which tries to learn the correlations between piano and orchestral scores. Hence, we also introduce the projective orchestration task, which consists in learning how to perform the automatic orchestration of a piano score. We show how this task can be addressed using learning methods and also provide methodological guidelines in order to properly use this database.

1. INTRODUCTION

Orchestration is the subtle art of writing musical pieces for the orchestra by combining the properties of various instruments in order to achieve a particular musical idea [11,23]. Among the variety of writing techniques for orchestra, we define as *projective orchestration* [8] the technique which consists in first writing a piano score and then orchestrating it (akin to a projection operation, as depicted in Figure 1). This technique has been used by classic composers for centuries. One such example is the orchestration by Maurice Ravel of *Pictures at an Exhibition*, a piano work written by Modest Mussorgsky. This paper introduces the first dataset of musical scores dedicated to projective orchestrations. It contains pairs of piano pieces associated with their orchestration written by famous composers. Hence, the purpose of this database is to offer a solid knowledge for studying the correlations involved in the transformation from a piano to an orchestral score.

The remainder of this paper is organized as follows. First, the motivations for a scientific investigation of orchestration are exposed (section 2). By reviewing the previous attempts, we highlight the specific need for a symbolic database of piano and corresponding orchestral scores. In an attempt to fill this gap, we built the *Projective Orchestral Database* (*POD*) and detail its structure in section 3. In section 4, the automatic projective orchestration task is proposed as an evaluation framework for automatic orchestration systems. We report our experiment with a set of learning-based models derived from the Restricted Boltzmann Machine [26] and introduce their performance in the previously defined evaluation framework. Finally, in section 5 we provide methodological guidelines and conclusions.

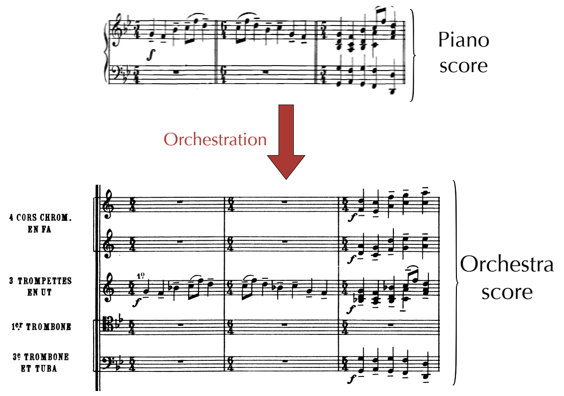


Figure 1. *Projective orchestration* of the first three bars of Modest Mussorgsky's piano piece *Pictures at an Exhibition* by Maurice Ravel. Piano notes are assigned to one or several instruments, possibly with doubling or harmonic enhancement.

2. A SCIENTIFIC INVESTIGATION OF ORCHESTRATION

Over the past centuries, several treatises have been written by renowned composers in an attempt to decipher some guiding rules in orchestration [11, 21, 23]. Even though they present a remarkable set of examples, none of them builds a systemic set of rules towards a comprehensive theory of orchestration. The reason behind this lack lies in the tremendous complexity that emerges from orchestral works. A large number of possible sounds can be created by combining the pitch and intensity ranges of each instruments in a symphonic orchestra. Furthermore, during a performance, the sound produced by a mixture of instruments is also the result of highly non-linear acoustic effects. Finally, the way we perceive those sounds involves complex psychoacoustic phenomena [14, 16, 25]. It seems almost impossible for a human mind to grasp in its entirety the intertwined mechanisms of an orchestral rendering.

Hence, we believe that a thorough scientific investigation could help disentangle the multiple factors involved in orchestral works. This could provide a first step towards a greater understanding of this complex and widely uncharted discipline. Recently, major works have refined our understanding of the perceptual and cognitive mechanisms specifically involved when listening to instrumental mixtures [15, 22, 25]. Orchids, an advanced tool for assisting composers in the search of a particular sonic goal has been developed [8]. It relies on the multi-objective optimization of several spectro-temporal features such as those described in [20].

However, few attempts have been made to tackle a scientific exploration of orchestration based on the study of musical scores. Yet, symbolic representations implicitly convey high-level information about the spectral knowledge composers have exploited for timbre manipulations. In [6] a generative system for orchestral music is introduced. Given a certain style, the system is able to generate a melodic line and its accompaniment by a full symphonic orchestra. Their approach relies on a set of templates and hand-designed rules characteristic of different styles. [19] is a case study of how to automatically transfer the *Ode to joy* to different styles. Unfortunately, very few details are provided about the models used, but it is interesting to observe that different models are used for different styles. Automatic arrangement, which consists in reducing an orchestral score to a piano version that is can be played by a two-hand pianist, has been tackled in [10] and [24]. The proposed systems rely on an automatic analysis of the orchestral score in order to split it into structuring elements. Then, each element is assigned a role which determines whether it is played or discarded in the reduction. To the best of our knowledge, the inverse problem of automatically orchestrating a piano score has never been tackled. However, we believe that unknown mechanisms of orchestration could be revealed by observing how composers perform projective orchestration, which essentially consists in highlighting an existing harmonic, rhythmic and melodic structure of a piano piece through a timbral structure.

Even though symbolic data are generally regarded as a more compact representation than a raw signal in the computer music field, the number of pitch combinations that a symphonic orchestra can produce is extremely large. Hence, the manipulation of symbolic data still remains costly from a computational point of view. Even through computer analysis, an exhaustive investigation of all the possible combinations is not feasible. For that reason, the approaches found in the literature rely heavily on heuristics and hand-designed rules to limit the number of possible solutions and decrease the complexity. However, the recent advents in machine learning have brought techniques that can cope with the dimensionality involved with symbolic orchestral data. Besides, even if a wide range of orchestrations exist for a given piano score, all of them will share strong relations with the original piano score. Therefore, we make the assumption that projective orchestration might be a relatively simple and well-structured transformation lying in a complex high-dimensional space. Neural networks have precisely demonstrated a spectacular ability for extracting a structured lower-dimensional manifold from a high-dimensional entangled representation [13]. Hence, we believe that statistical tools are now powerful enough to lead a scientific investigation of projective orchestration based on symbolic data.

These statistical methods require an extensive amount of data, but there is no symbolic database dedicated to orchestration. This dataset is a first attempt to fill this gap by building a freely accessible symbolic database of piano scores and corresponding orchestrations.

3. DATASET

3.1 Structure of the Database

The database can be found on the companion website [1] of this article, along with statistics and Python code for reproducibility.

3.1.1 Organization

The Projective Orchestral Database (*POD*) contains 392 *MIDI* files. Those files are grouped in pairs containing a piano score and its orchestral version. Each pair is stored in a folder indexed by a number. The files have been collected from several free-access databases [1] or created by professional orchestration teachers.

3.1.2 Instrumentation

As the files gathered in the database have various origins, different instrument names were found under a variety of aliases and abbreviations. Hence, we provide a comma-separated value (*CSV*) file associated with each *MIDI* file in order to normalize the corresponding instrumentations. In these files, the track names of the *MIDI* files are linked to a normalized instrument name.

3.1.3 Metadata

For each folder, a CSV file with the name of the folder contains the relative path from the database root directory, the composer name and the piece name for the orchestral and piano works. A list of the composers present in the database can be found in table 1. It is important to note the imbalanced representativeness of composers in the database. It can be problematic in the learning context we investigate, because a kind of stylistic consistency is *a priori* necessary in order to extract a coherent set of rules. Picking a subset of the database would be one solution, but another possibility would be to add to the database this stylistic information and use it in a learning system.

[1] `https://qsdfo.github.io/LOP/database`

Composer	Number of piano files	Percentage piano frames	Number of orchestra files	Percentage orchestra frames
Arcadelt. Jacob			1	0.07
Arresti. Floriano	3	0.57		
Bach. Anna Magdalena	3	0.43		
Bach. Johann Sebastian	9	4.57	4	0.81
Banchieri. Adriano			1	0.32
Beethoven. Ludwig Van	1	0.60	38	42.28
Berlioz. Hector			1	0.14
Brahms. Johannes			3	0.28
Buxtehude. Dietrich	1	0.21		
Byrd. William	1	0.13		
Charpentier. Marc-Antoine			2	0.38
Chopin. Frederic			2	0.44
Clarke. Jeremiah			1	0.23
Debussy. Claude	1	0.59	6	0.90
Dvorak. Anton			6	2.42
Erlebach. Philipp Heinrich			1	0.10
Faure. Gabriel			1	0.60
Fischer. Johann Caspar Ferdinand	1	0.10		
Gluck. Christoph Willibald			1	1.61
Grieg. Edvard			1	2.10
Guerrero. Francisco	1	0.12		
Handel. George Frideric	4	1.00	1	0.75
Haydn. Joseph			6	1.01
Kempff. Wilhelm	1	1.58		
Leontovych. Mykola			2	0.22
Liszt. Franz	34	39.98		
Mahler. Gustav			1	0.85
Mendelssohn. Felix			2	1.41
Moussorgsky. Modest			1	0.04
Mozart. Wolfgang Amadeus	1	0.71	8	1.45
Okashiro. Chitose	3	1.09		
Pachelbel. Johann	1	0.15		
Praetorius. Michael			2	0.14
Purcell. Henry			1	0.08
Ravel. Maurice	6	6.49	8	6.69
Rondeau. Michel	2	0.25	1	0.14
Schonberg. Arnold			1	0.21
Schumann. Robert			1	0.05
Shorter. Steve	1	0.26		
Smetana. Bedrich			1	0.61
Soler. Antonio	1	0.54		
Strauss. Johann			1	0.04
Strauss. Richard			1	0.22
Stravinsky. Igor			4	0.94
Tchaikovsky. Piotr Ilyich			36	20.08
Telemann. Georg Philipp			2	1.04
Unknown.	107	40.18	28	7.47
Vivaldi. Antonio			4	2.94
Walther. Johann Gottfried	1	0.14		
Wiberg. Steve			1	0.75
Zachow. Friedrich Wilhelm	1	0.32	2	0.23

Table 1. This table describes the relative importance of the different composers present in the database. For each composer, the number of piano (respectively orchestral) scores in the database are indicated in the second (respectively fourth) column. The total number of files is 184 x 2 = 392. As the length of the files can vary significantly, a more significant indicator of a composer's representativeness in the database is the ratio of the number of frames from its scores over the total number of frames in the database.

Figure 2 highlights the activation ratio of each pitch in the orchestration scores ($\frac{\#\{\text{pitch on}\}}{\#\{\text{pitch on}\}+\#\{\text{pitch off}\}}$, where $\#$ is the cardinal of an ensemble) over the whole dataset. Note that this activation ratio does not take the duration of notes into consideration, but only their number of occurrences. The pitch range of each instrument can be observed beneath the horizontal axis.

Two different kinds of imbalance can be observed in figure 2. First, a given pitch is rarely played. Second, some pitches are played more often compared with others. Class imbalance is known as being problematic for machine learning systems, and these two observations highlight how challenging the projective orchestration task is.

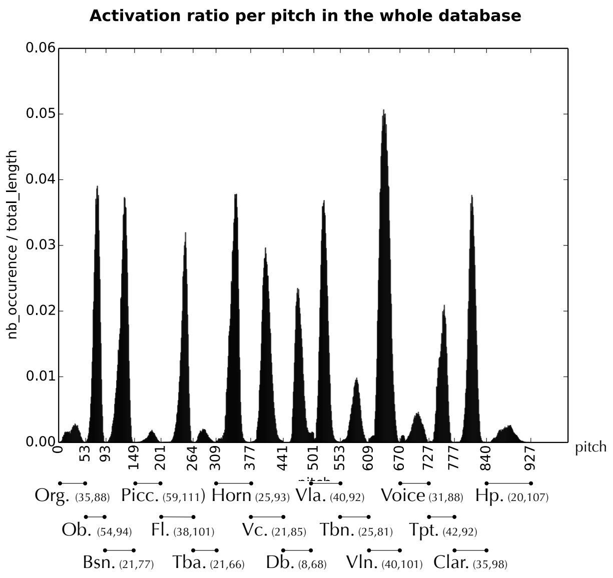


Figure 2. Activation ratio per pitch in the whole orchestral score database. For one bin on the horizontal axis, the height of the bar represents the number of notes played by this instrument divided by the total number of frames in the database. This value is computed for the event-level aligned representations 4.2. The different instruments are covered by the pitch axis, and one can observe the peaks that their medium ranges form. The maximum value of the vertical axis (0.06), which is well below 1, indicates that each pitch is rarely played in the whole database.

More statistics about the whole database can be found on the companion website.

3.1.4 Integrity

Both the metadata and instrumentation *CSV* files have been automatically generated but manually checked. We followed a conservative approach by automatically rejecting any score with the slightest ambiguity between a track name and a possible instrument (for instance *bass* can refer to *double-bass* or *voice bass*).

3.1.5 Formats

To facilitate the research work, we provide pre-computed piano-roll representations such as the one displayed in Figure 3. In this case, all the *MIDI* files of piano (respectively orchestra) work have been transformed and concatenated into a unique two-dimensional matrix. The starting and ending time of each track is indicated in the *metadata.pkl* file. These matrices can be found in Lua/Torch (.t7), Matlab (.m), Python (.npy) and raw (.csv) data formats.

3.1.6 Score Alignment

Two versions of the database are provided. The first version contains unmodified midi files. The second version contains *MIDI* files automatically aligned using the *Needleman-Wunsch* [18] algorithm as detailed in

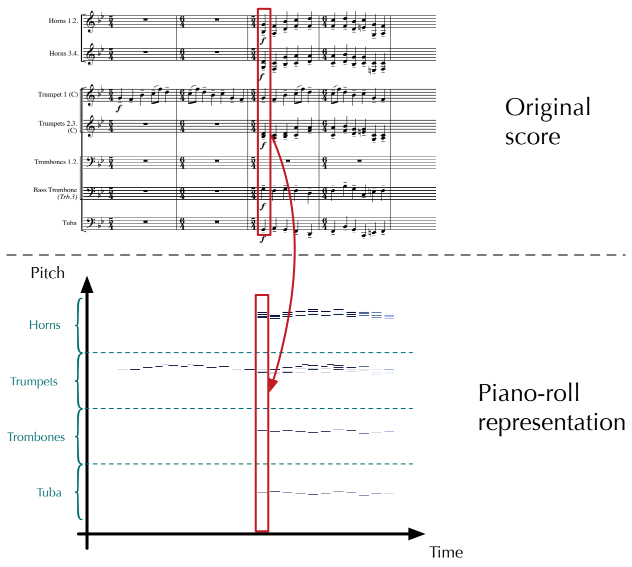


Figure 3. *Piano-roll* representation of orchestral scores. The *piano-roll* pr is a matrix. A pitch p at time t played with an intensity i is represented by $pr(p,t) = i$, where 0 is a note off. This definition is extended to an orchestra by simply concatenating the *piano-rolls* of every instrument along the pitch dimension.

Section 3.2.

3.2 Automatic Alignment

Given the diverse origins of the *MIDI* files, a piano score and its corresponding orchestration are almost never aligned temporally. These misalignments are very problematic for learning or mining tasks, and in general for any processing which intends to take advantage of the joint information provided by the piano and orchestral scores. Hence, we propose a method to automatically align two scores, and released its Python implementation on the companion website [2]. More precisely, we consider the piano-roll representations (Figure 3) where the scores are represented as a sequence of vectors. By defining a distance between two vectors, the problem of aligning two scores can be cast as a univariate sequence-alignment problem.

3.2.1 Needleman-Wunsch

The *Needleman-Wunsch* (*NW*) algorithm [18] is a dynamic programming technique, which finds the optimal alignment between two symbolic sequences by allowing the introduction of gaps (empty spaces) in the sequences. An application of the *NW* algorithm to the automatic alignment of musical performances is introduced in [9]. As pointed out in that article, *NW* is the most adapted technique for aligning two sequences with important structural differences like skipped parts, for instance.

The application of the *NW* algorithm relies solely on the definition of a cost function, which allows the pairwise comparison of elements from the two sequences, and the cost of opening or extending a gap in one of the two sequences.

3.2.2 Similarity Function

To measure the similarity between two chords, we propose the following process:

- discard intensities by representing notes being played as one and zero otherwise.
- compute the pitch-class representation of the two vectors, which flattens all notes to a single octave vector (12 notes). In our case, we set the pitch-class to one if at least one note of the class is played. For instance, we set the pitch-class of C to one if there is any note with pitch C played in the piano-roll vector. This provides an extremely rough approximation of the harmony, which proved to be sufficient for aligning two scores. After this step, the dimensions of each vector is 12.
- if one of the vectors is only filled with zeros, it represents a silence, and the similarity is automatically set to zero (note that the score function can take negative values).
- for two pitch-class vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, we define the score as

$$S(\boldsymbol{A},\boldsymbol{B}) = C \times \frac{\sum_{i=1}^{12} \delta(A_i + B_i)}{max(||\boldsymbol{A}+\boldsymbol{B}||_1, 1)} \qquad (1)$$

where δ is defined as:

$$\delta(x) = \begin{cases} 0 & \text{if x = 0} \\ -1 & \text{if x = 1} \\ 1 & \text{if x = 2} \end{cases}$$

C is a tunable parameter and $||x||_1 = \sum_i |x_i|$ is the $\mathcal{L}1$ norm.

Based on the values recommended in [18] and our own experimentations, we set C to 10. The gap-open parameter, which defines the cost of introducing a gap in one of the two sequences, is set to 3 and the gap-extend parameter, which defines the cost of extending a gap in one of the two sequences, is set to 1.

4. AN APPLICATION : PROJECTIVE AUTOMATIC ORCHESTRATION

In this section, we introduce and formalize the automatic projective orchestration task (Figure 1). In particular, we propose a system based on statistical learning and define an evaluation framework for using the *POD* database.

4.1 Task Definition

4.1.1 Orchestral Inference

For each orchestral piece, we define as **O** and **P** the aligned sequences of column vectors from the *piano-roll* of the orchestra and piano parts. We denote as T the length of the aligned sequences **O** and **P**.

[2] `https://qsdfo.github.io/LOP/code`

The objective of this task is to infer the present orchestral frame knowing both the recent past of the orchestra sequence and the present piano frame. Mathematically, it consists in designing a function f where

$$\hat{O}(t) = \boldsymbol{f}[P(t), O(t-1), ..., O(t-N)] \quad \forall t \in [N, ...T] \tag{2}$$

and N defines the order of the model.

4.1.2 Evaluation Framework

We propose a quantitative evaluation framework based on a one-step predictive task. As discussed in [5], we make the assumption that an accurate predictive model will be able to generate original acceptable works. Whereas evaluating the generation of a complete musical score is subjective and difficult to quantify, a predictive framework provides us with a quantitative evaluation of the performance of a model. Indeed, many satisfying orchestrations can be created from the same piano score. However, the number of reasonable inferences of an orchestral frame given its context (as described in equation 2) is much more limited.

As suggested in [4, 12], the accuracy measure [2] can be used to compare an inferred frame $\hat{O}(t)$ drawn from (2) to the ground-truth $O(t)$ from the original file.

$$\text{Accuracy}(t) = 100 \, . \, \frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{3}$$

where $TP(t)$ (true positives) is the number of notes correctly predicted (note played in both $\hat{O}(t)$ and $O(t)$). $FP(t)$ (false positive) is the number of notes predicted that are not in the original sequence (note played in $\hat{O}(t)$ but not in $O(t)$). $FN(t)$ (false negative) is the number of unreported notes (note absent in $\hat{O}(t)$, but played in $O(t)$).

When the quantization gets finer, we observed that a model which simply repeats the previous frame gradually obtains the best accuracy as displayed in Table 2. To correct this bias, we recommend using an event-level evaluation framework where the comparisons between the ground truth and the model's output is only performed for time indices in T_e defined as the set of indexes t_e such that

$$\text{O}(t_e) \neq \text{O}(t_e - 1)$$

The definition of event-level indices can be observed in Figure 4.

In the context of learning algorithms, splitting the database between disjoint *train* and *test* subsets is highly recommended [3, pg.32-33], and the performance of a given model is only assessed on the test subset. Finally, the mean accuracy measure over the dataset is given by

$$\frac{1}{K} \sum_{s \in \mathcal{D}_{test}} \sum_{t_e \in T_e(s)} Accuracy(t_e) \tag{4}$$

where $\mathcal{D}_{test}$ defines the test subset, $T_e(s)$ the set of event-time indexes for a given score s, and $K = \sum_{s \in \mathcal{D}_{test}} |T_e(s)|$.

4.2 Proposed Model

In this section, we propose a learning-based approach to tackle the automatic orchestral inference task.

4.2.1 Models

We present the results for two models called *conditional Restricted Boltzmann Machine* (*cRBM*) and *Factored Gated cRBM* (*FGcRBM*). The models we explored are defined in a probabilistic framework, where the vectors $O(t)$ and $P(t)$ are represented as binary random variables. The orchestral inference function is a neural network that expresses the conditional dependencies between the different variables: the present orchestral frame $O(t)$, the present piano frame $P(t)$ and the past orchestral frames $O(t-1, ..., t-N)$. *Hidden units* are introduced to model the co-activation of these variables. Their number is a hyper-parameter with an order of magnitude of 1000. A theoretical introduction to these models can be found in [26], whereas their application to projective orchestration is detailed in [7].

4.2.2 Data Representation

In order to process the scores, we import them as *piano-roll* matrices (see Figure 3). Their extension to orchestral scores is obtained by concatenating the *piano-rolls* of each instrument along the pitch dimension.

Then, new events $t_e \in T_e$ are extracted from both piano-rolls as described in Section 4.1. A consequence is that the trained model apprehends the scores as a succession of events with no rhythmic structure. This is a simplification that considers the rhythmic structure of the projected orchestral score to be exactly the same as the one of the original piano score. This is false in the general case, since a composer can decide to add nonexistent events in an orchestration. However, this provides a reasonable approximation that is verified in a vast majority of cases. During the generation of an orchestral score given a piano score, the next orchestral frame is predicted in the event-level framework, but inserted at the temporal location of the corresponding piano frame as depicted in Figure 4.

Automatic alignment of the two *piano-rolls* is performed on the event-level representations, as described in Section 3.2.

In order to reduce the input dimensionality, we systematically remove any pitch which is never played in the training database for each instrument. With that simplification the dimension of the orchestral vector typically decreases from 3584 to 795 and the piano vector dimension from 128 to 89. Also, we follow the usual orchestral simplifications used when writing orchestral scores by grouping together all the instruments of a same section. For instance, the *violin* section, which might be composed by several instrumentalists, is written as a single part. Finally, the velocity information is discarded, since we use binary units that solely indicate if a note is on or off.

Eventually, we observed that an important proportion of the frames are silences, which mathematically corresponds to a column vector filled with zeros in the piano-roll representation. A consequence of the over-representation of silences is that a model trained on this database will lean towards orchestrating with a silence any piano input, which is statistically the most relevant choice. Therefore, orches-

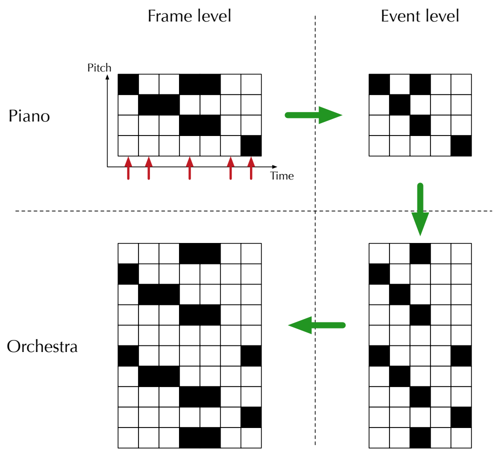


Figure 4. From a piano score, the generation of an orchestral score consists in extracting the event-level representation of the piano score, generating the sequence of orchestral events, and then injecting them at the position of the event from the piano score. Note that the silence in the fourth event of the piano score is not orchestrated by the probabilistic model, but is automatically mapped to a silence in the orchestral version.

tration of silences in the piano score ($P(t) = 0$) are not used as training points. However, it is important to note that they are not removed from the piano-rolls. Hence, silences could still appear in the past sequence of a training point, since it is a valuable information regarding the structure of the piece. During generation time, the silences in the piano score are automatically orchestrated with a silence in the orchestra score. Besides, silences are taken into consideration when computing the accuracy.

4.2.3 Results

The results of the *cRBM* and *FGcRBM* on the orchestral inference task are compared to two naïve models. The first model is a random generation of the orchestral frames obtained by sampling a Bernoulli distribution of parameter 0.5. The second model predicts an orchestral frame at time t by simply repeating the frame at time $t - 1$. The results are summed up in Table 2.

Model	Frame-level accuracy (Q = 4)	Frame-level accuracy (Q = 8)	Event-level accuracy
Random	0.73	0.73	0.72
Repeat	61.79	76.41	50.70
cRBM	5.12	34.25	27.67
FGcRBM	33.86	43.52	25.80

Table 2. Results of the different models for the projective orchestration task based on frame-level accuracies with a quantization of 4 and 8 and event-level accuracies.

4.3 Discussion

As expected, the random model obtains very poor results. The repeat model outperform all three other models, surprisingly even in the event-level framework. Indeed, we observed that repeated notes still occur frequently in the event-level framework. For instance, if between two successive events only one note out of five is modified, the accuracy of the repeat model on this frame will be equal to 66%.

If the *FGcRBM* model outperforms the *cRBM* model in the frame-level framework, the *cRBM* is slightly better than the *FGcRBM* model in the event-level framework.

Generations from both models can be listened to on the companion website [3]. Even though some fragments are coherent regarding the piano score and the recent past orchestration, the results are mostly unsatisfying. Indeed, we observed that the models learn an extremely high probability for every note to be off. Using regularization methods such as weight decay has not proven efficient. We believe that this is due to the sparsity of the vectors $O(t)$ we try to generate, and finding a more adapted data representation of the input will be a crucial step.

5. CONCLUSION AND FUTURE WORK

We introduced the Projective Orchestral Database (*POD*), a collection of *MIDI* files dedicated to the study of the relations between piano scores and corresponding orchestrations. We believe that the recent advent in machine learning and data mining has provided the proper tools to take advantage of this important mass of information and investigate the correlations between a piano score and its orchestrations. We provide all *MIDI* files freely, along with aligned and non-aligned pre-processed piano-roll representations on the website `https://qsdfo.github.io/LOP/index.html`.

We proposed a task called automatic orchestral inference. Given a piano score and a corresponding orchestration, it consists in trying to predict orchestral time frames, knowing the corresponding piano frame and the recent past of the orchestra. Then, we introduced an evaluation framework for this task based on a train and test split of the database, and the definition of an accuracy measure. We finally present the results of two models (the *cRBM* and *FGcRBM*) in this framework.

We hope that the *POD* will be useful for many researchers. Besides the projective orchestration task we defined in this article, the database can be used in several other applications, such as generating data for a source-separation model [17]. Even if small errors still persist, we thoroughly checked manually the database and guarantee its good quality. However, the number of files collected is still small with the aim of leading statistical investigations. Hence, we also hope that people will contribute to enlarge this database by sharing files and helping us gather the missing information.

[3] `https://qsdfo.github.io/LOP/results`

6. REFERENCES

[1] Imslp. `http://imslp.org/wiki/Main_Page`. Accessed : 2017-01-23.

[2] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *ISMIR*, pages 315–320, 2009.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

[5] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[6] J. Cookerly. Complete orchestration system, May 18 2010. US Patent 7,718,883.

[7] Leopold Crestel and Philippe Esling. Live orchestral piano, a system for real-time orchestral music generation. In *Proceedings of the 14th Sound and Music Computing Conference*, Aalto, Finland, July 2017.

[8] Philippe Esling, Grégoire Carpentier, and Carlos Agon. Dynamic musical orchestration using genetic algorithms and a spectro-temporal description of musical instruments. *Applications of Evolutionary Computation*, pages 371–380, 2010.

[9] Maarten Grachten, Martin Gasser, Andreas Arzt, and Gerhard Widmer. Automatic alignment of music performances with structural differences. In *In Proceedings of 14th International Society for Music Information Retrieval Conference (ISMIR*. Citeseer, 2013.

[10] Jiun-Long Huang, Shih-Chuan Chiu, and Man-Kwan Shan. Towards an automatic music arrangement framework using score reduction. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(1):8, 2012.

[11] Charles Koechlin. *Traité de l'orchestration*. Éditions Max Eschig, 1941.

[12] Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 120–129. ACM, 2003.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 05 2015.

[14] Sven-Amin Lembke and Stephen McAdams. Timbre blending of wind instruments: acoustics and perception. 2012.

[15] Stephen McAdams. Timbre as a structuring force in music. In *Proceedings of Meetings on Acoustics*, volume 19, page 035050. Acoustical Society of America, 2013.

[16] Stephen McAdams and Bruno L Giordano. The perception of musical timbre. *The Oxford handbook of music psychology*, pages 72–80, 2009.

[17] M. Miron, J. Janer, and E. Gómez. Generating data to train convolutional neural networks for classical music source separation. In *Proceedings of the 14th Sound and Music Computing Conference*, pages 227–233, Aalto, Finland, 2017 2017.

[18] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970.

[19] François Pachet. A joyful ode to automatic orchestration. *ACM Trans. Intell. Syst. Technol.*, 8(2):18:1–18:13, October 2016.

[20] Geoffroy Peeters, Bruno L Giordano, Patrick Susini, Nicolas Misdariis, and Stephen McAdams. The timbre toolbox: Extracting audio descriptors from musical signals. *The Journal of the Acoustical Society of America*, 130(5):2902–2916, 2011.

[21] Walter Piston. *Orchestration*. New York: Norton, 1955.

[22] Daniel Pressnitzer, Stephen McAdams, Suzanne Winsberg, and Joshua Fineberg. Perception of musical tension for nontonal orchestral timbres and its relation to psychoacoustic roughness. *Perception & psychophysics*, 62(1):66–80, 2000.

[23] Nikolay Rimsky-Korsakov. *Principles of Orchestration*. Russischer Musikverlag, 1873.

[24] Hirofumi Takamori, Haruki Sato, Takayuki Nakatsuka, and Shigeo Morishima. Automatic arranging musical score for piano using important musical elements. In *Proceedings of the 14th Sound and Music Computing Conference*, Aalto, Finland, July 2017.

[25] Damien Tardieu and Stephen McAdams. Perception of dyads of impulsive and sustained instrument sounds. *Music Perception*, 30(2):117–128, 2012.

[26] Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning*, pages 1025–1032. ACM, 2009.

ANALYSIS OF INTERACTIVE INTONATION IN UNACCOMPANIED SATB ENSEMBLES

Jiajie Dai, Simon Dixon
Centre for Digital Music, Queen Mary University of London, United Kingdom
{j.dai, s.e.dixon}@qmul.ac.uk

ABSTRACT

Unaccompanied ensemble singing is common in many musical cultures, yet it requires great skill for singers to listen to each other and adjust their pitch to stay in tune. The aim of this research is to investigate interaction in four-part (SATB) singing from the point of view of pitch accuracy (intonation). In particular we compare intonation accuracy of individual singers and collaborative ensembles. 20 participants (five groups of four) sang two pieces of music in three different listening conditions: solo, with one vocal part missing and with all vocal parts. After semi-automatic pitch extraction and manual correction, we annotated the recordings and calculated the pitch error, melodic interval error, harmonic interval error and note stability. We observed significant differences between individual and interactional intonation, more specifically: 1) Singing without the bass part has less mean absolute pitch error than singing with all vocal parts; 2) Mean absolute melodic interval error increases when participants can hear the other parts; 3) Mean absolute harmonic interval error is higher in the one-way interaction condition than the two-way interaction condition; and 4) Singers produce more stable notes when singing solo than with their partners.

1. INTRODUCTION AND BACKGROUND

Voice is our original instrument [8], even from prehistoric times [13], and it is one of the defining features of humanity [26]. This instrument communicates emotion, expressing joy and sadness, hope and despair. Throughout the history of vocal performance, various theories have been set forth on vocal aesthetics and intonation in both individual and ensemble settings. This paper investigates the influence of interaction between singers on the intonation of singing ensembles.

Intonation describes how a pitch is played or sung in tune [7]. Its extreme importance in Western music arises from the fact that it relates to both melody and harmony, two central aspects of tonal music. The accuracy of intonation is determined by culturally specific tuning systems such as the equal tempered tuning system in Western music [25].

Without interaction or accompaniment, it is extremely difficult to sing with accurate pitch. Only 0.01% of people have *absolute pitch* [22], which is the ability to identify or reproduce any given note on demand [2]. Others must rely on relative pitch for tuning, comparing current auditory feedback with the memory of recently heard tones. As this memory fades, singers may sing out of tune or exhibit pitch drift, where intonation moves away from the reference pitch during a performance [9, 12, 20]. Singers also use their muscle memory, a learnt relationship between muscle strength and pitch, to tune their pitch [1].

Although the intonation of singers in individual and group settings has been investigated, very little of this research addresses interaction between singers in vocal ensembles. In Western music, one common configuration for singing ensembles and choirs comprises four musical voices or parts: soprano, alto, tenor and bass (SATB); so we chose the SATB ensemble as the research target for this paper.

Music ensembles are well-characterised examples of interactive work groups [28]. Every member of a musical ensemble needs to execute his or her own part flawlessly as well as contribute to the overall performance in a manner that produces a cohesive, unified sound [3]. This means that individual singers have to stay in tune with their own part (their previous notes) and with other singers' parts (concurrent and previous notes) [18, p. 151]. This creates a practical difficulty for SATB singers, because they have multiple potentially conflicting reference pitches, as well as their own tonal reference, on which they could base their relative pitch, and attending to any specific one of these may be difficult.

Interaction plays an important role in ensemble performance, but its effects can be negative. Terasawa and Hiroko [23] claimed that the intonation accuracy of choral members was influenced by the progression of chord roots. Brandler and Peynircioglu [3] observed that participants learned new pieces of music more efficiently when learning it individually than with companions. Mürbe *et al.* [15] observed that singers' intonation accuracy is reduced in the absence of auditory feedback. When singers cannot hear themselves, they have to rely on their muscle memory to tune which leads to an inaccurate intonation. Dai and Dixon [4] noted that even the presence of an in-tune stimulus during singing reduced singers' accuracy.

Although many publications give guidelines to keep singers in tune by training them as excellent soloists [1,2], the interaction in SATB ensemble performance as it unfolds in real-time has not been fully researched. The target of this study is to test the influence of the various vocal parts and how the singers interact with each other, especially how hearing other singers influences the performance of each vocal part. These effects are tested in terms of their effect on intonation.

In the next section, we describe the research questions, hypotheses and experimental design. The methodology section follows, covering musical materials, experimental procedure and intonation metrics. Then in section 4 we present results in terms of pitch error, melodic interval error, harmonic interval error and note variability in different experimental conditions. This is followed by a discussion in section 5, and a conclusion in section 6. The recordings, annotated data and software are made freely available for research; details are given in section 8.

2. EXPERIMENTAL DESIGN

2.1 Research Questions and Hypotheses

This study of interactive intonation in unaccompanied SATB singing is driven by a number of research questions. Firstly, we wish to determine whether singers rely on a particular vocal part for intonation, which we test by systematically isolating each vocalist so that the other singers cannot hear them. We expect that the bass part, which often contains the root notes of chords, is more important as a tonal reference [23], leading to our first hypothesis: pitch error will be higher when the bass part is missing than when other voices are isolated.

The second research question involves the effect of hearing other voices on intonation. Previous work suggests that singers are distracted by simultaneous sounds when they are singing (see section 1), and they are less able to attend to their auditory feedback loop in order to sing accurately. This leads to hypothesis 2, that the conditions in which singers hear no other voice will have less melodic interval error than the conditions in which they hear other singers. This effect might be strengthened by conscious adjustment of singers to the other parts in order to improve the harmonic intervals. Thus as a corollary we frame our third hypothesis, that we expect to see less harmonic interval error when singers can hear each other than when they are isolated. An additional effect of interaction should be that singers adjust their pitch more during notes where they hear other singers (who might also be adjusting). Thus our fourth hypothesis is that within-note variability in pitch will be higher (note stability will be lower) when singers hear each other than when they do not.

2.2 Design

To test these hypotheses, a novel experiment was designed and implemented, by which we investigate the interaction between the four vocal parts. We define three different listening conditions, based on what the singer can hear as they sing. In the *closed condition*, the singer hears no other voice than their own, thus they are effectively singing solo. In the *partially-open condition* (or *partial condition* for short), the singer can only hear some, but not all of the other vocal parts. This is achieved by isolating one singer from the other three, and allowing acoustic feedback (via microphones and loudspeakers) in one direction only, either from the isolated singer to the other three singers (*one-to-three condition*), or from the three singers to the isolated one (*three-to-one condition*). Finally, in the *open condition*, all singers can hear each other.

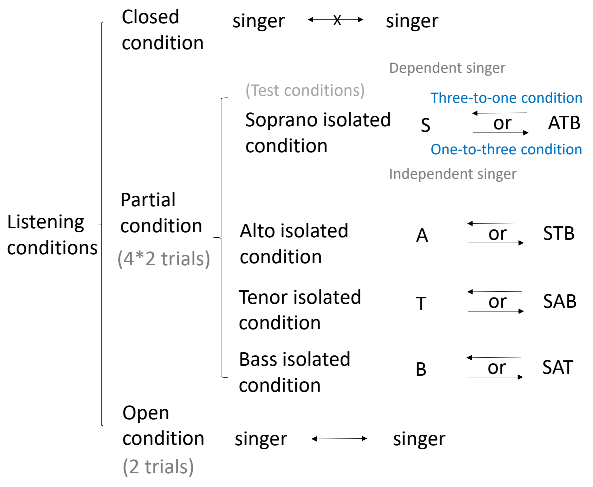


Figure 1: Listening and test conditions. The arrows indicate the direction of acoustic feedback.

For testing the partial condition, there are four pairs of test conditions corresponding to the vocal part that is isolated and the direction of feedback. For example, one test condition is called the *soprano isolated one-to-three condition*, where the soprano sings in a closed condition, but all other parts hear each other (the soprano's voice being provided to the others via a loudspeaker). In such a case the isolated singer is called the *independent singer* as they are not able to react to the other vocal parts to choose their tuning. In other cases the singer can hear all (open condition) or some (partial condition) of the other voices, and thus is called a *dependent singer*. Figure 1 gives an overview of the listening and test conditions.

3. EXPERIMENTAL METHODS

3.1 Participants

20 adult amateur singers (10 male and 10 female) with choir experience volunteered to take part in the study. The age range was from 20 to 55 years old (mean: 27.95, median: 26.50, std.dev.: 7.84). Participants were compensated £10 for their participation. The participants were able to sing their parts comfortably and they were given the score and sample audio files at least 2 weeks before the experiment. They came from the music society and *a capella* society of the university and a local choir.

Training is a crucial factor for intonation accuracy. For testing the effect of training, all the participants were given a questionnaire based on the Goldsmiths Musical Sophistication Index [14]. The participants had an average of 3.3 years of music lessons and 5.8 years of singing experience.

3.2 Materials

Two contrasting musical pieces were selected for this study: a Bach chorale, "Oh Thou, of God the Father" (BWV 164/6) and Leo Mathisen's jazz song "To be or not to be". Both pieces were chosen for their wide range of harmonic intervals (see section 3.5.2): the first piece has 34 different harmonic intervals between parts and the second piece has 30 harmonic intervals. To control the duration of the experiment, we shortened the original score by deleting the repeat. We also reduced the tempo from that specified in the score, in order to make the pieces easier to sing and compensate for the limited time that the singers had to learn the pieces. The resulting duration of the first piece is 76 seconds and the second song is 100 seconds. Links to the score and training materials can be found in section 8.

The equipment included an SSL MADI-AX converter, five cardioid microphones and four loudspeakers. All the tracks were controlled and recorded by the software Logic Pro 10. The metronome and the four starting reference pitches were also given by Logic Pro. The total latency of the system is 4.9 ms (3.3 ms due to hardware and 1.6 ms from the software).

3.3 Procedure

A pilot experiment with singers not involved in the study was performed to test the experimental setup and minimise potential problems such as bleed between microphones. Then the participants in the study were distributed into 5 groups according to their voice type, time availability and collaborative experience (the singers from the same music society were placed in the same group). Each group contained two female singers (soprano and alto) and two male singers (tenor and bass). Each participant had at least two hours practice before the recording, sometimes on separate days. They were informed about the goal of the study, to investigate interactive intonation in SATB singing, and they were asked to sing their best in all circumstances.

For each trial, the singers were played their starting notes before commencing the trial, and a metronome accompanied the singing to ensure that the same tempo was used by all groups. Each piece was sung 10 times by each group. The first and the last trial were recorded in the open condition. The partial and closed condition trials, consisting of 8 test conditions, 4 (isolated voice) $\times$ 2 (direction of feedback), were recorded in between. The order of isolated conditions was randomly chosen to control for any learning effect. For each isolated condition, the three-to-one condition always preceded the one-to-three condition. We use the performance of isolated singers in the one-to-three conditions as the data for the closed condition.

The singers were recorded in two acoustically isolated rooms. For the partial and closed conditions, the isolated singers were recorded in a separate room from the other three singers. Loudspeakers in each room provided acoustic feedback according to the test condition. There was no visual contact between singers in different rooms. With the exception of warm-up and rehearsal, but including all the trials and the questionnaire, the total duration of the experiment for each group was about one hour and a half.

3.4 Annotation

The experimental data comprises 5 (groups) $\times$ 4 (singers) $\times$ 2 (pieces) $\times$ 10 (trials) = 400 audio files, each containing 65 to 116 notes. The software *Tony* [10] was chosen as the annotation tool. *Tony* performs pitch detection using the PYIN algorithm, which outperforms the YIN algorithm [11], and then automatically segments pitch trajectories into note objects, and provides a convenient interface for manual checking and correction of the resulting annotations. For each audio file, we exported two `.csv` files, one containing the note-level information (for calculating pitch and interval errors) and the other containing the pitch trajectories (for calculating pitch variability). All the intonations were measured by twelve-tone equal temperament, expressed in semitones according to MIDI standard pitch numbering. It took about 67 hours to manually check and correct the 400 files, resulting in 49200 annotated single notes, to which we added information on the singer (anonymised), score notes and metrics of accuracy.

3.5 Intonation Metrics

To quantify the effects of interaction on intonation, we measure pitch accuracy in terms of pitch error, melodic interval error, harmonic interval error and note stability, defined below.

3.5.1 Pitch Error

Assuming that a reference pitch has been given, *pitch error* can be defined as the difference between observed pitch and score pitch [12]:

$$e_i^p = \bar{p}_i - p_i^s \tag{1}$$

where $\bar{p}_i$ is the median of the observed pitch trajectory of note i (calculated over the duration of an individual note), and p_i^s is the score pitch of note i.

To evaluate the pitch accuracy of a sung part, we use *mean absolute pitch error* (MAPE) as the measurement. For a group of M notes with pitch errors $e_1^p, \ldots, e_M^p$, the MAPE is defined as:

$$\text{MAPE} = \frac{1}{M}\sum_{i=1}^{M} |e_i^p| \tag{2}$$

3.5.2 Melodic and Harmonic Interval Error

A musical *interval* is the difference between two pitches [19], which is proportional to the logarithm of the ratio of the fundamental frequencies of the two pitches. We distinguish two types of interval in this experiment: in a *melodic interval*, the two notes are sounded in succession; while in a *harmonic interval*, both notes are played simultaneously (Figure 2).

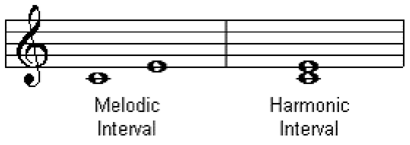


Figure 2: A melodic interval and harmonic interval of a major third (four semitones).

We thus calculate the melodic interval error as the difference between the observed and score intervals:

$$e_i^m = (\bar{p}_{i+1} - \bar{p}_i) - (p_{i+1}^s - p_i^s) \tag{3}$$

where p_i^s and p_{i+1}^s are the score pitches of two sequenced notes, and $\bar{p}_i$ and $\bar{p}_{i+1}$ are their observed median pitches. Similarly, harmonic interval error is defined as:

$$e_{i,A,j,B}^h = (\bar{p_{i,A}} - \bar{p}_{j,B}) - (p_{i,A}^s - p_{j,B}^s) \tag{4}$$

where $p_{i,A}^s$ and $p_{j,B}^s$ are the score pitches of two simultaneous notes from singers A and B respectively, and $\bar{p}_{i,A}$ and $\bar{p}_{j,B}$ are their observed median pitches.

The *mean absolute melodic interval error* (MAMIE) for M intervals is calculated as follows:

$$\text{MAMIE} = \frac{1}{M}\sum_{i=1}^{M} |e_i^m|. \tag{5}$$

The *mean absolute harmonic interval error* (MAHIE) is calculated similarly (where we simplify the notation and assume M harmonic intervals in total, indexed by i):

$$\text{MAHIE} = \frac{1}{M}\sum_{i=1}^{M} |e_i^h|. \tag{6}$$

Harmonic intervals were evaluated for all pairs of notes which overlap in time. If one singer sings two notes while the second singer holds one note in the same time period, two harmonic intervals are observed. Thus indices i and j in Eq. (4) are not assumed to be equal.

3.5.3 Note Stability

Pitch stability has been defined as the mean square pitch error of the note trajectory [17, 24], annotated using a fine time resolution, in this case Tony's default hop size of 5.8ms (section 3.4). We prefer to call this *pitch variability*, as higher values correspond to less stable notes. For a note trajectory for note i consisting of N frames, if the pitch of frame n is $p_{i,n}^f$ and the median pitch $\bar{p}_i$, the note variability ν_i is given by:

$$\nu_i = \frac{1}{N}\sum_{n=1}^{N} |p_{i,n}^f - \bar{p}_i|^2 \tag{7}$$

The *mean note variability* (MNV) is the mean variability of M notes:

$$\text{MNV} = \frac{1}{M}\sum_{i=1}^{M} \nu_i \tag{8}$$

4. RESULTS

The primary aim of this study was to test experimentally whether, and under what conditions, interaction is beneficial or detrimental to SATB intonation accuracy. We tested the intonation accuracy of individuals by pitch error (section 4.1), melodic interval error (section 4.2) and note stability (section 4.4); and tested the intonation of pairs of singers by harmonic interval error (section 4.3). In order to avoid biasing mean errors by outliers, where a participant sang a wrong note rather than an out-of-tune attempt at the correct pitch, all the tests exclude notes with pitch error or interval error larger in magnitude than one semitone. 96.4% of observed notes had an absolute pitch error less than one semitone.

4.1 Pitch Error

The first task is to investigate whether the ensemble depends on a certain vocal part to tune their pitch. After excluding the notes which have an absolute pitch error larger than one semitone (3.6%), most of the observed notes are relatively accurate (mean: 0.25 semitones; median: 0.26; std.dev.: 0.07).

We compute pitch error for the three non-isolated singers in each three-to-one condition and open condition, and analyse results by test condition. The MAPE was computed as an average across the three non-isolated singers and the five groups. For example, in the soprano isolated three-to-one condition, we average the pitch errors of alto, tenor, bass parts from each group and report the resulting MAPE. We compare these results with the performance of the same three singers in the open conditions.

A correlated samples analysis of variance (ANOVA) showed a significant difference in MAPE between three-to-one and open conditions (F(1,21625)=13, p<.001). The MAPE of the three-to-one condition is less than the MAPE of the open condition. We then performed separate ANOVAs for each isolated voice type (Table 1), and found that the results vary across test conditions. The bass and tenor isolated three-to-one conditions both showed significant differences, while the results for the other two voice types were not significant.

Test condition	Partial vs open condition
Soprano isolated	F(1,9391)=2.86, p=0.09
Alto isolated	F(1,9614)=0.61, p=0.11
Tenor isolated	F(1,9742)=5.07, p=0.02*
Bass isolated	F(1,10223)=14.39, p<.001***

Table 1: Results of correlated samples ANOVAs for three-to-one and open listening conditions (***p<.001; **p<.01; *p<.05))

These results suggest that the bass part is the most influential vocal part in all observed groups. However, the direction of influence is the opposite of that hypothesised: removing the bass vocal part from the ensemble reduces the observed pitch error on average.

The next ANOVA shows that the MAPE is significantly different between the test conditions in the three-to-one listening condition (F(3,12948)=28.67, p<.001). Table 2 shows the 95% confidence intervals, which demonstrate that the bass and tenor isolated conditions are significantly different from all other three-to-one conditions. The bass isolated condition has 4 cents MAPE less than soprano and alto isolated conditions, and 2 cents MAPE smaller than the tenor isolated condition.

Test condition	MAPE	Confidence interval
Soprano isolated	0.2484	[0.2420, 0.2548]
Alto isolated	0.2483	[0.2422, 0.2545]
Tenor isolated	0.2328	[0.2271, 0.2385]
Bass isolated	0.2082	[0.2028, 0.2135]

Table 2: Mean absolute pitch error (MAPE) and 95% confidence intervals for three-to-one test conditions, for all non-isolated singers and all groups.

These results contradict hypothesis one: when singers do not hear the bass part, they sing more accurately on average, as shown by comparisons within the three-to-one conditions and between the three-to-one and open conditions.

4.2 Melodic Interval Error

To test the influence of interaction on adjacent notes within a voice (hypothesis two), melodic interval error was calculated. 91.9% of the note pairs have a melodic interval error smaller than one semitone (mean:0.21; median:0.21; std.dev.:0.07).

We performed a correlated-samples ANOVA to test the effect of listening condition on MAMIE. The MAMIE is significantly different across listening conditions (F(2,18333)=27.96, p<.001). The listening condition of singing without hearing any partners (closed) has smaller MAMIE than the listening conditions with partners (partial and open). Table 3 shows the mean and confidence intervals for the three listening conditions where the closed listening condition has 3 cents smaller MAMIE than the open listening condition.

Listening condition	MAMIE	Confidence interval
Closed condition	0.1874	[0.1828, 0.1919]
Partial condition	0.2001	[0.1953, 0.2049]
Open condition	0.2138	[0.2102, 0.2174]

Table 3: Mean absolute melodic interval error (MAMIE) and 95% confidence intervals for each listening condition.

The acoustic feedback from other vocal parts increases MAMIE, which concurs with findings from previous research [15] and supports hypothesis two. The accompaniment from other vocal parts may mask the singer's own voice or distract the singer's attention from their own intonation. Alternatively, the increase in melodic interval error could be a side effect of deliberate adjustment of intonation to reduce harmonic interval error.

4.3 Harmonic Interval Error

Beside the intonation accuracy of individual singers, the accuracy of pairs of singers was also tested. There are four individual singers and up to six harmonic intervals simultaneously present at any point in time. All the harmonic intervals were observed under two circumstances: one-way interaction and two-way interaction.

In the partial conditions, some of the communication is only in one direction, so that any deliberate adjustment in harmonic interval must be attributed to the singer who can hear their partner. In this case, we have a *one-way interaction*. In the open conditions, both singers in a pair are able to adjust to each other, creating a *two-way interaction*. Taking soprano isolated conditions as an example, the harmonic intervals involving soprano are one-way interactions, and the harmonic intervals between alto, tenor and bass are two-way interactions (Figure 3).

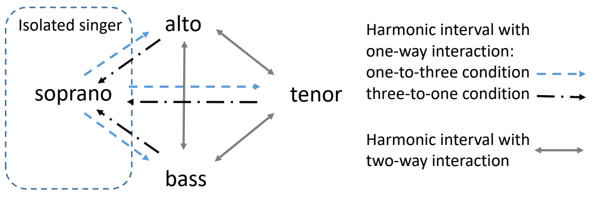

Isolated singer
alto
soprano
tenor
bass
Harmonic interval with one-way interaction:
one-to-three condition
three-to-one condition
Harmonic interval with two-way interaction

Figure 3: Interaction in the soprano isolated conditions

We compare the MAHIE for two-way interactions with those for one-way interactions in the three-to-one test conditions. MAHIE is significantly smaller for the two-way interactions than for one-way interactions (F(1,23659)=10.94, p<.001). This supports the third hypothesis, and indicates that acoustic feedback helps singers to interactively tune harmonic intervals.

However, no significant difference was found between MAHIE for different directions of intonation, that is the three-to-one condition versus the one-to-three condition (F(1,23524)=0.39, p=0.53). When one side of interactive intonation is without acoustic feedback, the direction of the feedback does not appear to influence the harmonic interval.

4.4 Note Stability

The note stability is measured by its converse, note variability (Eq. 7). The acoustic feedback of other singers not only has an influence on intonation accuracy (section 4.2) but also has an influence on note variability.

The note variability in the closed condition is significantly different from that in the partial and open conditions (F(1,23659)=41.23, p<.001), but no significant difference was found between the partial and open conditions (F(1,22514)=1.37, p=0.24). Note trajectories become less stable when singers can hear other singers in addition to their own voice, which is further evidence of interaction in intonation. This agrees with previous studies, which show that singers perform worse when singing with an unstable reference pitch [4, 16].

Moreover, the note variability is weakly positively correlated to the MAPE of individual notes (r=0.18, p<.001), but it is not obviously related to the singer (r=0.01, p=0.01) or training experience (r=0.08, p<.001).

The fourth hypothesis has been tested, and the results confirm that there is a relationship between the listening condition and note stability. This complements results from other research which assert that note stability of individual singers depends on emotional expression [5, 21]. Other possible relationships, such as a connection between musical training and note stability, were not supported by the experimental results.

5. DISCUSSION AND FUTURE WORK

This study tested four hypotheses using various metrics of singing accuracy and statistical tests. In each case, significant results were found. In three of the four cases, the results supported the hypotheses, however for the first hypothesis, the direction of the observed effect was the opposite of what was predicted.

Participants noted that the bass part (male singer) is the most difficult vocal part to recruit. It is possible that this leads to a lower average standard among bass singers. A comparison of pitch error by vocal type reveals that the bass vocal part has a larger MAPE than the other vocal parts. This may be the cause of the unexpected result for the bass isolated condition: i.e. because the bass voice had greater pitch error, other parts which tuned to the bass also increased their pitch error.

The factor of interaction, that is when singers can hear each other, increases the pitch error of the individual singers but decreases the harmonic interval error between the singers. Although these results may appear to be contradictory, this can occur when interval errors accumulate, and the sung pitches drift away from the initial tonal reference, as has been demonstrated by Howard [6].

Many factors of influence have been researched which are crucial for singing, such as age and gender (boys are more likely to sing out of tune than girls), and individual differences [27]. As it is not possible to cover all aspects in this paper, we leave the analysis of results from the questionnaire to future work, including the investigation of the relationships between intonation accuracy and active engagement with music, perceptual abilities, musical training and singing ability.

6. CONCLUSIONS

For analysis of the effect of interaction on intonation in unaccompanied SATB singing, we designed a novel experiment and tested the intonation accuracy of five groups of singers in a series of test and listening conditions. The results confirm that interaction exists between singers and influences their intonation, and that intonation accuracy depends on which other singers each individual singer can hear.

In particular, we observed that the three-to-one bass isolated test condition had a significantly lower MAPE compared with other three-to-one conditions, and compared with the open condition. In other words, singers were more accurate when they could not hear the bass. This surprising result might be due to the fact that the bass singers were less accurate on average than other singers in this experiment.

We observed increases in pitch error and melodic interval error when singers could hear each other. The closed condition had the smallest MAMIE, while the open condition had the largest. At the same time, acoustic feedback decreased the harmonic interval error, while the direction of the feedback did not influence the harmonic interval error.

Interaction also has the effect of reducing the note stability, or increasing its variability. Pitch within a note varies more when singers hear each other, as one might expect if the singers are adjusting their intonation to be in tune with each other.

In conclusion, this paper addresses a gap in singing intonation studies, by investigating the effects of interaction between singers. We found that interaction significantly influences the pitch accuracy, leading to increases in the pitch error, melodic interval error, and note stability but a decrease in the harmonic interval error. Although many aspects of the data remain to be explored, we hope the current results provide useful information and better understanding of interactive intonation.

7. ACKNOWLEDGEMENT

Many thanks to all of the participants who contributed to this project, including the QMUL *A Capella* Society, QMUL Music Society, London Philharmonic Choir, the Hi-Fan Vocal Group. We also thank Marcus Pearce, Daniel Stowell and Christophe Rhodes for their advice on experimental design. Jiajie Dai is supported by a China Scholarship Council and Queen Mary Joint PhD Scholarship.

8. DATA AVAILABILITY

Annotated data, experimental score and code to reproduce our results are available at: `https://code.soundsoftware.ac.uk/projects/analysis-of-interactive-intonation-in-unaccompanied-satb-ensembles/repository`.

9. REFERENCES

[1] Per-Gunnar Alldahl. *Choral Intonation*. Gehrmans, 2008.

[2] Jocelei C S Bohrer. Intonational Strategies in Ensemble Singing. *Doctoral thesis, City University London*, 2002.

[3] Brian J Brandler and Zehra F Peynircioglu. A Comparison of the Efficacy of Individual and Collaborative Music Learning in Ensemble Rehearsals. *Journal of Research in Music Education*, 63(3):281–297, 2015.

[4] Jiajie Dai and Simon Dixon. Analysis of Vocal Imitation of Pitch Trajectories. *17th ISMIR Conference*, 2016.

[5] Janina Fyk. *Melodic Intonation, Psychoacoustics and the Violin*. Organon, 1995.

[6] David M Howard. Intonation drift in a capella soprano, alto, tenor, bass quartet singing with key modulation. *Journal of Voice*, 21(3):300–315, 2007.

[7] Joyce B Kennedy and Michael Kennedy. *The Concise Oxford Dictionary of Music*. Oxford University Press, 2004.

[8] Joan La Barbara. Voice is the Original Instrument. *Contemporary Music Review*, 21(1):35–48, 2002.

[9] Peggy A Long. Relationships Between Pitch Memory in Short Melodies and Selected Factors. *Journal of Research in Music Education*, 25(4):272–282, 1977.

[10] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Bello J. Salamon and, J. Dai, and S. Dixon. Tony: a Tool for Efficient Computer-aided Melody Note Transcription. In *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR)*, 2015.

[11] 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)*, pages 659–663, 2014.

[12] Matthias Mauch, Klaus Frieler, and Simon Dixon. Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory. *The Journal of the Acoustical Society of America*, 136(1):401–411, 2014.

[13] Steven J Mithen. *The Singing Neanderthal: A Search for the Origins of Art, Religion, and Science*. Harvard University Press, Cambridge, MA, 2007. esp. Ch. 16, pp. 246–265.

[14] Daniel Müllensiefen, Bruno Gingras, Jason Musil, and Lauren Stewart. The musicality of non-musicians: An index for assessing musical sophistication in the general population. *PloS one*, 9(2):e89642, 2014.

[15] Dirk Mürbe, Friedemann Pabst, Gert Hofmann, and Johan Sundberg. Significance of Auditory and Kinesthetic Feedback to Singers' Pitch Control. *Journal of Voice*, 16(1):44–51, 2002.

[16] Peter Q Pfordresher and Steven Brown. Poor-pitch Singing in The Absence of "Tone Deafness". *Music Perception: An Interdisciplinary Journal*, 25(2):95–115, 2007.

[17] Peter Q Pfordresher, Steven Brown, Kimberly M Meier, Michel Belyk, and Mario Liotti. Imprecise Singing is Widespread. *The Journal of the Acoustical Society of America*, 128(4):2182–2190, 2010.

[18] John Potter, editor. *The Cambridge Companion to Singing*. Cambridge University Press, 2000.

[19] E. Prout. *Harmony: Its Theory and Practice*. Cambridge University Press, 2011.

[20] R. Seaton, D. Pim, and D. Sharp. Pitch Drift in a Cappella Choral Singing-Work in Progress Report. *Institute of Acoustics Annual Spring Conference 2013*, 2013.

[21] Johan Sundberg, Filipa M B Lã, and Evangelos Himonides. Intonation and Expressivity: A Single Case Study of Classical Western Singing. *Journal of Voice*, 27(3):391.e1–e8, 2013.

[22] Annie H Takeuchi and Stewart H Hulse. Absolute pitch. *Psychological Bulletin*, 113(2):345–361, 1993.

[23] Hiroko Terasawa. Pitch drift in choral music, 2004. Music 221A final paper, Center for Computer Research in Music and Acoustics, Stanford University, CA.

[24] Sten Ternström and Johan Sundberg. Intonation Precision of Choir Singers. *The Journal of the Acoustical Society of America*, 84(1):59–69, 1988.

[25] Richard A Warren and Meagan E Curtis. The Actual vs. Predicted Effects of Intonation Accuracy on Vocal Performance Quality. *Music Perception: An Interdisciplinary Journal*, 33(2):135–146, 2015.

[26] Graham F Welch. Singing as communication. *Musical Communication*, pages 239–259, 2005.

[27] Graham F Welch, Desmond C Sergeant, and Peta J White. Age, Sex, and Vocal Task as Factors In Singing "in Tune" During The First Years of Schooling. *Bulletin of the Council for Research in Music Education*, pages 153–160, 1997.

[28] Vivienne M Young and Andrew M Colman. Some Psychological Processes in String Quartets. *Psychology of Music*, 7(1):12–18, 1979.

AUTOMATIC DRUM TRANSCRIPTION FOR POLYPHONIC RECORDINGS USING SOFT ATTENTION MECHANISMS AND CONVOLUTIONAL NEURAL NETWORKS

Carl Southall, Ryan Stables and Jason Hockman
DMT Lab, Birmingham City University
Birmingham, United Kingdom
{carl.southall, ryan.stables, jason.hockman} @bcu.ac.uk

ABSTRACT

Automatic drum transcription is the process of generating symbolic notation for percussion instruments within audio recordings. To date, recurrent neural network (RNN) systems have achieved the highest evaluation accuracies for both drum solo and polyphonic recordings, however the accuracies within a polyphonic context still remain relatively low. To improve accuracy for polyphonic recordings, we present two approaches to the ADT problem: First, to capture the dynamism of features in multiple time-step hidden layers, we propose the use of soft attention mechanisms (SA) and an alternative RNN configuration containing additional peripheral connections (PC). Second, to capture these same trends at the input level, we propose the use of a convolutional neural network (CNN), which uses a larger set of time-step features. In addition, we propose the use of a bidirectional recurrent neural network (BRNN) in the peak-picking stage. The proposed systems are evaluated along with two state-of-the-art ADT systems in five evaluation scenarios, including a newly-proposed evaluation methodology designed to assess the generalisability of ADT systems. The results indicate that all of the newly proposed systems achieve higher accuracies than the state-of-the-art RNN systems for polyphonic recordings and that the additional BRNN peak-picking stage offers slight improvement in certain contexts.

1. INTRODUCTION

Music notation, which portrays the instrumentation and playing techniques used within a musical recording, is produced through the process of automatic music transcription (AMT). Fast and accurate production of music notation would benefit multiple areas including the creative, analytical and educational industries. The majority of previous AMT systems has been developed to address pitched instrumentation, while relatively few systems have focussed on the transcription of percussive instruments. Automatic drum transcription (ADT) systems soley focus on producing notation for drum instruments, which strongly portray the rhythm, groove and feel of the piece. High ADT accuracies have been achieved on audio recordings containing only basic drum classes such as kick drum, snare drum and hi-hats [15, 19]. However, accuracies are significantly lower in a *polyphonic context*—in which the recordings contain either additional percussion (e.g., toms, cymbals) or pitched instrumentation (e.g., guitar, piano) [20].

1.1 Background

Several early ADT systems have been proposed that perform well on solo drum recordings [3, 5, 10, 13, 18, 23], however a relatively small number of systems have demonstrated the capacity for high performance in a polyphonic context. Wu and Lerch [21] proposed a non-negative matrix factorisation technique with a specialised basis function to capture harmonic activity outside of those for the drum classes under observation. Paulus et al. [12] used a hidden Markov model to detect the presence of individual drum onsets within frames of a spectrogram. Southall et al. [15] and Vogl et al. [19] also formalise ADT as a frame-wise drum onset detection problem, using recurrent neural networks (RNN) for classification. Southall et al. [15] presented a bidirectional RNN (BRNN) system and Vogl et al. [19] presented a RNN system with time-shifted classification labels. RNN systems have achieved the best drum solo performance to date, however their accuracies in the polyphonic context has been marginalised. Vogl et al. [20] later proposed the incorporation of gated recurrent unit (GRU) cells, which incorporate more time-step information into the RNN model, resulting in the highest ADT accuracies to date in a polyphonic context.

1.2 Motivation

The increase in accuracy achieved by the GRU RNN in [20] over the standard RNN in [19] demonstrates the effect of storing additional information on classification performance. In a solo drum context, instrumentation overlap is limited to the drums under observation, whereas in a polyphonic context, drums are present along with other instruments. This may obscure the presence of features belonging to the drums under observation, and is mitigated by the incorporation of additional time-step information in the

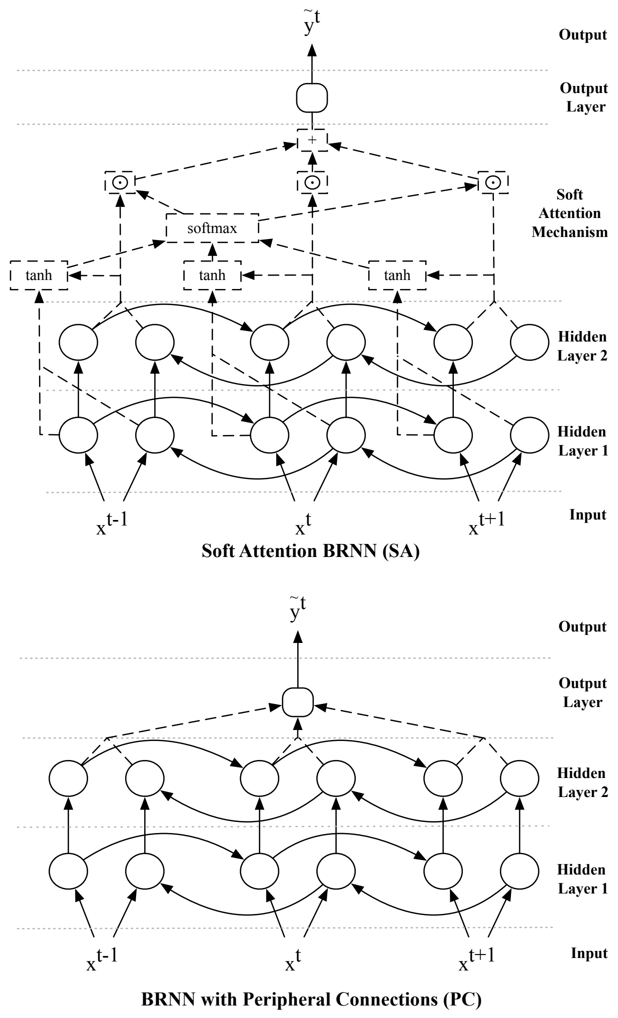


Figure 1: Overview of the proposed SA and PC systems. Solid lines depict connections of a standard BRNN configuration and dashed lines depict additional SA and PC connections when attention number $a = 1$. x^t and $\tilde{y}$ are input features and output activation function at time step t. $\odot$ in the SA system represents element-wise multiplication.

GRU RNN. Inclusion of additional information in previous RNN ADT systems however, is still restricted by a bottleneck at the output layer, which is determined by the hidden state sizes. Additionally, larger input feature sizes can not be used at each time step, due to the computational cost of fully-connected layers. We present two approaches in an attempt to overcome the above-stated limitations to ADT in a polyphonic context: First, to capture the dynamism of features in multiple time-step hidden layers, we propose the use of soft attention mechanisms (SA) and an alternative RNN configuration containing additional peripheral connections (PC). Second, to capture these same trends at the input level, we propose the use of a convolutional neural network (CNN), which uses a larger set of time-step features. To further improve the accuracy of the systems, we also propose the use of an additional BRNN for selecting drum onsets from the output activation functions, as peak-picking within a polyphonic context has proven to be more difficult than that of drum solos [15, 19, 20].

The remainder of this paper is structured as follows: Section 2 presents our three newly proposed systems and our new peak-picking technique. The evaluation is outlined in Section 3 and the results are presented in Section 4. Conclusions and future work are provided in Section 5.

2. METHOD

For the three new proposed systems, we use the same frame-wise classification ADT technique outlined in [15]. Input features are fed into a separate pre-trained neural network for each instrument under observation. Peak-picking is then performed on the resulting activation functions to determine onset locations.

2.1 Soft Attention BRNN (SA)

Attention mechanisms allow the network to focus on different parts of the data stored within a RNN for different tasks. This is achieved by enabling the information fed to the output layer to be created from multiple time-step final hidden layers. This was initially achieved through binary connections in hard attention mechanisms and then by weighted connections in soft attention mechanisms (SA). They have improved RNN results in multiple fields including: machine translation [1] and image caption generation [11, 22]. An overview of the implemented SA ADT system based on [6] is given at the top of Figure 1. We use a BRNN with each hidden layer containing 100 long short-term memory cells with peephole connections (LSTMP) as the basis of the system. This is due to its ability to pass information through its memory cell c, which is updated using the input i, forget f and output o gates. The equations for a LSTMP cell layer are:

$$i_l^t = \sigma(Wi_l\big[x^t, h_l^{t-1}, c_l^{t-1}\big] + bi_l) \tag{1}$$

$$f_l^t = \sigma(Wf_l\big[x^t, h_l^{t-1}, c_l^{t-1}\big] + bf_l) \tag{2}$$

$$\tilde{c_l^t} = tanh(Wc_l\big[x^t, h_l^{t-1}, c_l^{t-1}\big]) \tag{3}$$

$$c_l^t = f^t \odot c_l^{t-1} + i^t \odot \tilde{c_l^t} + bc_l) \tag{4}$$

$$o_l^t = \sigma(Wo_l\big[x^t, h_l^{t-1}, c_l^t\big] + bo_l) \tag{5}$$

$$h_l^t = o_l^t \odot tanh(c_l^t), \tag{6}$$

where h_l^t is the hidden layer of layer l at time step t, the weights W, and the biases b. x is the input feature where $x^t = h_{l-1}^t$ if $l > 1$. After each hidden layer dropouts [16] are implemented with a probability of p. Based on preliminary tests, we use 2 hidden layers as using more did not improve performance.

The SA feeds the LSTM BRNN output into the output layer as a weighted combination of $2a + 1$ time-step final hidden layers, centred on the current time-step t, where a is the attention number. First, an intermediate variable m is determined for each attention step i ($i = t-a : t+a$) using the concatenated outputs of the forwards and backwards directional LSTMs Q ($Q = \big[y_L^{\rightarrow}, y_L^{\leftarrow}\big]$) and a context U:

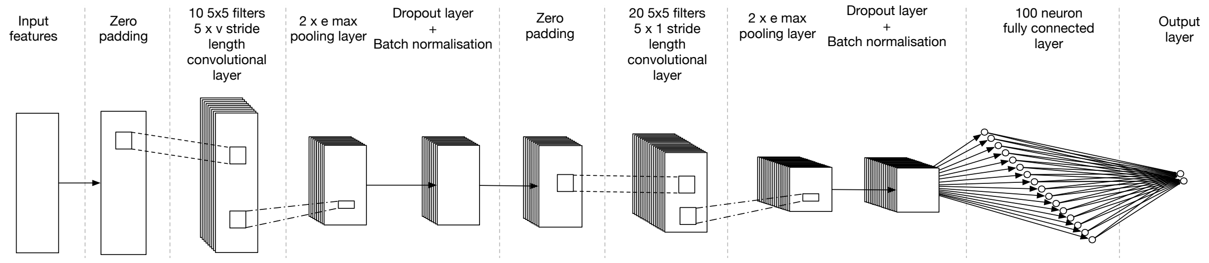

Input features
Zero padding
10 5x5 filters 5 x v stride length convolutional layer
2 x e max pooling layer
Dropout layer + Batch normalisation
Zero padding
20 5x5 filters 5 x 1 stride length convolutional layer
2 x e max pooling layer
Dropout layer + Batch normalisation
100 neuron fully connected layer
Output layer

Figure 2: Overview of the proposed CNN. Information flows through the network from left to right; solid lines represent connections, with dashed lines representing convolution and dash-dotted lines representing max pooling.

$$m^i = tanh(W_q Q^i + W_U U). \tag{7}$$

The aim of U is to feed the SA mechanism information regarding the wider scope of the current data. We first attempted to use the the cell state of the final hidden layer c_L as in [8], however using the outputs of the first hidden layer ($U = \left[h_1^{t\rightarrow}, h_1^{t\leftarrow}\right]$) resulted in better performance during preliminary testing. The attention weights s are determined using a softmax function across i:

$$s^i \propto \exp {W_m}^T m^i, \tag{8}$$

so that $\sum_i s^i = 1$. The output layer input z is then calculated using Q and s and fed into an output layer similar to the BRNN architecture in [15]:

$$z = \sum_i s^i \odot Q^i \tag{9}$$

$$\tilde{y}^t = softmax(W_z z + b_z). \tag{10}$$

s can be thought of as percentage determining how much information from each of the time-step final hidden layers Q is used in the input to the output layer z.

2.2 BRNN with Peripheral Connections (PC)

Although the SA system allows the information fed into the output layer to be determined directly from multiple time-step hidden layers, the amount of information is still limited by the hidden layer size. We propose an increase in the amount of information passed to the output layer by including direct connections from multiple time-step hidden layers to the output layer, which we term *peripheral connections* (PC). An overview of the PC system is presented at the bottom of Figure 1. The PC system is the same as the SA system in eqns. 1–6. However, these connections are implemented in the output layer using:

$$\tilde{y}^t = softmax(W_v Q^{t-a:t+a} + b_v), \tag{11}$$

where v highlights the weights and biases belonging to the PC output layer and $Q^{t-a:t-a}$ is the concatenation of multiple LSTM time-step outputs:

$$Q = \left[h_L^{\rightarrow t-a}, ..., h_L^{\rightarrow t+a}, h_L^{\leftarrow t-a}, ..., h_L^{\leftarrow t+a}\right]. \tag{12}$$

If $a = 0$, then both the SA and PC systems are the same as a standard BRNN network with LSTMP cells.

2.3 Convolutional Neural Network (CNN)

As RNNs contain fully-connected layers, large input feature sizes can not be used as they become extremely computationally expensive. Convolutional neural networks (CNN) overcome this problem by combining feature learning, dimensionality reduction and classification stages in a single trainable network. This ability has enabled CNNs to achieve higher accuracies than RNNs in the closely related fields of onset detection [14] and downbeat detection [4]. We propose to use a convolutional neural network to enable multiple time-step features to be used as input for each frame classification. An overview of the implemented CNN ADT system is outlined in Figure 2 where j frames on either side of the current frame t are included in the input features and different values of v and e are used as j is increased. It consists of two sets of convolutional, max pooling, dropout [16], and batch normalisation [7] layers before a 100-neuron fully-connected layer and a two neuron softmax output layer.

2.4 Implementation

The newly proposed models are implemented using the Tensorflow Python library. Four SA and PC systems (`SA1`, `SA2`, `SA3` and `SA5`) and (`PC1`, `PC2`, `PC3` and `PC5`) are implemented where a = [1, 2, 3, 5] and four CNN systems (`CNN2`, `CNN5`, `CNN10`, and `CNN20`) are implemented where j = [2, 5, 10, 20]. These values are chosen as they cover various ranges of important information regarding the typical envelope length of drums.

2.4.1 Input Features

In order for an audio file to be processed by the neural networks, it must be procedurally segmented into framewise spectral features. First, the input audio (16-bit .wav file sampled at 44100 kHz) is segmented into T frames using a Hanning window of n samples ($n = 2048$) with a $\frac{n}{4}$ hopsize. A frequency representation of each of the frames is then created using the magnitudes of a discrete Fourier transform resulting in a $\frac{n}{2}$ x T spectrogram. The spectrogram is input into the SA systems in a frame-wise manner and as a combination of frames (j frames either side of the current frame t) for the CNN systems.

2.4.2 Peak Picking

Once the activation functions $\tilde{Y}$ are output from the systems, peak-picking is used to identify the onset candidates.

In this paper, we implement two peak-picking strategies for each of the systems. The first approach, termed mean threshold (MT), is an updated version of the technique used in [15], in which a threshold is determined for each frame (τ^t) using:

$$\tau^t = mean(\tilde{y}^{t-\theta} : \tilde{y}^{t+\theta}) * \lambda \tag{13}$$

$$\tau^t = \begin{cases} tmax, & \tau > tmax \\ tmin, & \tau < tmin, \end{cases} \tag{14}$$

where θ sets the number of frames in each direction to calculate the mean, λ is a constant and $tmax$ and $tmin$ are the possible maximum and minimum values. The current frame of $\tilde{y}$ is accepted as an onset if it is the maximum of a surrounding number of frames and above the threshold τ:

$$O^t = \begin{cases} 1, & \tilde{y}^t == max(\tilde{y}^{t-\Omega} : \tilde{y}^{t+\Omega}) \quad \& \quad \tilde{y}^t > \tau^t \\ 0, & otherwise, \end{cases} \tag{15}$$

where $O(t)$ represents an onset at time step t and Ω is the number of frames on either side of the current frame t used to calculate the maximum.

For the second approach we train an additional neural network using the activation functions from the training data in an attempt to learn to identify the drum onsets more difficult to detect. To do this we use a BRNN, with a single 10 LSTMP-cell hidden layer and a softmax output layer. The output of the new BRNN is then processed by the MT technique (eqns. 10–12), we refer to this second technique as BRNN-MT.

2.4.3 Training

The three models and the BRNN-MT peak-picking networks are trained using the Adam optimiser [9] with a learning rate of 0.003. The training data is created by generating a feature matrix from input features x and an associated class vector from the target activation functions Y. Mini-batch gradient descent (batch size = 1000) created from 10 segments (segment length = 100) is used. The activation function output from the models $\tilde{Y}$ are used as the input to the BRNN-MT networks which are trained using the same targets used to train the systems. A new BRNN-MT network is trained independently for each system in an attempt to increase adaptability, similar to [2]. Training is stopped when the following criteria have been met: (1) a minimum of 10 epochs have commenced; and (2) the validation set accuracy has not increased between epochs. To ensure training commences correctly, the weights are initialised using a random uniform distribution scaled to keep constant variance [17] and the biases are initialised to zero. Cross entropy is used as the loss function.

3. EVALUATION

To evaluate the newly proposed methods along with the current state-of-the-art systems, we implement four evaluations similar to those carried out in [15, 19, 20], along with an additional evaluation to test the generalisability of the systems. The systems are trained to identify kick drum, snare drum and hi-hat onsets. The first evaluation, termed *drum solo*, aims to demonstrate system performance on drum solo recordings that contain only the three drum instruments under observation. The second evaluation, termed *drum mixture*, aims to demonstrate system performance in a drum-only polyphonic context, where the recordings contain additional drum instrumentation to those under observation (e.g., toms and cymbals). The third evaluation, termed *multi-instrument mixture*, aims to demonstrate system performance in a fully-polyphonic context where multiple instruments are present in addition to the drum instruments under observation (e.g., piano and guitar) and the fourth evaluation, termed *cross-context*, aims to test the systems adaptability to before unseen timbres. The newly proposed evaluation, termed *multi-context*, aims to test the ability of a single system to be trained and used in multiple contexts.

3.1 Evaluation Methodology

F-measure is used as the evaluation metric with precision and recall determined using the onset candidates from the peak-picking stage. Detected onsets are accepted as true positives if they fall within 50ms of the ground truth annotations. The individual instrument F-measures are calculated as the mean F-measure across test tracks and the mean instrument F-measure is calculated as the mean F-measure across the individual instruments. The peak-picking parameters (θ, λ, $tmax$, $tmin$ and Ω) are found using a grid-search on the validation set and the dropout probability p is set to 0.25.

3.1.1 Drum Solo Evaluation

To test the capability of the systems in the *drum solo* evaluation, we use the updated version of the IDMT-SMT-Drums dataset [3]. This dataset contains 104 tracks divided into three subsets (20 *real drum* tracks, 14 *techno drum* tracks, and 70 *wave drum* tracks) with an average track length of 15 seconds. The dataset is divided by track in equal distributions across the three subsets into 70% training 15% validation and 15% test sets. The training set is used to train the neural network systems, the validation set to prevent overfitting during training and to optimise the peak-picking parameters, and the test subset is used as unseen data for testing. The four SA systems, the four PC systems and the four CNN systems are evaluated along with two current state-of-the-art ADT systems: (1) tanhB, a BRNN system containing tanh cells [15] and (2) lstmpB, a BRNN system containing LSTMP cells. The LSTMP architecture was chosen as it outperformed GRU cells in preliminary testing on the same datasets. Drum onsets are selected from the output activation functions using the two peak-picking techniques.

3.1.2 Drum Mixture and Multi-instrument Evaluations

To determine system performance in a polyphonic context we use the *minusone* subset of the ENST Drums dataset

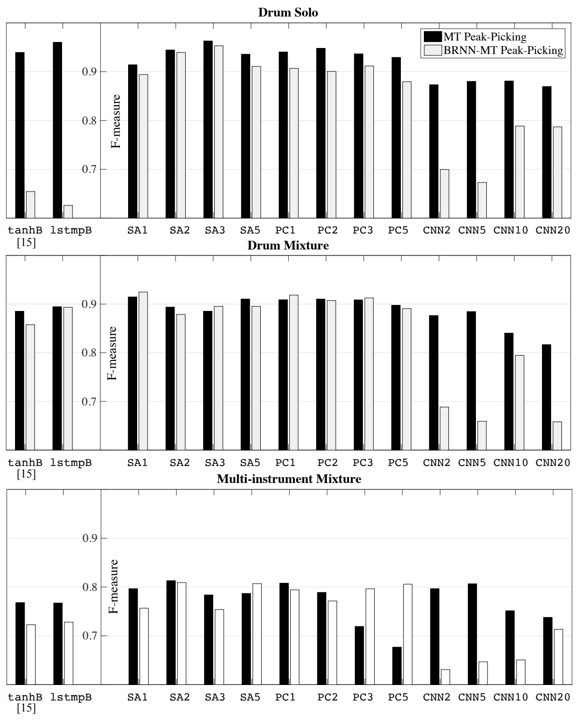


Figure 3: Mean instrument F-measures for *drum solo* (top), *drum mixture* (middle) and *multi-instrument mixture* (bottom) evaluations. Previous state-of-the-art RNN systems are on left and the SA, PC and CNN systems on right.

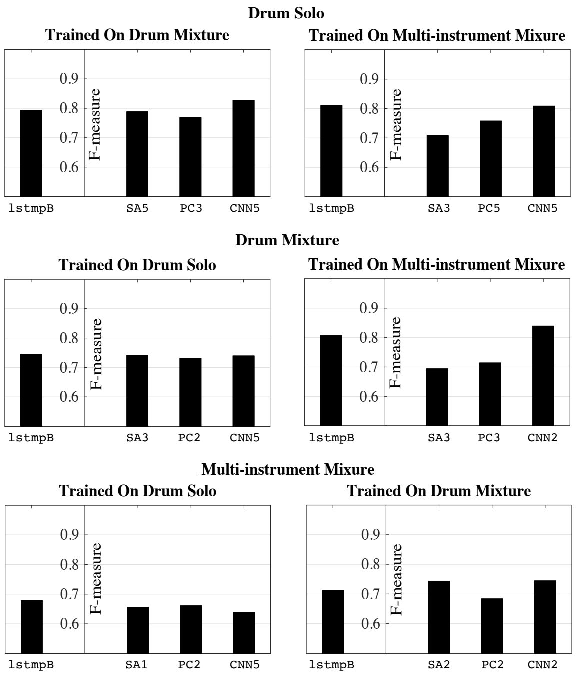


Figure 4: Mean instrument F-measure results with `MT` peak-picking for the *cross-context* evaluation: *drum solo* combinations (top); *drum mixture* combinations (middle); and *multi-instrument mixture* combinations (bottom).

[5]. The dataset contains 64 tracks divided into three different drummers (21 tracks by drummer 1, 22 tracks by drummer 2, and 21 tracks by drummer 3) with an average track length of 55 seconds. The dataset is composed of drum-only recordings which contain multiple drum instruments as well as accompaniment files. The drum only recordings are used for the *drum mixture* evaluation and the drum-only recordings are mixed with the accompaniment files using a ratio of 2/3 to 1/3 respectively for the *multi-instrument mixture* evaluation. The same training, validation and evaluation procedures are used as in the *drum solo* evaluation (Section 3.1.1).

3.1.3 Cross-context Evaluation

To test the adaptability of the trained systems to before unseen contexts we use the three systems trained in the previous evaluations (i.e., *drum solo*, *drum mixture*, and *multi-instrument mixture*) to process the datasets from the other two evaluations. This results in six *cross-context* evaluation combinations (e.g., train with *drum solo* test with *multi-instrument mixture*).

3.1.4 Multi-context Evaluation

To test how well a single system can be trained to adapt and perform in multiple contexts, we combine the training and validation data from the *drum solo*, *drum mixture* and *multi-instrument mixture* evaluations. The test data from the three evaluations is then processed using the single newly trained systems. Of the five evaluations this is the most realistic scenario.

4. RESULTS AND DISCUSSION

4.1 Drum Solo, Drum Mixture and Multi-instrument Mixture Results

Figure 3 highlights the mean instrument F-measure results of the SA, PC, CNN, and two previous state-of-the-art systems with both of the peak-picking strategies for the *drum solo*, *drum mixture* and *multi-instrument mixture* evaluations. The SA systems achieve the highest mean instrument F-measure in all three evaluations; 0.9880 (`SA3`), 0.9287 (`SA1`) and 0.9274 (`SA2`) respectively. The PC systems achieve higher F-measures in the *drum mixture* and *multi-instrument mixture* evaluations and the CNN systems achieve higher F-measures than the state-of-the-art systems in the *multi-instrument mixture* evaluation. This demonstrates that within the harder polyphonic contexts, allowing the output layer to access multiple hidden states and including the input features of multiple frames does enable higher performance to be achieved. The `BRNN-MT` peak-picking strategy improves the results of some of the SA and PC systems in both the *drum mixture* and *multi-instrument mixture* evaluations, demonstrating that the `BRNN-MT` strategy is able to improve performance in some contexts by learning to identify peaks within the noisier activation functions. For both the SA and PC systems the systems where $a \leq 3$ achieved the highest F-measures, we believe this is because of the extra information in the `SA5` and `PC5` systems is beyond the scope of the

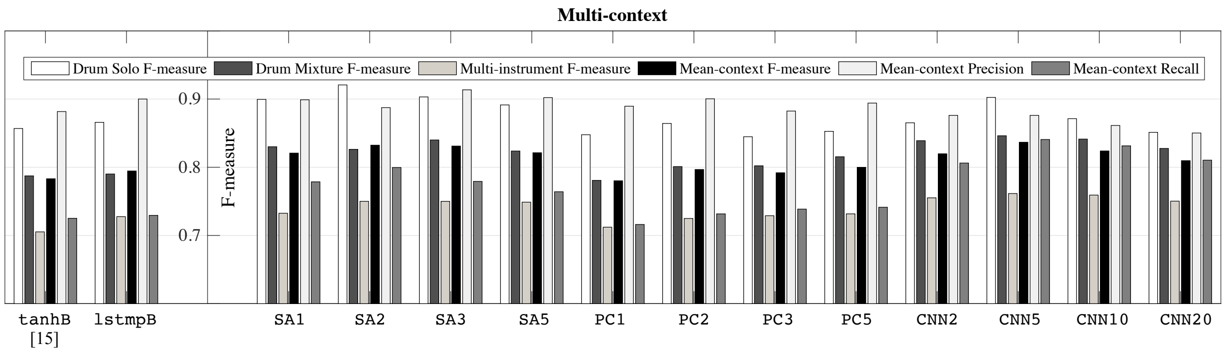


Figure 5: Results of the *multi-context* evaluation. For each system using the MT peak-picking technique the *drum solo*, *drum mixture*, *multi-instrument mixture*, and mean-context F-measures are shown in addition to the mean-context precision and mean-context recall.

onset and so has a negative effect on the performance. A similar trend is seen with the CNN systems which again can be explained by the larger input feature sizes reducing the impact of the relevant features. We believe that due to the *drum solo* evaluation being a relatively simple task, the less-complex RNN systems are able to achieve similar accuracies to the newly proposed systems and the CNN performs poorly on this same task due to noisy output activation functions which are the result of not passing information between time steps. This would also explain why the BRNN-MT strategy did not improve the results for the CNN systems and for any of the systems in the *drum solo* evaluation.

4.2 Cross-context Results

For each cross evaluation combination the top performing configuration of the existing state-of-the-art RNN, SA, PC and CNN systems using the MT peak-picking technique is displayed in Figure 4. The highest performing CNN system achieves a higher mean instrument F-measure than the highest performing current state-of-the-art RNN system (lsmtpB) in three out of the six combinations, the highest performing SA system only outperforms the current state-of-the-art RNN system in one of the combinations and the PC doesn't out perform the RNN system in any combinations. This suggests that the CNN system is more adaptable than the SA and PC systems even though the SA and PC systems achieve higher mean instrument F-measures than the CNN systems in the previous three evaluations. None of the highest accuracies were achieved by systems that used the BRNN-MT peak-picking strategy, which suggests that it is not suited for adapting to unseen situations.

4.3 Multi-context Results

Figure 5 highlights the *drum solo*, *drum mixture*, *multi-instrument mixture*, and mean-context F-measures using the MT peak-picking technique. Also included are the mean-context precision, and recall for each of the systems in the *multi-context* evaluation. The SA and CNN systems outperform the existing state-of-the-art and PC systems, further demonstrating the high performance of the SA systems and the adaptability of the CNN systems. This is achieved through higher recall, but not necessarily higher precision, suggesting that the improvement made by these systems is due to their ability to produce fewer false spikes within the resulting activation functions. All of the highest context F-measures were lower than the F-measures achieved by the systems trained in the single context focused evaluations (i.e., *drum solo*, *drum mixture*, and *multi-instrument mixture* evaluation) demonstrating that a system trained in multiple contexts can not outperform systems trained solely in one situation. The BRNN-MT peak-picking strategy again does not improve the performance of any of the systems in this evaluation.

5. CONCLUSIONS AND FUTURE WORK

We have presented three new neural network based systems for ADT in a polyphonic context: First, SA and PC systems that enable multiple time-step hidden states to be accessed by the output layer; and second, a CNN system that allows larger input feature sizes to be used. The results from the conducted evaluations demonstrate that all of the newly proposed systems achieve higher accuracies than the current state-of-the-art systems in polyphonic contexts, highlighting the effect of increased access to more information. Of all the tested systems, the SA performs best in either the single or multi-context, while the CNN systems perform best in situations in which the context is unseen. A possible future step would be to combine the SA and CNN systems into a single system possibly allowing the system to work in both situations (i.e., single and multiple contexts). An open source version of the newly proposed ADT systems can be found within the ADT libray (ADTLib). [1]

6. REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the international Conference on Learning Representations*, 2015.

[1] https://github.com/CarlSouthall/ADTLib

[2] Sebastian Böck, Jan Schlüter, and Gerhard Widmer. Enhanced peak picking for onset detection with recurrent neural networks. In *Proceedings of the 6th International Workshop on Machine Learning and Music (MML)*, pages 15–18, Prague, Czech Republic, 9 2013.

[3] Christian Dittmar and Daniel Gärtner. Real-time transcription and separation of drum recordings based on NMF decomposition. In *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, pages 187–194, Erlangen, Germany, September 2014.

[4] Simon Durand, Juan Pablo Bello, Bertrand David, and Gaël Richard. Robust downbeat tracking using an ensemble of convolutional networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):76–89, 2017.

[5] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(3):529–540, 2008.

[6] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015.

[7] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[8] Jeremy Irvin, Elliott Chartock, and Nadav Hollander. Recurrent neural networks with attention for genre classification. 2016.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] Marius Miron, Matthew E. P. Davies, and Fabien Gouyon. An open-source drum transcription system for pure data and max MSP. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 221–225, Vancouver, BC, Canada, 2013.

[11] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[12] Jouni Paulus. *Signal Processing Methods for Drum Transcription and Music Structure Analysis*. PhD thesis, Tampere University of Technology, Tampere, Finland, 2009.

[13] Axel Röbel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. On automatic drum transcription using non-negative matrix deconvolution and itakura saito divergence. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 414–418, Brisbane, Australia, 2015.

[14] Jan Schlüter and Sebastian Böck. Improved musical onset detection with convolutional neural networks. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6979–6983. IEEE, 2014.

[15] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 591–597, New York City, United States, August 2016.

[16] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] David Sussillo and Laurence F. Abbott. Training very deepnonlinear feed-forward networks with smart initialization. *arXiv preprint arXiv*, 1412, 2014.

[18] Lucas Thompson, Simon Dixon, and Matthias Mauch. Drum transcription via classification of bar-level rhythmic patterns. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 187–192, Taipei, Taiwan, 2014.

[19] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, New York City, United States, August 2016.

[20] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum transcription from polyphonic music with recurrent neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 201–205, New Orleans, Louisiana, United States, March 2017.

[21] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, Malaga, Spain, October 2015.

[22] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[23] Kazuyoshi Yoshii, Masataka Goto, and Hiroshi G. Okuno. Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):333–345, 2007.

AUTOMATIC DRUM TRANSCRIPTION USING THE STUDENT-TEACHER LEARNING PARADIGM WITH UNLABELED MUSIC DATA

Chih-Wei Wu, Alexander Lerch
Georgia Institute of Technology, Center for Music Technology
{cwu307, alexander.lerch}@gatech.edu

ABSTRACT

Automatic drum transcription is a sub-task of automatic music transcription that converts drum-related audio events into musical notation. While noticeable progress has been made in the past by combining pattern recognition methods with audio signal processing techniques, the major limitation of many state-of-the-art systems still originates from the difficulty of obtaining a meaningful amount of annotated data to support the data-driven algorithms. In this work, we address the challenge of insufficiently labeled data by exploring the possibility of utilizing unlabeled music data from online resources. Specifically, a student neural network is trained using the labels generated from multiple teacher systems. The performance of the model is evaluated on a publicly available dataset. The results show the general viability of using unlabeled music data to improve the performance of drum transcription systems.

1. INTRODUCTION

Data availability, listed by Schedl et al. as one of the open challenges in the field of Music Information Retrieval (MIR) [21], is an important problem that concerns a large variety of data-driven MIR systems. To create intelligent music (analysis) systems, music data with detailed annotations is crucial as training input for machine learning algorithms. However, multiple constraints impede the availability of large datasets, including (i) the complexity and variety of music in terms of genres, instrumentation, tonality, etc., (ii) the difficult and time-consuming process of manually adding annotations which —- for most tasks — might also depend on perception and thus require multiple annotators, and (iii) intellectual property laws, restricting the compilation and sharing of music datasets. Many laudable efforts have been made to address (some of) these problems, leading to the release of new datasets or the extension of existing datasets. Nevertheless, the majority of the commonly used datasets for various MIR tasks is still limited in different aspects, which can impact research focus. For example, Benetos et al. pointed out that a large subset of Automatic Music Transcription (AMT) approaches only performed experiments on piano data for which the audio aligned ground truth was easily obtained [1]. This emphasis on piano may lead to models that are strongly biased towards piano-like instruments and cannot be generalized to other melodic instruments.

Automatic Drum Transcription (ADT), a sub-task in AMT that involves the extraction of drum events from audio signals, is also confined to the scope of the existing labeled datasets. Wu observed [30] that most of the ADT related datasets focus on collecting recordings of single drum hits [18, 24] and simple drum sequences without accompaniment [5]. Although these datasets provide the essential ingredients for building basic ADT systems, they cannot properly represent the real-world scenario of drum sounds embedded in a continuous stream of polyphonic audio sources. Thus, they might fail in addressing real-world use cases. The ENST drum dataset [8] partly compensates these drawbacks by offering more realistic and complex drum sequences with accompaniments, however, its size and diversity of music styles are still limited. Previous studies attempt to alleviate these issues through data augmentation [26, 30], but the inherent limitations of the datasets continue to impede the advancement of ADT systems.

One potential solution to addressing this challenge in a scalable way without introducing the additional cost of manual annotations is to explore the usefulness of the vast collection of unlabeled music data; this can be formulated as a *Semi-supervised Learning* problem as defined in the field of machine learning [3]. The general goal of this type of problem is to find the optimal solution given both labeled and unlabeled examples, and it has been applied successfully to different applications such as music genre classification [19], music genre tagging [13], and music emotion recognition [28].

Inspired by the above-mentioned approaches, this paper aims to address the issue of data availability in ADT systems by harnessing the information from the unlabeled music data. Specifically, this paper focuses on improving ADT performance on polyphonic mixtures. The contributions of this paper include: (i) new insights into the viability of using unlabeled music data in ADT tasks, (ii) a general scheme for integrating unlabeled data into ADT and other MIR systems, and (iii) the demonstration of potential improvements of ADT systems using the proposed method. The remainder of the paper is structured as follows: Sect. 2 provides an overview of ADT research and the student-teacher learning paradigm. In Sect. 3, we introduce our approach; the results and discussion are presented in Sect. 4. Sect. 5 provides a summary, conclusion, and directions of future work.

2. RELATED WORK

In the broadest definition of ADT, it can be described as the process of converting drum related audio events, such as drum onset times and playing techniques, into musical representations such as a score or sheet music. To simplify this task while still capturing the essence, most of the existing systems mainly focus on detecting the onset times of *Hi-Hat* (HH), *Snare Drum* (SD) and *Bass Drum* (BD). In many of the early systems, which are summarized by FitzGerald and Paulus [6], the focus was on transcribing signals containing only drum sounds.

Gillet and Richard propose to categorize automatic drum transcription systems into three categories [9]: (i) *segment and classify* [7, 9], which follows the basic pattern recognition approach by segmenting the signals into individual instances, and subsequently classifying each instance with pre-trained classifiers, (ii) *separate and detect* [5, 20, 29], in which the signal is converted into separated activation functions that represent the activities of different drums, followed by a simple peak picking process to identify their corresponding onset times, and (iii) *match and adapt* [31], which identifies the drum events by template matching using a set of pre-trained drum templates and customized distance measures; the templates are iteratively adapted throughout the process. In addition to these three categories, a language-model-based approach using Hidden Markov Models (HMM) [17] and a pattern-matching approach using bar information [23] have also been applied to ADT tasks in previous work.

Following the recent success in deep learning [10], several state-of-the-art ADT systems utilize Deep Neural Networks (DNNs). Specifically, Recurrent Neural Networks (RNNs), a DNN variant modeling the temporal dependency of the input using recurrently connected nodes, have been adopted for this task [22, 25, 26]. Although this method is capable of learning complicated representations of drums from the audio signals, it is extremely demanding in terms of the required amount of training data and computing power. To reach their full potential, DNNs require large amounts of training data; the sizes of currently available datasets appear to be insufficient, as exemplified by the performance degradation in polyphonic mixtures reported in several ADT systems [22, 26, 29]

To overcome the problem of possibly insufficient input data for data-hungry approaches such as DNNs, the idea of utilizing the unlabeled data seems very appealing. Recently, the concept of the student-teacher learning paradigm has emerged as an interesting way of incorporating unlabeled data in the training of DNNs. Originally proposed as a model compression method [2], the basic idea of student-teacher learning is to transfer the knowledge of a large teacher model into a small and concise student model with minimum performance loss; this process, referred by Hinton et al. as "knowledge distillation" [11], is achieved by training the student model with the soft targets generated from the teacher model. In other words, instead of learning from the hard targets (i.e., the ground truth), the student model indirectly acquires the knowledge by mimicking the output from the teacher model. As demonstrated by Li et al. [16], this process can use labeled as well as unlabeled data. Successful applications of this paradigm can be found in tasks such as speech recognition [27] and multilingual models [4], in which superior performances from the student model have also been reported.

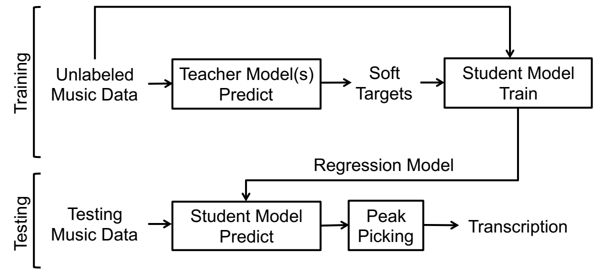


Figure 1. The flowchart of the proposed method

3. METHOD

3.1 System Overview

The processing steps of the proposed method, as shown in Figure 1, can be split into two phases, namely the training and testing phase. In the training phase, the unlabeled music data are passed through the teacher models in order to generate the soft targets. Specifically, these teacher models are ADT systems that will convert the audio signals into drum-related activation functions (i.e., soft targets). The same unlabeled music data and the generated soft targets will then be used to train a student model, which is a regression model that minimizes the differences between its output and the soft targets. In the testing phase, the trained student model predicts the drum activations of the test music data. Finally, a simple peak picking algorithm with an adaptive threshold will be used to identify the drum onset times from each activation function, producing the final transcription output. More elaborate descriptions of the teacher and student models can be found in the following sections.

3.2 Teacher Model

The teacher model used in this paper is the drum transcription system presented by Wu and Lerch [29]. This NMF-based ADT system is chosen for its simplicity, its lack of need for substantial amounts of training data, as well as the adaptability in polyphonic mixtures; it extends the basic NMF model to Partially-Fixed Non-negative Matrix Factorization (PFNMF) by assuming the co-existence of both percussive and harmonic components in the audio signals. More specifically, the template matrix is split into a pre-defined part containing the drum templates which kept fixed and not iteratively updated and a randomly initialized part

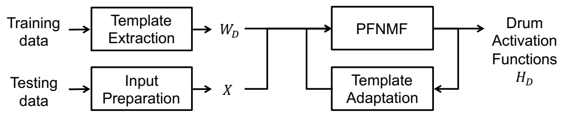


Figure 2. The flowchart of PFNMF [29]

for modeling the remaining harmonic components in the signal. Formally, this can be expressed as

$$X \approx W_{\mathrm{D}} H_{\mathrm{D}} + W_{\mathrm{H}} H_{\mathrm{H}}, \qquad (1)$$

with X being a $m \times n$ magnitude spectrogram matrix with m frequency bins and n blocks, W_{D} and W_{H} representing the drum and harmonic dictionary matrices with a dimensionality of $m \times r_{\mathrm{D}}$ and $m \times r_{\mathrm{H}}$, and H_{D} and H_{H} their corresponding activation matrices with dimensionality of $r_{\mathrm{D}} \times n$ and $r_{\mathrm{H}} \times n$, respectively. r_{D} usually corresponds to the number of drums to detect (e.g., $r_{\mathrm{D}} = 3$ for the detection of HH, BD, and SD), and r_{H} is an user-defined parameter that varies according to the complexity of the target signal.

The basic flowchart of PFNMF is shown in Figure 2. It firstly decomposes the magnitude spectrogram of the polyphonic mixtures with a fixed pre-trained drum dictionary W_{D} and a randomly initialized harmonic dictionary W_{H}. Once the signal is decomposed, the NMF based activation function $H_{\mathrm{D}}(r,:)$ of each individual drum can be extracted, in which $r = \{1, 2, 3\}$ is the instrument index that corresponds to HH, BD, and SD, respectively. These activation functions can be interpreted as the activity level of each instrument over time, and a sharp peak indicates the presence of a single drum hit.

The conversion of the resulting activation functions into the soft targets takes another step of standard min-max scaling across the training data for each instrument; this process scales the soft targets to a numerical range between 0 and 1 and ensures the compatibility between the soft targets and the student model output (see Sect. 3.3). Finally, to introduce diversity into the soft targets, two PFNMF systems are created by initializing the algorithm with two different sets of drum dictionaries, forming an ensemble-like scenario that could potentially lead to better student performance.

3.3 Student Model

The proposed student model is a fully connected, feed-forward DNN with three hidden layers. A neural network is a graphical model that comprises multiple layers of interconnected non-linear units (i.e., neurons). The basic formulation of a neuron can be expressed in Eq. (2)

$$a_k^l = g\left(\sum_{j=1}^{M} W_j a_j^{l-1} + b_j^{l-1}\right), \qquad (2)$$

in which a is the activation of the neuron, W is the weight matrix, b is the bias matrix, l is the layer index, j is the index of input neuron, and k is the index of output neuron; $g()$ is usually a non-linear function such as a *sigmoid*, *tanh* or *relu*. When multiple layers of neurons are stacked, the model creates a non-linear transformation from the input to the output, which allows the model to approximate any arbitrary function with great flexibility.

The architecture of the DNN in this paper is as follows: the input layer contains 1025 neurons that correspond to the size of the input representation. The first hidden layer comprises of 1025 neurons of *tanh* units with Batch Normalization [12]. The second and third hidden layers have 512 and 32 neurons with *relu* units, respectively. Finally, the output layer consists of 3 neurons with *sigmoid* units that represent the activities of three different drums (i.e., HH, SD, and BD). The architecture and type of neurons are selected based on the results of smaller-scale preliminary experiments, and the fully connected layers are chosen for their simplicity and generality. To solve the optimization problem of learning the weights W in a DNN, a stochastic gradient descent based optimization method, Adam [14], is selected as the optimizer. The student neural network is configured as a regressor that minimizes the mean squared error between its output and the soft targets. A mini-batch consisting of 640 instances is used for training, and the early stopping technique is applied to stop the training process when the loss decrease is less than 10^{-6} for three consecutive epochs.

3.4 Implementation

The input representation to both the teacher and student models is the magnitude spectrogram of the Short Time Fourier Transform (STFT) computed using a block size of 2048 and hop size of 512 samples with a Hann window applied to each block. Prior to the calculation of STFT, the audio signals are down-mixed to mono and resampled to a sampling rate of 44.1 kHz. The resulting magnitude spectrogram is a $m \times n$ matrix, in which $m = 1025$ and n equals the number of blocks.

For PFNMF, the authors' open source Matlab implementation [1] is used in our experiments. Since both the unlabeled music data and the test data are polyphonic mixtures, the harmonic rank r_{H} for the PFNMF is set to 50 as suggested [29]. To speed up the process, template adaptation is deactivated. The extraction of the pre-defined (fixed) drum templates takes place on two publicly available drum datasets, namely the SMT-DRUM dataset [5] and 200 drum machines. [2]

Preliminary experiments show that these two sets of templates exhibit capabilities of capturing different types of drum sounds, thus adding diversity to this learning paradigm. The construction of the drum dictionary involves the concatenation of all the spectra and the extraction of the median spectrum for each individual instrument. It should be noted that, since the ENST drum dataset is the main test dataset for evaluation, no single drum hits from ENST are

[1] https://github.com/cwu307/NmfDrumToolbox Last accessed: 2017/04/26

[2] http://www.hexawe.net/mess/200.Drum.Machines Last accessed: 2017/04/26

Experiments				Averaged F-measure		
Role		Method	# Training Data	HH	BD	SD
Teacher	Baseline	PFNMF (SMT)	N/A	0.69	0.80	**0.50**
Teacher	Baseline	PFNMF (200D)	N/A	0.68	0.85	0.48
	Baseline	PFNMF (SMT + 200D)	N/A	0.69	0.83	0.48
Student	Baseline	Linear SGD Regressor	200 * 4 = 800	0.43	0.69	0.43
Student	Proposed	DNN	200 * 4 = 800	**0.78**	**0.86**	0.45

Table 1. A comparison of the averaged F-measures between the proposed method and the baseline methods

used for template extraction in order to ensure the generality of the proposed approach.

The DNN is implemented in Python using Keras [3] with the Tensorflow [4] backend. The parameters of the optimizer are set to default.

To get the final transcription results for evaluation, a standard peak picking method with a signal adaptive median threshold is used [15]. The median threshold $t(n)$ can be computed using Eq. (3):

$$t(n) = \lambda * max(x) + median(x(n), p), \qquad (3)$$

in which x is a vector of novelty function, λ is the offset coefficient relative to the maximum value, p is the order (length) of the median filter, and the n is the block index. All systems are using the peak picking parameters $p = 0.1$ s and $\lambda = 0.12$ as described in [29]. No grid search is performed.

4. EXPERIMENTS

4.1 Dataset Description

The collection of the unlabeled data is a crucial step for ensuring a successful learning process. Generally speaking, the unlabeled dataset should have following attributes: (i) the collection should contain drums whenever possible, (ii) the collection should be diverse in terms of music genres or playing styles, (iii) the collection should contain no duplicates, and (iv) the collection should be as consistent as possible in terms of audio quality. To build a collection that meets the above-mentioned criteria, we compile a list from the Billboard Charts. [5] In particular, we start with an uniform distribution across a set of 4 genres selected for commonly featuring strong drum beats or rhythmic patterns, namely R&B/HipHop, Pop, Rock, and Latin. For this study, 200 songs from each genre has been selected. All the songs are cross-checked for duplicates, and a final list of 800 songs has been compiled and retrieved from Youtube [6] using open source Python library pafy. [7]

All songs are converted into mp3 files with a sampling rate of 44.1 kHz using ffmepg. [8] The source code for constructing the unlabeled music dataset is available online on Github. [9] In order to speed up the process while retaining diversity, only a segment of 30 s from each song is used for training. This segment starts at 30 s into the song in order to avoid possible inactivity at the beginning. Since the same unlabeled data is trained twice with two different sets of soft targets generated from two different teachers, the total duration of the training audio is 800 mins (approximately 13.5 hours), which is significantly larger than any existing drum dataset.

The most popular labeled drum dataset, ENST drum [8], is used as the test set for evaluation. This dataset consists of recordings from three different drummers performing on their own drum kits. The recordings from each drummer contain individual hits, short phrases of drum beats, drum solos, and short excerpts played with accompaniments. Since this paper focuses on ADT in polyphonic mixtures of music, only the *minus one subset* is used for evaluation. This subset has 64 tracks of polyphonic music with a sampling rate of 44.1 kHz. Each track in this subset has a length of approximately 50–70 s with a variety of playing styles. More specifically, the subset contains various drum playing techniques such as ghost notes, flam, and drag, which is close to a real-world setting [30]. The accompaniments are mixed with their corresponding drum tracks using a scaling factor of 1/3 and 2/3 in order to be consistent with prior studies [17, 22, 29]. Only the wet mix recordings of the dataset are used.

4.2 Experiment Setup

The performance of the following systems is evaluated and compared:

(i) PFNMF (SMT): a PFNMF system initialized with a drum dictionary matrix extracted from SMT-DRUM dataset. This baseline system is used as a teacher model to generate the soft targets

(ii) PFNMF (200D): a PFNMF system initialized with a drum dictionary matrix extracted from 200 drum machines dataset. This baseline system is the second teacher model for generating the soft targets

(iii) PFNMF (SMT + 200D): another baseline system by simply taking the averaged activation functions of the above systems as the prediction output

(iv) Linear SGD Regressor: a baseline student model using a simple linear regression with stochastic gradient descent optimization. A Python implementation

[3] https://keras.io Last accessed: 2017/04/27
[4] https://www.tensorflow.org Last accessed: 2017/04/27
[5] http://www.billboard.com/charts Last accessed: 2017/04/25
[6] https://www.youtube.com Last accessed: 2017/04/25
[7] https://pypi.python.org/pypi/pafy Last accessed: 2017/04/25
[8] https://ffmpeg.org/download.html Last accessed: 2017/04/25
[9] https://github.com/cwu307/unlabeledDrumDataset

Experiments				Averaged F-measure		
Role	Method	Genres	# Training Data	HH	BD	SD
Student	DNN	Rock	200 * 1 = 200	0.76	0.83	0.44
Student	DNN	Pop	200 * 1 = 200	**0.78**	**0.85**	0.45
Student	DNN	RnB	200 * 1 = 200	0.74	0.83	**0.48**
Student	DNN	Latin	200 * 1 = 200	**0.78**	0.83	0.44
Student	DNN	All	50 * 4 = 200	0.77	**0.85**	0.45

Table 2. A comparison of different student models trained with unlabeled music data of different genres

of this method from the open source library scikit-learn [10] is used with all parameters set to default values.

(v) DNN: the proposed student model

4.3 Metrics

The evaluation metrics follow the standard calculation of the precision (P), recall (R), and F-measure (F). To be consistent with [9, 22, 29], an onset is considered to be a match with the ground truth if the time deviation between reference and detected onset time is less or equal to 50 ms. It should be noted that some authors use more restrictive settings, compare, for instance, the 30 ms and 20 ms tolerance windows as used in [17] and [26], respectively.

4.4 Results

The experiment results are shown in Table 1. The reported accuracies are the averaged F-measures across all 64 tracks from the ENST minus-one subset. Since the proposed method does not use the ENST drum dataset for training purposes, a three-fold cross validation scheme as reported in [17, 22, 25, 26, 29] is not necessary; this ensures the generality of the proposed method, but prohibits the direct comparison of the results with other publications.

The evaluation results show that both teacher systems PFNMF (SMT) and PFNMF (200D) perform similarly except for BD. This could be due to the discrepancy of the pre-defined drum dictionaries. The 3rd simple baseline system PFNMF (SMT+200D) averaging the teacher outputs gives almost identical performance as the teacher systems. This result shows that a simple combination of the two teacher systems does not result in any improvement. This means either that the performance cannot be improved given the teacher information or that a more sophisticated method is required for combining the outputs. The student baseline system is a simple linear regression model trained using the student-teacher learning paradigm as described in Sect. 3. This baseline serves as a sanity check for the necessity of a complex model such as DNN. As expected, the performance of the linear regression model is the worst among all the evaluated systems, indicating the need of deploying a non-linear model in order to benefit from this training scheme. Finally, the proposed DNN-based student model is actually able to outperform both teachers with higher F-measures for both HH and BD. The results for the SD are somewhat inconclusive; here, one teacher outperforms all other systems. This could imply the similarity between the SD sounds in SMT and ENST dataset, but the inferior performance from the student model still needs further investigation.

Based on these results, another interesting question arises: does music genre play a role in the preparation of unlabeled data? To answer this question, a follow-up experiment has been conducted by training the DNN model with unlabeled data of each individual genre. The experiment results are shown in Table 2. In this experiment, the number of training samples is fixed at 200 in order to eliminate the influence of data size. For the *All* case, 50 songs from each genre are randomly selected. Interestingly, the best performance of different instruments, as highlighted in the table, belongs to different genres. This implies the advantage of having various genres in the training data, for they could potentially complement each other and boost the performance of the student model.

Although the cross-genre model trained on the equally distributed data does not achieve the highest accuracy in every individual instrument, it is still better than majority of the single-genre models and generally well-balanced. Overall, providing diverse unlabeled training data in terms of music genre seems to be beneficial in this learning paradigm.

From all of the above experiment results, the results for HH show the most obvious and consistent improvement over the teacher models. This observation leads to another question: where do these improvements come from? A closer look at the experiment results reveals the strength of the DNN student model. As shown in Table 3, the DNN student model outperforms the teacher models on both precision and recall for HH. The DNN student model also achieves the highest BD precision. Since these improvements in precision are achieved without sacrificing recall, they suggest a reduction in false positives from the student model output. One possible explanation is that the songs presented in the unlabeled music data have a higher agreement on HH sound; this allows the student model to acquire a more consistent internal representation of HH that leads to a more accurate estimation during testing.

It is noticeable that the DNN student model seems to consistently have problems detecting SD. Since the snare drum tends to have larger spectral overlap with the other instruments, it is conceivable that DNN student model will have difficulties learning a robust internal representation for this instrument. A collection of unlabeled data with a stronger presence of snare drum might be possibly able

[10] http://scikit-learn.org Last accessed: 2017/04/25

Method	HH		BD		SD	
	P	R	P	R	P	R
PFNMF (SMT)	0.77	0.69	0.74	**0.91**	**0.67**	**0.49**
PFNMF (200D)	0.75	0.68	0.82	0.90	0.60	**0.49**
DNN	**0.87**	**0.72**	**0.83**	0.89	0.60	0.44

Table 3. A comparison of precision (P) and recall (R) between student and teacher models

to alleviate the problem, however, this issue requires further investigation before any conclusion can be drawn. In general, this deficiency in SD is also consistent with the previous studies [17, 22, 25, 29], where the detection of Snare Drum in polyphonic mixtures has been reported as the most difficult task in ADT. It is also possible that the Snare Drum is for some reason particularly hard to detect in the ENST set that is commonly used for evaluation.

5. CONCLUSION

This paper presents a system for Automatic Drum Transcription based on the student-teacher learning paradigm with the unlabeled music data. The proposed method integrates two NMF-based ADT teacher systems with a DNN-based student model by transferring knowledge using unlabeled music data, and the evaluation results indicate the possibility of obtaining a student model that outperforms the teacher model based on this approach. This result is generally encouraging and demonstrates the great potential of using unlabeled music data in ADT tasks. The experiment results also imply the benefit of having relevant music genres in the unlabeled training data, which could lead to the construction of an improved unlabeled dataset in the future studies. The proposed method has the following advantages: first, the approach allows for complete separation between training and test data, therefore reducing the likelihood of over-fitting and supporting the claim of generality of this approach. Second, the proposed method is able to support data-driven approaches with the need of large amounts of training data given the availability of existing teacher models. Third, the proposed method could not only be easily applied to other ADT systems but also inform data-hungry systems from other transcription tasks or MIR problems in general. Last but not least, this learning scheme has the potential of summarizing multiple complicated teacher systems, providing competitive performance with one concise student model.

The possible future directions of this work are:

(i) *Increasing the number and diversity of teacher systems.* Since the proposed training scheme does not tie to any particular ADT approach, the teacher models can be easily swapped with other ADT expert systems. Intuitively, more teacher models should lead to a more versatile student model. However, the influence of having a more diverse pool of teacher systems still requires further investigation.

(ii) Varying architectures and approaches of the student models. In addition to DNNs, other neural networks architecture may have great potential of achieving better student performance as well. For instance, the RNN based model that incorporates the temporal information could be a good fit in the context of ADT tasks.

(iii) *Evaluating different input representations.* As reported by Cui et al. [4], the student model is able to outperform the teacher model especially when it is trained on the same soft targets but with a stronger input representation. Following this observation, one possible future direction of this work is to investigate the effectiveness of other input representations, such as CQT, Cepstrum, or Wavelet transforms.

(iv) *Evaluating alternative approaches for using unlabeled data.* To fully benefit from the unlabeled data, it is also worth investigating how the proposed method compares to other approaches such as unsupervised feature learning [19].

The presented work represents only a preliminary study of what the authors see as a likely path for the future of training MIR systems as the issue of an insufficient amount of annotated data is likely to get worse with increasing complexity of machine learning systems applied to MIR tasks. Drawing on the vast potential of using existing state-of-the-art MIR-systems as teachers and the overwhelming public availability of unlabeled music data might enable exciting ways of creating new and more powerful MIR systems.

6. REFERENCES

[1] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, July 2013.

[2] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proc. of the International Conference on Knowledge Discovery and Data mining (SIGKDD)*, page 535, 2006.

[3] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 2006.

[4] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, Tom Sercu, Kartik Audhkhasi, Abhinav Sethy, Markus Nussbaum-Thom, and Andrew Rosenberg. Knowledge Distillation Across Ensembles of Multilingual Models for Low-resource Languages. In *Proc. of the International Conference on Acoustics, Speech*

and Signal Processing (ICASSP), pages 4825–4829, 2017.

[5] Christian Dittmar and Daniel Gärtner. Real-time Transcription and Separation of Drum Recording Based on NMF Decomposition. In *Proc. of the International Conference on Digital Audio Effects (DAFX)*, pages 1–8, 2014.

[6] Derry FitzGerald and Jouni Paulus. Unpitched percussion transcription. In *Signal Processing Methods for Music Transcription*. Springer, 2006.

[7] Nicolai Gajhede, Oliver Beck, and Hendrik Purwins. Convolutional Neural Networks with Batch Normalization for Classifying Hi-hat, Snare, and Bass Percussion Sound Samples. In *Proc. of the Audio Mostly*, pages 111–115, 2016.

[8] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2006.

[9] Olivier Gillet and Gaël Richard. Transcription and separation of drum signals from polyphonic music. *IEEE Transactions on Audio, Speech and Language Processing*, 16(3):529–540, March 2008.

[10] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural computation*, 18:1527–1554, 2006.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, pages 1–9, 2015.

[12] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167*, pages 1–11, 2015.

[13] Ping-Keng Jao and Yi-Hsuan Yang. Music Annotation and Retrieval using Unlabeled Exemplars: Correlation and Sparse Codes. *IEEE Signal Processing Letters*, 22(10):1771–1775, 2015.

[14] Diederik P. Kingma and Jimmy Lei Ba. Adam: a Method for Stochastic Optimization. In *Proc. of the International Conference on Learning Representations (ICLR)*, pages 1–15, 2015.

[15] Alexander Lerch. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, 2012.

[16] Jinyu Li, Rui Zhao, Jui Ting Huang, and Yifan Gong. Learning small-size DNN with output-distribution-based criteria. In *Proc. of the Conference of the International Speech Communication Association (INTERSPEECH)*, pages 1910–1914, 2014.

[17] Jouni Paulus and Anssi Klapuri. Drum Sound Detection in Polyphonic Music with Hidden Markov Models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:1–9, 2009.

[18] Matthew Prockup, Erik M. Schmidt, Jeffrey Scott, and Youngmoo E. Kim. Toward Understanding Expressive Percussion Through Content Based Analysis. In *Proc. of the International Society of Music Information Retrieval Conference (ISMIR)*, 2013.

[19] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 759–766, 2007.

[20] Axel Roebel, Jordi Pons, Marco Liuni, and Mathieu Lagrange. On Automatic Drum Transcription Using Non-Negative Matrix Deconvolution and Itakura Saito Divergence. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[21] Markus Schedl, Emilia Gómez, and Julián Urbano. Music Information Retrieval: Recent Developments and Applications. *Foundations and Trends® in Information Retrieval*, 8(2-3):127–261, 2014.

[22] Carl Southall, Ryan Stables, and Jason Hockman. Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[23] Lucas Thompson, Matthias Mauch, and Simon Dixon. Drum Transcription via Classification of Bar-Level Rhythmic Patterns. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[24] Adam R. Tindale, Ajay Kapur, George Tzanetakis, and Ichiro Fujinaga. Retrieval of percussion gestures using timbre classification techniques. In *Proc. of the International Society of Music Information Retrieval Conference (ISMIR)*, pages 541–544, 2004.

[25] Richard. Vogl, Matthias Dorfer, and Peter Knees. Recurrent Neural Networks for Drum Transcription. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 730–736, 2016.

[26] Richard Vogl, Matthias Dorfer, and Peter Knees. Drum Transcription From Polyphonic Music With Recurrent Neural Networks. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 201–205, 2017.

[27] Shinji Watanabe, Takaaki Hori, Jonathan L. Roux, and John R. Hershey. Student-Teacher Network Learning with Enhanced Features. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5275–5279, 2017.

[28] Bin Wu, Erheng Zhong, Derek Hao Hu, Andrew Horner, and Qiang Yang. SMART : Semi-Supervised Music Emotion Recognition with Social Tagging. In *SIAM Conference on Data Mining*, pages 279–287, 2013.

[29] Chih-Wei Wu and Alexander Lerch. Drum transcription using partially fixed non-negative matrix factorization with template adaptation. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 257–263, 2015.

[30] Chih-Wei Wu and Alexander Lerch. On drum playing technique detection in polyphonic mixtures. In *Proc. of International Society for Music Information Retrieval Conference (ISMIR)*, pages 218–224, 2016.

[31] Kazuyoshi Yoshii, Masataka Goto, and Hiroshi G. Okuno. Drum sound recognition for polyphonic audio signals by adaptation and matching of spectrogram templates with harmonic structure suppression. *IEEE Transactions on Audio, Speech and Language Processing*, 15(1):333–345, 2007.

CHORD GENERATION FROM SYMBOLIC MELODY USING BLSTM NETWORKS

Hyungui Lim[1,2], Seungyeon Rhyu[1] and Kyogu Lee[1,2]
[1]Music and Audio Research Group, Graduate School of Convergence Science and Technology
[2]Center for Super Intelligence
Seoul National University, Korea
{goongding7, rsy1026, kglee}@snu.ac.kr

ABSTRACT

Generating a chord progression from a monophonic melody is a challenging problem because a chord progression requires a series of layered notes played simultaneously. This paper presents a novel method of generating chord sequences from a symbolic melody using bidirectional long short-term memory (BLSTM) networks trained on a lead sheet database. To this end, a group of feature vectors composed of 12 semitones is extracted from the notes in each bar of monophonic melodies. In order to ensure that the data shares uniform key and duration characteristics, the key and the time signatures of the vectors are normalized. The BLSTM networks then learn from the data to incorporate the temporal dependencies to produce a chord progression. Both quantitative and qualitative evaluations are conducted by comparing the proposed method with the conventional HMM and DNN-HMM based approaches. Proposed model achieves 23.8% and 11.4% performance increase from the other models, respectively. User studies further confirm that the chord sequences generated by the proposed method are preferred by listeners.

1. INTRODUCTION

Generating chords from melodies is an artistic process for musicians, which requires knowledge of chord progression and tonal harmony. While it plays an important role in music composition studies, the implementation of its process can be difficult especially for individuals who do not have prior experience or domain knowledge in musical studies. For this reason, the chord generation process often serves as an obstacle for novices who try to compose music based on a melody.

To overcome this limitation, automatic chord generation systems have been implemented based on machine learning methods [1, 2]. One of the most popular approaches for this task is probabilistic modelling, which commonly applies the hidden Markov model (HMM). A single-HMM is used with 12-semitone vectors of melody as observations and corresponding chords as hidden states [3, 4]. Allan and Williams trained a first-order HMM which learns from pieces composed by Bach, to generate chorale harmonies [5]. A more complex method is presented by Raczyński et al. [6], using time-varying tonalities and bigrams as observations with melody variables. In addition, a multi-level graphical model using tree structures and HMM is proposed by Paiement et al. [7]. Their model generates chord progressions based on the root note progression predicted from a melodic sequence. Forsyth and Bello [8] also introduced a MIDI based harmonic accompaniment system using a finite state transducer (FST).

Although the HMM has been successfully used for various tasks, it has several drawbacks. According to one of the assumptions of the Markov model, observations occur independently of their neighbors, depending only on the current state. Moreover, the current state of a Markov chain is only affected by its previous state. These drawbacks are also observable in chord generation from melody tasks because long-term dependencies exist in chord progressions and melodic sequences of Western tonal music [6].

Meanwhile, deep learning based approaches have recently shown great improvements in machine learning tasks of large datasets. Especially for temporal sequences, recurrent neural networks (RNN) and long short term memory (LSTM) networks have proven to be more powerful models than HMM in the field of handwriting recognition [9], speech recognition [10], and emotion recognition [11]. Nowadays, even music generation researches have increasingly adapted RNN/LSTM models in two major stream – one that aims to generate complete music sequences [12, 13], and the other which concentrates on generating music components such as melody, chord and drum sequence [14, 15]. We attempt an extended approach to the latter stream by implementing a chord generation system with a melody input.

In this paper, we implement a chord generation algorithm based on bidirectional LSTM (BLSTM) and evaluate its performance on reflecting temporal dependencies on melody/chord progressions by comparing with two HMM-based methods: a simple HMM, and deep neural networks-HMM (DNN-HMM). We then present the quantitative analysis and the accuracy results of the three models. We also describe the qualitative results based on subjective ratings provided by 25 non-musicians.

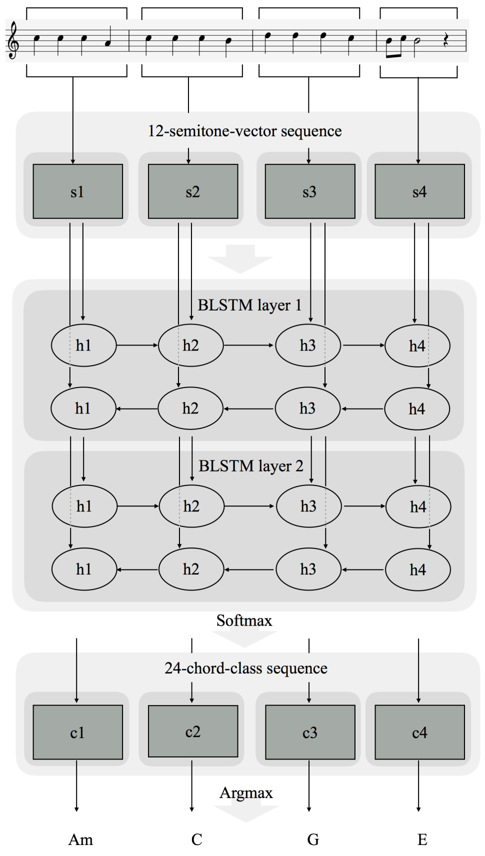


Figure 1. The overview of proposed system

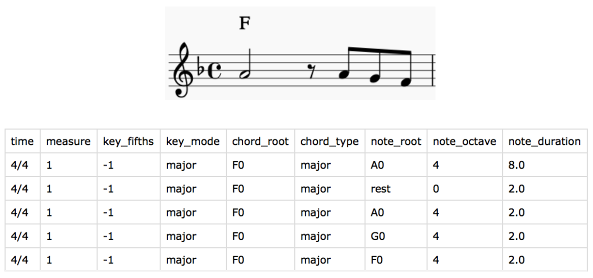
The remainder of the paper is organized as follow. In Section 2, we explain the preprocessing step and the details of the machine learning methods we apply. Section 3 describes the experimental setup for evaluating the proposed approach. The experimental results are presented in Section 4, with additional discussions. Finally, we draw a conclusion followed by limitations and future works in Section 5.

2. METHODOLOGY

The method proposed in this paper can be divided into two main parts. The first part is a preprocessing procedure to extract input/output features from lead sheets. The other part consists of model training and a chord generation processes. We apply BLSTM networks for the proposed model and two types of HMM for the comparable models. The overall framework of our proposed method is shown in Figure 1.

2.1 Preprocessing

To extract appropriate features for this task, we first collect musical features such as time signature, measure (bar), key {fifths, mode}, chord {root, type} and note {root, octave, duration} from the lead sheets. These features are then represented in a matrix by concatenating rows, which respectively represent the musical features of a single note



time	measure	key_fifths	key_mode	chord_root	chord_type	note_root	note_octave	note_duration
4/4	1	-1	major	F0	major	A0	4	8.0
4/4	1	-1	major	F0	major	rest	0	2.0
4/4	1	-1	major	F0	major	A0	4	2.0
4/4	1	-1	major	F0	major	G0	4	2.0
4/4	1	-1	major	F0	major	F0	4	2.0

Figure 2. An example of extracted data from a single bar.

as shown in Figure 2.

The generated data is then preprocessed in order to make an acceptable relation between melody input and chord output. All songs are in major key in the database and are transposed to C major key for data consistency. In other words, all roots of chords and notes are shifted to C major key to normalize different characteristics of melodies and chords in different songs.

Each song contains a time signature, which has a variety of meters such as 4/4, 3/4, 6/8, etc. The variety in time signature causes the imbalance of total note durations in a bar among different songs, so note durations are normalized by multiplying them with the reciprocal number of each time signature. After that, every note in a bar is stored into 12 semitone classes, without the octave information. Each class consists of a single value that accumulates the duration of the corresponding semitone in the bar.

Since the total number of chord types is quite large, if all of these chord types exist as independent classes, then each chord may not have enough samples. For such reason, all types of chords are mapped into one of two primary triads: major and minor. Each chord is represented with a binary 24-dimensional class to indicate the 24 major/minor chords.

2.2 BLSTM Networks

Recurrent neural networks (RNN) is a deep learning model, which learns complex networks not only by reconstructing the input features in a nonlinear process, but also by using the parameters of previous states in its hidden layer. A concept of "time step" exists in RNN, which is able to control the number of feedbacks on a recurrent process. This property enables the model to incorporate temporal dependencies by storing the past information in its internal memory, in contrast to a simple feedforward deep neural networks (DNN).

Despite such advantages of RNN models, there still exist problems regarding the long-term dependency. This is caused by vanishing gradient during the back propagation through time (BPTT) [16]. In the process of calculating the gradient of the loss function, the error between the estimated value and the actual value diminishes as the number of hidden layers increases. Thus, we instead use long short-term memory (LSTM) layers, which improve the limitation of storing long-term history with three multiplicative gates [17].

Generally, chords and melodies are formed in a sequential order, which is affected by both the previous and next order. Based on this, we can predict that if we reverse the lead sheet and train the musical progressions, a meaningful sequential context similar to the originals will appear. Hence, we apply a BLSTM so that the network can reflect musical context not only in forward but also in backward directions.

As shown in Figure 1, the input semitone vectors from each bar enter the network sequentially during the time step (i.e. a fixed number of bars) and emit the corresponding output chord classes in the same order. This is possible because the hidden layer in the network returns the output for each input. In order to train this sequence of multiple bars, we reconstruct our dataset by applying the window with the size of the time step and overlapping the window with the hop size of one bar. Each window, composed of multiple bars, is then used as a sample to train the network.

For our model, we build a time distributed input layer with 12 units, which represents the sequence of semitone vectors, 2 hidden layers with 128 BLSTM units, and a time distributed output layer with 24 units, which represents the sequence of chord classes. We empirically choose the number of hidden layers and units that yield the best result. We use hyperbolic tangent activation function for the hidden layers to reconstruct the features in a nonlinear process. We then apply the softmax function for the output layer to generate values corresponding to the probability of each class. Dropout is also employed with a rate of 0.2 on all hidden layers to prevent overfitting. We use mini-batch gradient descent with categorical cross entropy as the cost function and Adam as the optimizer. In addition, for the model training process, we use a batch size of 512 and early stopping for 10 epoch patience.

2.3 Hidden Markov Model

We apply two types of supervised HMM as baseline models. First is a simple HMM which is a generative model and the other is hybrid deep neural network–HMM (DNN-HMM) which is a sequence-discriminative model [18].

2.3.1 Simple HMM

The simple HMM consists of three parameters: initial state distribution, transition probability and emission probability. In our case, the initial state distribution is the histogram of each chord in our train set. The transition probability is computed using the bigram of chord transition and it is assumed to follow the rule of general first-order Markov chains. A higher-order transition probability is not taken into account because the fixed length of an input bar in our task is not long enough. The emission probability is determined by a multinomial distribution of semitone observations from each chord class.

Once the parameters are learned, the model can generate a sequence of hidden chord states from a melody with three steps. First, the probabilities of 24 chord classes in each bar are determined by the melody distribution in each bar. As mentioned above, the simple HMM is a generative model. Hence, it uses not only the emission probability but also a class prior to calculate posterior probability with the Bayes rule. We define the class prior same as the initial probability, which is the histogram of each chord. Secondly, in order to reflect sequential effects, transition probability is applied to adjust the probabilities of the chord classes. In case of the first chord state, since there is no previous state to consider the transition, the initial probability is applied instead. After that, a Viterbi decoding algorithm is implemented to find the optimal chord sequence that is most likely to match along with the observed melody sequence [19].

2.3.2 DNN-HMM

The hybrid DNN-HMM is a popular model in the field of speech recognition [20]. It is a sequence-discriminative model, which adapts the advantage of sequential modeling method of HMM, but does not require the class prior and the emission probability to get posterior probability. DNN makes it possible because the probability result from a softmax output layer can be assumed as a posterior probability. Then the two of HMM parameters - initial state distribution and transition probability – are applied identically with the simple HMM to employ the Viterbi decoding algorithm.

We build an input layer with 12 units, 3 hidden layers with 128 units that are all identical and an output layer with 24 units. We use hyperbolic tangent activation function for the hidden layer and softmax for the output layer. Other features such as dropout, loss function, optimizer and batch size are applied in the same settings of BLSTM.

3. EXPERIMENTS

In this section, we first introduce our dataset, which is parsed from digital lead sheets. Then we present the experimental setup for evaluating the performance of chord generation models. We conduct both quantitative and qualitative evaluations for this task.

3.1 Dataset

We use the lead sheet database provided by Wikifonia.org, which was a public lead sheet repository. The site unfortunately stopped service in 2013, but some of the data, which consists of 5,533 Western music lead sheets in MusicXML format, including rock, pop, country, jazz, folk, R&B, children's song, etc., was obtained before the termination and we extracted features from the data for only academic purpose. From the obtained database, we collect 2,252 lead sheets, which are all in major key, and the majority of the bars in the lead sheets have a single chord per bar. If a bar consists of two or more chords, we choose the first chord in the bar. Then we extract musical features and convert them to a CSV format (see Section 2.1). The set is split into two sets – a training set of 1802 songs, which consists of 72,418 bars and a test set of 450 songs, which consists of 17,768 bars. Since musical features in this dataset can be useful for not only chord generation but also for other kinds of symbolic music tasks, the dataset is shared on our website (`http://marg.snu.ac.kr/chord_generation/`) for public access.

3.2 Quantitative Evaluation

We perform a quantitative analysis by comparing the accuracies of chord estimation from each model using the test set. The accuracy is calculated by counting the number of matching samples between the predicted and the true chords and by dividing it by the total number of samples. We mainly apply a 4-bar melody input for our task, but also experiment with 8-, 12- and 16-bar inputs to analyze the influence on the length of a melody sequence.

Determining the "right" chord is a difficult process because chord selection can vary among people based on their musical styles and tastes. However, the aforementioned accuracy calculation is often used to evaluate the capability of incorporating the long-term dependency in the musical progression [6, 8]. Therefore, we use it for measuring which model reflects the relationship between chord and melody most adequately.

3.3 Qualitative Evaluation

As mentioned above, there is a limit to evaluate the model performance only by a quantitative analysis. Thus, we also conduct qualitative evaluation based on subjective rating from actual user. This assessment allows us to determine the validity of each model by comparing how the chords generated from different models are perceived by actual users. For the experiment, we collect eighteen 4-bar-length melodies from lead sheets of thirteen K-pop songs and five Western pop songs. Every melodic sequence is converted into a vector of 12 semitones as described in Section 2.1. HMM, DNN-HMM, and BLSTM then generate chord sequences from each vector. Those sequences are evaluated by 25 musically untrained participants (13 males and 12 females) through a web-based survey.

The participants complete 18 sets of surveys in their own pace. At the beginning of each set, participants listen to a melody. After that, participants listen to the four types of chord progressions, including the one from the original song, along with the melody. Participants are asked to rate each chord progression on a five-point scale (1 – 'not appropriate'; 5 – 'very appropriate'). At the end of each set, participants also are asked to answer a question whether they have pre-existing familiarity with the original songs. The audio samples used for experiment are available on our website.

# of Bars	HMM (%)	DNN-HMM (%)	BLSTM (%)
4	40.33	45.02	50.55
8	40.43	44.82	50.32
12	40.41	44.95	49.23
16	40.45	44.68	49.90
Average	40.41	44.87	50.00

Table 1. Chord prediction performance using different number of input bar.

4. RESULTS

4.1 Chord Prediction Performance

Table 1 presents the accuracy results of three models for four instances of different bar lengths. The results show that the BLSTM method achieves the best performance on the test set followed by DNN-HMM and HMM. According to the average scores of models, BLSTM has 23.8% and 11.4% performance increase from the HMM and DNN-HMM, respectively. The results also demonstrate that the number of input bars is not an important factor affecting the accuracy for all models since they don't show obvious linear variations.

To examine the quality of predicted chords from each model more in depth, we compute the results of each model into a confusion matrix. This allows us to easily analyze the results through visualization. We normalize the matrix with the number of samples in respective chords so that each row represents the distribution of predicted chords on each true chord class. In Figure 3, we display this normalized confusion matrix of each model.

A number of noteworthy findings from each matrix are observed. First, HMM yields a skewed result that shows severe misclassification of chords especially on C, F and G as shown in Figure 3(a). We hypothesize this is resulted from the lack of complexity of the model. Emission probability, one of the parameters of the model, does not properly capture the accurate correlation between the chords and corresponding melodies. Moreover, the fact that the training data contains more frequent occurrences of C, F and G chords (over 60% in total samples) reduced the accuracy of the HMM model which uses the prior probability to obtain the posterior as mentioned in Section 2.3.1. Lastly, a noticeable bias in transition matrix moving to C chord also seems to lower the precision of the model.

The result of DNN-HMM is similar to HMM but the skewness on C chord spreads out little bit to F and G chords. Despite our initial expectation that the DNN would perform better since it is a discriminative model that calculates posterior directly, still many misclassifications on three chords exist as shown in Figure 3(b). To find the reasoning behind this observation, we test simple DNN with 1-bar input without the sequential parameter of HMM. The accuracy is higher than DNN-HMM (46.93%) and the confusion matrix produces more diagonal elements as shown in Figure 4. This finding supports that the transition

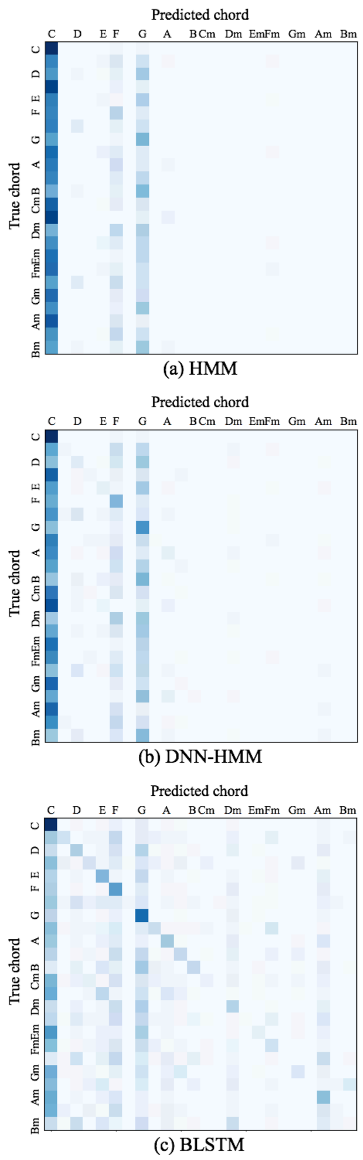


Figure 3. Normalized confusion matrix of HMM(a), DNN-HMM(b), and BLSTM(c) using 4-bar melody input.

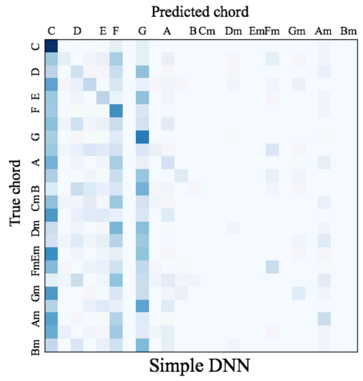


Figure 4. Normalized confusion matrix of simple DNN using single bar melody input.

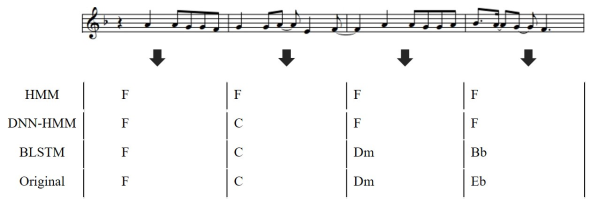


Figure 5. An example of generated chord progressions from three different models and the original progression.

probability of HMM forces the model to generate limited classes and also that the model is not adequate to train various chord progressions.

In contrast to the HMM based method, the confusion matrix of the BLSTM shows a less skewed distribution and clearer diagonal elements as shown in Figure 3(c). BLSTM has much more complex parameters in hidden layers, which train the sequential information of both melodies and chords. We believe this property makes the performance better compared to the others.

4.2 User Preference

In the user subjective test, evaluation scores are obtained from 450 sets (18 sets x 25 participants). Each set contains chord sequences from HMM, DNN-HMM, and BLSTM. An original chord sequence is also included for relative comparison of the generated results to the original. These four chord sequences are evaluated as described in Section 3.3. Figure 5 shows the example of melody and chord sequences which is used in the user test and more examples are available to listen on our website.

The average score of each model is shown in Figure 6. The original chord progression is preferred the most followed by BLSTM, DNN-HMM, and HMM. To investigate whether differences on scores between the results are critical, we conduct one-way repeated measure ANOVA setting each model as a variable. The result shows that at least one out of four scores is significantly different from the others. ($F(3, 1772) = 310$, $p < 0.001$). We then conduct a pairwise t-test with Bonferroni correction on the mean scores between each pair of models for a post-hoc analysis. As a result, differences between all pairs are proven to be significant ($p < 0.01$). Therefore, it can be concluded that the BLSTM produces the most satisfying chord sequences among the other computational models but it produces less satisfying results than the original. Moreover, since the difference between BLSTM and DNN-HMM is bigger than other pairs, it seems there is a big quality difference between them.

To verify our hypothesis that having familiarity with the original song affects the result we perform a further analysis. We separate 450 evaluation sets into two, 248 sets marked as known and the rest as unknown, and conduct further analysis. A simple comparison of those two sets based on the evaluation scores shows that awareness of the songs does not affect the preference rank of the models. We also perform one-way repeated measure

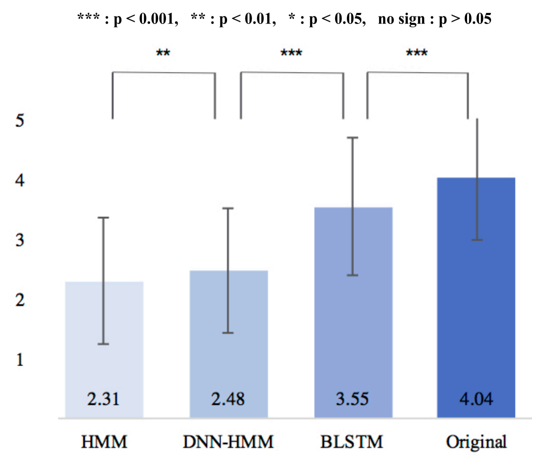


Figure 6. Mean score of subjective evaluation of each model.

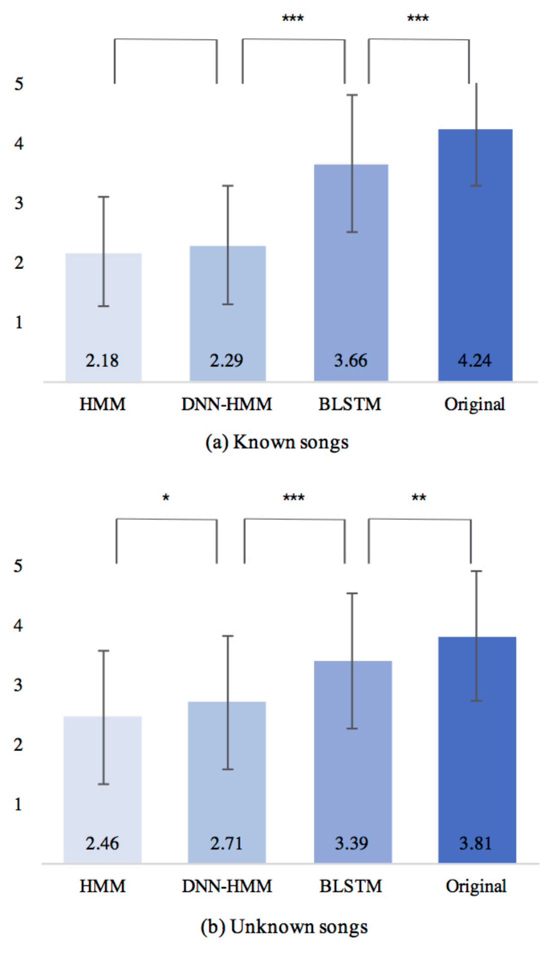


Figure 7. Mean score of subjective evaluation for a group of known songs (a), and of unknown songs (b).

ANOVA for each group of awareness (group of known songs: $F(3, 964) = 286, p < 0.001$; group of unknown songs: $F(3, 780) = 72, p < 0.001$) and pairwise t-test with Bonferroni correction. The results are presented in Figure 7. As shown in the figure, when songs are unknown, the preference for HMM based models increases while it decreases for BLSTM generated and original chords. A plausible explanation for this observation can be that when the listener knows the song, he/she is more perceptive of the monotonous chord sequences generated from HMM and DNN-HMM which tend to produce more of C, F and G than other chords. However, when the listener does not know the song, he/she is less aware of the monotonous progression of the chords and tend to give more generous scores to those two models. For BLSTM, the result is the opposite. Listeners who are more used to the dynamic chord progression of the original song tend to give relatively higher scores to BLSTM than to HMM based methods probably because BLSTM often generates a more diverse chord sequences. On the other hand, when the songs are unknown, relative preference towards both BLSTM and the original chords is less strong. The reduced gap among four different options when the songs are unknown may be explained by the assumption that when the songs are not familiar, all four options are relatively equally acceptable to the listeners. Regardless of the difference in the results, however, BLSTM is preferred over the other two models in both cases.

5. CONCLUSIONS

We have introduced a novel approach for generating a chord sequence from symbolic melody using neural network models. The result shows that BLSTM achieves the best performance followed by DNN-HMM and HMM. Therefore, the recurrent layer of BLSTM is more appropriate to model the relationship between melody and chord than HMM based sequential methods.

Our work can be further improved by modifying data extracting and preprocessing steps. First, since the lead sheets used in this study have one chord in each bar, the task is constrained to one-chord generation for each bar. Since actual music usually contains a lot of bars with multiple chords, additional extraction process is needed to allow the model to generate multiple chords per bar. Secondly, in the preprocessing step, all chords are mapped into only 24 classes of major and minor. Thus, further chord classes such as maj7 and min7 need to be included for performance improvement. Lastly, our input feature vectors consist of 12 semitones by accumulating the melody notes in each bar, so the sequential information of melodies in each bar disappears in this step. Thus, another feature-preprocessing step may be needed not to omit the information, which can be a crucial factor in the future work. We hope that more researches will be done through our published data to overcome the limitations as well as to further develop of this task.

6. ACKNOWLEDGEMENTS

This work was supported by Kakao Corp. and Kakao Brain Corp.

7. REFERENCES

[1] E. C. Lee and M. W. Park: “Music Chord Recommendation of Self Composed Melodic Lines for Making Instrumental Sound,” *Multimedia Tools and Applications*, pp. 1-17, 2016.

[2] S. D. You and P. Liu: “Automatic Chord Generation System Using Basic Music Theory and Genetic Algorithm,” *Proceedings of the IEEE Conference on Consumer Electronics (ICCE)*, pp. 1-2, 2016.

[3] I. Simon, D. Morris, and S. Basu: "MySong: Automatic Accompaniment Generation for Vocal Melodies," *Proceedings of the Special Interest Group on Computer-Human Interaction (SIGCHI) Conference on Human Factors in Computing Systems,* pp. 725-734, 2008.

[4] H. Lee and J. Jang: "i-Ring: A System for Humming Transcription and Chord Generation," *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, Vol. 2, pp. 1031–1034, 2004.

[5] M. Allan and C. Williams: “Harmonizing Chorales by Probabilistic Inference,” *Advances in Neural Information Processing Systems*, Vol. 17, pp. 25-32, 2005.

[6] S. A. Raczyński, S. Fukayama, and E. Vincent: “Melody Harmonization with Interpolated Probabilistic Models,” *Journal of New Music Research*, Vol. 42, No. 3, pp.223-235, 2013.

[7] J. Paiement, D. Eck, and S. Bengio: "Probabilistic Melodic Harmonization," *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, pp. 218-229, 2006.

[8] J. P. Forsyth and J. P. Bello: "Generating Musical Accompaniment Using Finite State Transducers," *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13)*, 2013.

[9] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidlhuber: “A Novel Connectionist System for Unconstrained Handwriting Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 5, pp. 855-868, 2009.

[10] H. Sak, A. Senior, and F. Beaufays: “Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition,” *CoRR arXiv: 1402.1128*, 2014.

[11] M. Wöllmer, A. Metallinou, F. Eyben, B. Schuller, and S. Narayanan: “Context-Sensitive Multimodal Emotion Recognition from Speech and Facial Expression Using Bidirectional LSTM Modeling,” *Interspeech*, pp. 2362-2365, 2010.

[12] I. Liu and B. Ramakrishnan: “Bach in 2014: Music Composition with Recurrent Neural Network,” *CoRR arXiv: 1412.3191*, 2014.

[13] D. D. Johnson: “Generating Polyphonic Music Using Tied Parallel Networks,” *International Conference on Evolutionary and Biologically Inspired Music and Art*, pp. 128-143, 2017.

[14] A. E. Coca, D. C. Corrêa, and L. Zhao: “Computer-aided Music Composition with LSTM Neural Network and Chaotic Inspiration,” *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 1-7, 2013.

[15] K. Choi, G. Fazekas, and M. Sandler: “Text-based LSTM Networks for Automatic Music Composition,” *CoRR arXiv: 1604.05358*, 2016.

[16] F. A. Gers, J. Schmidhuber, and F. Cummins: “Learning to Forget: Continual Prediction with LSTM,” *Neural Computation,* Vol. 12, pp. 2451-2471, 2000.

[17] S. Hochreiter and J. Schmidhuber: “Long Short-Term Memory,” *Neural Computation*, Vol. 9, No. 8, pp. 1735-1780, 1997.

[18] K. Veselý, A. Ghoshal, L. Burget, and D. Povey: “Sequence-Discriminative Training of Deep Neural Networks,” *Interspeech*, pp. 2345-2349, 2013.

[19] K. Lee and M. Slaney: “Automatic Chord Recognition from Audio Using a Hmm with Supervised Learning,” *Proceedings of the 7th International Conference on Music Information Retrieval*, pp. 133–137, 2006.

[20] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury: “Deep Neural Networks for Acoustic Modeling in Speech Recognition,” *IEEE Signal Processing Magazine*, Vol. 29, No. 6, pp. 82-97, 2012.

COVER SONG IDENTIFICATION WITH METRIC LEARNING USING DISTANCE AS A FEATURE

Hoon Heo[1] **Hyunwoo J. Kim**[2] **Wan Soo Kim**[1] **Kyogu Lee**[1]
[1] Music and Audio Research Group, Seoul National University, Republic of Korea
[2] Department of Computer Sciences, University of Wisconsin–Madison, USA
cubist04@snu.ac.kr, hwkim@cs.wisc.edu, wansookim@snu.ac.kr, kglee@snu.ac.kr

ABSTRACT

Most of cover song identification algorithms are based on the pairwise (dis)similarity between two songs which are represented by harmonic features such as chroma, and therefore the choice of a distance measure and a feature has a significant impact on performance. Furthermore, since the similarity measure is query-dependent, it cannot represent an absolute distance measure. In this paper, we present a novel approach to tackle the cover song identification problem from a new perspective. We first construct a set of core songs, and represent each song in a high-dimensional space where each dimension indicates the pairwise distance between the given song and the other in the pre-defined core set. There are several advantages to this. First, using a number of reference songs in the core set, we make the most of relative distances to many other songs. Second, as all songs are transformed into the same high-dimensional space, kernel methods and metric learning are exploited for distance computation. Third, our approach does not depend on the computation method for the pairwise distance, and thus can use any existing algorithms. Experimental results confirm that the proposed approach achieved a large performance gain compared to the state-of-the-art methods.

1. INTRODUCTION

A cover song, or simply cover, is a new version of existing music that is recorded or arranged by another musician. A cover reuses the melody and lyrics of the original song, but it is performed with new singers and instruments. The other musical factors such as key, rhythm, and genre can be reinterpreted by the new artist. Since the copyright of composition and lyrics of the cover still belongs to the author of the original song, releasing a cover song without permission of the original author may cause a legal conflict. Another case is music sampling, which is the act of process that reuses a snippet of existing sound recordings. The sampling is widely considered to be a technique for creating music today, but licensing that the original creator authorizes its reuse is a legal requirement. Cover song identification is a task that aims to measure the similarity between two songs. It can be used to prevent the infringement of copyright, and also to be an objective reference in case of conflict.

For a decade, many approaches for cover song identification have been proposed. Humans generally recognize the cover through the melodic or lyric similarity, but separation of the predominant melody from a mixed music signal is still not at a reliable level, and extraction of the lyrics can be attempted only if it is clearly separated. For this reason, most of the existing algorithms use the harmonic progression represented by an acoustic feature such as chroma [6], and measure the similarity in the features to determine the distance between two songs.

Cover song identification generally consists of two main stages: feature extraction and distance calculation. In most related works, chroma or harmonic pitch class profile (HPCP) are usually chosen, as well as its variants such as CENS [9], CRP [8], and MPLPLC [2]. It is reported that the abstraction of the chroma-like feature to focus on the chord progression rather than instantaneous note changes improves the identification performance [2, 15]. In early days, the feature was synchronized with the beat to take into account the covers with different tempo [4]. However, since the error in beat tracking degrades the performance and the tempo change is usually not extreme, the hop size with a fixed length is recently preferred [14]. Besides, two-dimensional Fourier transform magnitude (2DFTM) of the chroma feature is applicable for large-scale cover song identification [1]. The 2DFTM is key-invariant and thus does not require any preprocessing for key transposition. Also, regardless of the duration of the song, its fixed size has the advantage of keeping the locality.

In respect to the distance calculation, an early approach finds the best-correlated point using cross-correlation of the beat-synchronous chroma [4]. The next popular approach is based on dynamic time warping (DTW), which can be sensitive to tempo changes even when the hop size is fixed [14]. This approach uses the overall distance after aligning over the whole region of the two given songs. On the other hand, a more recent approach called similarity matrix profile (SiMPle) yields a high similarity when many local similar regions are found [15].

The conventional approaches described above calculate

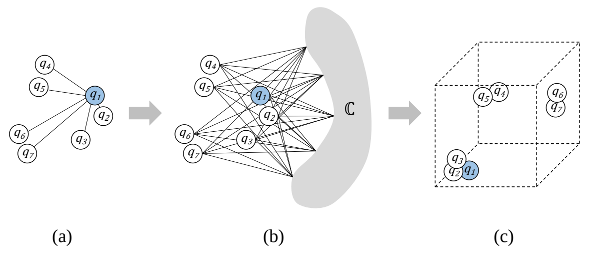


Figure 1. (a) The original distance between a query q_1 and the other songs. (b) The distance between each query and the core set $\mathbb{C}$. (c) New representation of songs in the $|\mathbb{C}|$-space.

the distance between a query and the songs to be compared, and determine that the song with the nearest distance is highly likely to be a cover. Since this process is separate from each query, the result from "another version of the same cover" cannot be taken into account. If it is possible, songs with different lengths can be represented in the same space. Furthermore, if similar/dissimilar song pairs are known, the metric to measure the song distance can be optimized, rather than using the Euclidean distance. Instead of taking the distance matrix directly to rank the similarity, we first perform a nonlinear transformation using kernel principal component analysis (KPCA) to rearrange each song in the high-dimensional space. Next, the distance metric is learned from song pairs in the new representation and their labels. We select "core songs" with diverse musical properties and use them for both embedding and training. In summary, our approach assumes that the distance between the core set and each song can be a discriminating feature to easily group the same covers. The conceptual illustration of this new representation is shown in Figure 1.

The goal of this paper is to examine whether the distance metric learning can be effective to retrieve the similarity between songs. Also, this paper aims to achieve the best performance in cover song identification by applying the metric learning to the distance matrix generated by existing algorithms. Currently, MIREX hosts an annual task for cover song identification, but the dataset is not publicly available. In the later section, we report a performance comparison using our own dataset with the same specification as that of the MIREX.

The rest of this paper is organized as follows. Section 2 defines some important terms throughout this paper, and summarizes three popular algorithms for measuring the distance between songs. In section 3, we describe the technical method for better representation of songs and metric learning. After that, the experimental setup and results are presented in section 4. Finally, the conclusions of this paper are drawn in section 5.

2. DISTANCE MATRIX

The distance matrix is defined by a two-dimensional matrix that contains the pairwise distances for all possible combinations of two songs. The range of distance may vary depending on the algorithm, but it should be low between songs belonging to the same cover group, and should be high if they are not associated.

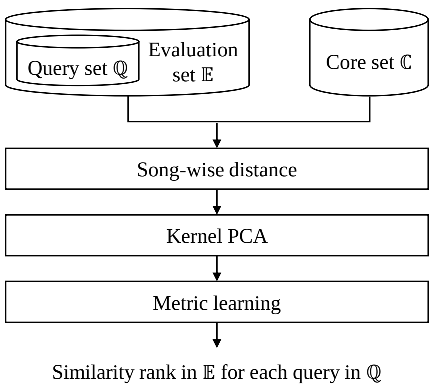


Figure 2. Block diagram of the proposed method.

We define three sets of songs as follows:

- Query set ($\mathbb{Q}$): A set of songs to be a query for identification. Each cover group consists of the same number of versions.
- Evaluation set ($\mathbb{E}$): A set for performance evaluation which includes the query set $\mathbb{Q}$. The remainders are "confusing songs" that are not associated with any cover groups.
- Core set ($\mathbb{C}$): An additional set of songs for embedding and training in the proposed method. It is good to select songs in the core set with diverse musical styles (i.e. genre, tempo, instruments).

Among these sets, $\mathbb{Q} \subset \mathbb{E}$ and $\mathbb{E} \cap \mathbb{C} = \varnothing$ should be satisfied.

The distance matrix is a square matrix calculated from all the songs in the three sets. We employed three algorithms for measuring the song-wise distance: dynamic time warping (DTW), Smith–Waterman algorithm, and similarity matrix profile (SiMPle). In the following subsections, we give a brief overview of each algorithm to construct the distance matrix.

2.1 Dynamic Time Warping

DTW performs dynamic programming to retrieve the optimal path that minimizes the warping cost. Given a sequence A of length n and a sequence B of length m, it constructs an n-by-m matrix that contains the Euclidean distance $\delta_{i,j}$ between both sequences at two time instances i and j. The cumulative distance $\gamma_{i,j}$ is the sum of the distance in the current point and the minimum cumulative distance from the three adjacent points,

$$\gamma_{i,j} = \delta_{i,j} + \min\left(\gamma_{i-1,j-1}, \gamma_{i-1,j}, \gamma_{i,j-1}\right). \quad (1)$$

The overall distance between two sequences A and B is determined by the cumulative distance at the end of the path,

$$d_{A,B} = \gamma_{n,m}. \tag{2}$$

To prevent unrealistic warping and reduce the number of paths to consider, DTW can be implemented with global and local constraints. The two popular global constraints are Sakoe–Chiba band [11] and Itakura parallelogram [5]. On the other hand, the local constraints allows deviations of the double or half the original tempo by using warpings $(i-1, i-1)$, $(i-2, j-1)$, and $(i-1, j-2)$ [10].

2.2 Smith–Waterman Algorithm

Similar to DTW, Smith–Waterman algorithm performs dynamic programming to find the optimal path that maximizes the similarity score between two sequences [16]. The main difference to the classic DTW is that the optimal path is produced locally. That is, it is not necessary that the path with the maximum similarity covers the whole sequence. Given a sequence A of length n and a sequence B of length m, it constructs an $(n+1)$-by-$(m+1)$ scoring matrix H. The first row and column are initialized with 0. The recursion formula to fill the rest of the scoring matrix is,

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ \max_{k\geq 1}\{H_{i-k,j} - W_k\}, \\ \max_{l\geq 1}\{H_{i,j-l} - W_l\}, \\ 0 \end{cases} \tag{3}$$

where $s(a_i, b_j)$ is the similarity score between ith element of A and jth element of B, and W_n is the penalty of a gap with length n. The overall similarity of the Smith–Waterman algorithm is defined as the maximum value on the scoring matrix.

2.3 Similarity Matrix Profile

Similarity matrix profile (SiMPle) efficiently evaluates similarities between songs based on subsequence similarity joins in the features [15]. For a time-frequency representation A of length m and B of length n, SiMPle identifies the nearest neighbor of each continuous subsets in A from all continuous subsets in B. Euclidean distance between the subset of A with time index i and the subset of B with time index j, $d_{i,j}$, is calculated using MASS (Mueen's Algorithm for Similarity Search), the fastest known algorithm for distance vector computation [7].

$$d_{i,j} = \text{MASS}(A[i], B[j]) \tag{4}$$

SiMPle P_i is obtained by choosing the minimum value in the distance between a subset of A and each subset of B.

$$P_i = \min(d_{i,1}, d_{i,2}, \cdots, d_{i,n}) \tag{5}$$

The overall distance between two sequences A and B is defined as the median value of SiMPle [15].

$$d_{A,B} = \text{median}(P_i) \tag{6}$$

Note that SiMPle is not a symmetric distance measure, i.e., $d_{B,A} \neq d_{A,B}$.

3. DISTANCE METRIC LEARNING

Distance metric learning has been studied in machine learning literature. Classical metric learning algorithms are motivated by Mahalanobis distance given as

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^T \Sigma^{-1} (x_1 - x_2)}, \tag{7}$$

where Σ is the covariance matrix of X. The main intuition behind Mahalanobis distance is that it calculates the Euclidean distance in a linearly transformed space by R, where $R^T R = S^{-1}$. Mahalanobis distance is a convenient metric since it is scale-invariant, and it takes the correlations of data set into account. The linear transform R makes the data have the isotropic covariance as the same as the covariance of multivariate normal distribution. The goal of metric learning algorithms is to learn A, which corresponds to the precision matrix (Σ^{-1}) based on a variety of criterion.

$$d(x_1, x_2) = \sqrt{(x_1 - x_2)^T A (x_1 - x_2)}, \tag{8}$$

where A is a symmetric positive semidefinite matrix $A \succeq 0, A = A^T$. Training may require additional labels such as classes and similar/dissimilar pairs depending on the objective of the frameworks.

The main difficulty to apply the classical metric learning algorithms to cover song identification problems is that the songs should be represented in a vector space. One simple approach is to extract a set of fixed length features from songs, e.g., mean MFCCs, mean Chroma, and beats per minute (BPM). But these features do not capture the temporal information within a song. So, a variety of time series analysis methods has been shown to be more effective such as dynamic time warping (DTW).

Can we embed songs in a vector space preserving the temporal information? If this is possible, then distance metric learning algorithms are able to find a better distance between songs with both the temporal information and additional labels (similar/dissimilar pairs or classes). One option is kernel PCA. Fortunately, distance metric learning can be extended in the context of kernel methods as well. The kernel methods do not require the original data to be in a vector space. We can get a gram matrix (or inner product matrix) by pairwise dissimilarity measures. For embedding, other embedding algorithms can be used for instance multidimensional scaling (MDS), ISOMAP, locally linear embedding (LLE) and so on. We discuss our framework to calculate the gram matrix and embed songs in a vector space shortly.

3.1 Embedding of songs

As discussed above, we start from a pairwise dissimilarity measures. We calculate the distance matrix as described in Section 2. The gram matrix in the conventional kernel methods should be symmetric positive-semidefinite matrix. If the matrix is given as not symmetric (e.g. SiMPle), it needs to be symmetrized by $d'_{i,j} = \frac{1}{2}(d_{i,j} + d_{j,i})$, where $d_{i,j}$ is defined in Eqn (6).

After symmetrization of the distance matrix, we perform a kernel PCA. PCA seeks for eigenvectors of the covariance matrix of the data given as

$$C = \frac{1}{N}\sum_{i}^{N} x_i, x_i^T. \tag{9}$$

Similarly, kernel PCA seeks for eigen functions of the covariance function. In other words, Given a nonlinear function $\Phi(\cdot)$ to map data to feature space, the covariance matrix is calucated by

$$\bar{C} = \frac{1}{N}\sum_{i}^{N} \Phi(x_i)\Phi(x_i)^T, \tag{10}$$

where $\Phi(x)$ is centered, i.e., $\sum_i^N \Phi(x_i) = 0$. Thanks to the kernel trick, without performing the map Φ, kernel methods can be computed by kernel functions $K_{ij} = k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$. In this paper, we used the Radial basis function (Gaussian kernel). The kernel function is given by

$$\begin{aligned} k(x_i, x_j) &= \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{(d'_{ij})^2}{2\sigma^2}\right) \end{aligned} \tag{11}$$

where d'_{ij} is the symmetrized dissimilarity measure (distance) and σ is a tuning parameter. So only with the pairwise dissimilarity measure, the gram matrix for kernel PCA is obtained. The remaining procedure is similar to classical PCA. For more details, we refer the reader to [12].

Let $z_1, \cdots, z_N$ be the new representation of songs from KPCA described above. In our experiments, the number of basis functions and the bandwidth σ in Eqn (11) were empirically selected.

Remarks. When KPCA embeds songs in a vector space based on dissimilarity measured by SiMPle, we found that in the vector representations of some songs may have extremely large norms. So regardless of the metric learned by A in Eqn (8), these songs tend to have large distance from most of other songs. In other words, these songs cannot be detected as a cover song. To prevent this problem, we normalized the vector representation of songs $z_1, \cdots, z_N$ by their ℓ_2 norms. All songs now are on the unit sphere and the problem can be alleviated. The empirical performance gain is provided in Section 4.3. The normalized vector representation will be used for metric learning.

3.2 Metric Learning

We adopt the Information-Theoretic Metric Learning (ITML) [3] except the regularization to make A close to the prior A_0, which is selected by users. Let $\mathbb{S}$ and $\mathbb{D}$ be a similar set and a dissimilar set, respectively. Then optimization program is given as

$$\begin{aligned} \min_A \sum_{(i,j)\in\mathbb{S}} & \max(0, \mathrm{Tr}(AZ_{ij}Z_{ij}^T) - u) \\ + \sum_{(i,j)\in\mathbb{D}} & \max(0, l - \mathrm{Tr}(AZ_{ij}Z_{ij}^T)), \\ \text{s.t. } & A \succeq 0 \text{ and } A^T = A, \end{aligned} \tag{12}$$

where $Z_{ij} = z_i - z_j$ and $\mathrm{Tr}(\cdot)$ is the trace. The input z_i for the metric learning in Eqn (12) is the new (normalized) representation of ith song obtained by KPCA. The objective of this metric learning is to seek for an A matrix, which make the distance of dissimilar pairs larger than a threshold l (and the distance of similar pairs smaller than a threshold u). A similar pair consists of an original song and its cover song, or it can be two cover songs from an original song. The dissimilar pairs in our experiments are all possible pairs of songs except the similar pairs.

The way we label the relationship between songs naturally yields highly skewed labels. For example, if two out of ten songs are the only covers, then we have one similar pair against $\binom{10}{2} - 1 = 44$ dissimilar pairs. Interestingly, it turns out that the skewness of labels does not hurt the performance of our framework. Rather, as the number of dissimilar pairs increases, the performance increases. Our experiment evidences this phenomenon, see Section 4.3.

The formulation in Eqn (12) is optimized by projected stochastic subgradient descent as in Alg. 1. Since the objective function is a nonsmooth and convex function, we used the subgradient descent function. Also for the symmetric positive semidefinite constraint, the projection is added in line 12. The step size α can be updated by any reasonable method.

Algorithm 1 Projected SSGD for metric learning.

```
1:  for k=1:maxiter do
2:      DATA′ = randperm(DATA)
3:      for $(i, j)$ = DATA′ do
4:          $p = 0$
5:          if $(i, j) \in S$ then
6:              if $\max(0, \mathrm{Tr}(AZ_{ij}Z_{ij}^T) - u) > 0$ then
7:                  $p = Z_{ij}Z_{ij}^T$
8:          else
9:              if $\max(0, l - \mathrm{Tr}(AZ_{ij}Z_{ij}^T)) > 0$ then
10:                 $p = -Z_{ij}Z_{ij}^T$
11:         $A = A - \alpha p$
12:         $A = \pi_{\text{psd}}(A)$
13:     update $\alpha$
```

4. EVALUATION

4.1 Dataset and Metrics

We used two separate datasets to evaluation and train our method. The specification of our evaluation dataset resem-

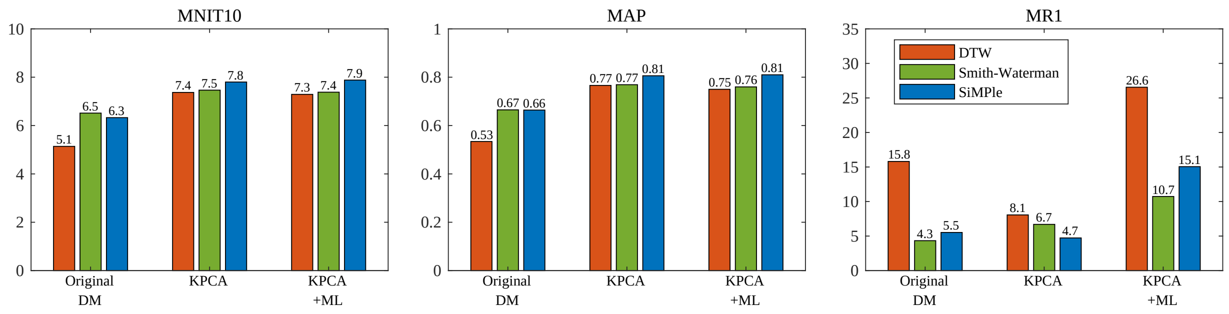


Figure 3. Improved performance by each step in the proposed method: the original distance matrix, kernel PCA, and metric learning with kernel PCA.

bles as in the MIREX cover song identification task [1]. The evaluation set $\mathbb{E}$ consists of 330 cover songs, which make the query set $\mathbb{Q}$, and 670 non-covers. There are 30 different kinds of cover songs and each has 11 cover versions. The training dataset consists of 254 covers and each cover has two to five different versions, and have 1,175 songs in total. It was used as the core set $\mathbb{C}$ in the experiments. Both datasets are disjoint, and contain various genres of Korean pops released from 1980 to 2016.

We employed four conventional metrics that have been used in the MIREX: total number of covers identified in top 10, mean number of covers identified in top 10 (MNIT10), mean average precision (MAP), and mean rank of the first correctly identified cover (MR1). In the experimental results, we skipped the first one because it is exactly the same as the second metric multiplied by $|\mathbb{Q}|$.

4.2 Experiments

Since selection of features and calculation of pairwise song distance are not our interest, the chroma energy normalized statistics (CENS) [9] was fixed as the feature vector and extracted for every half a second in all the following experiments. Also, before calculating the distance between two songs, we transposed one using the optimal transpose index (OTI) [13] so that both songs have the same key.

In the first experiment, we examined the effect of two proposed steps on identification performance: new representation transformed by the kernel PCA, and the metric learning using similar/dissimilar pairs in the core set. 135 basis functions were empirically selected, and 2435 similar pairs (for covers) and 687k dissimilar pairs (for non-covers) were used as training data for metric learning. This experiment allows reporting the maximum performance we could achieve and how each part of the proposed method contributes to the performance improvement.

The second experiment aims to verify that the metric learning converges to a higher performance as more training data are used. We tested different numbers of the training data, which are song pairs in the core set. Songs are randomly chosen with the given number of pairs in each class. Since we have much less similar pairs than dissimilar pairs, the training will be imbalanced when all possible similar pairs are used. In this experiment, we fixed the original distance measure by the SiMPle algorithm.

4.3 Results and Discussions

The first experimental result is shown in Figure 3. When comparing the original performance of the existing algorithms, Smith–Waterman algorithm achieved 26% higher performance than classic DTW. This is almost the same result as reported in a previous work [15]. The SiMPle algorithm, which we consider to be the state-of-the-art method, originally scored a slightly lower performance than Smith–Waterman algorithm in our experiment. However, the proposed method improved its original performance by 25% (in MNIT10), which was the largest improvement. Algorithms based on dynamic programming (DP) seem to have limitations in potential performance gain. One possible reason is that the differences in distance between similar and dissimilar pairs are not so discriminated; while the SiMPle mainly depends on local similarity joins with a fixed length of 10 seconds, DP-based algorithms may take much longer sequences into account. Meanwhile, MR1 was increased by the metric learning. This will be discussed in detail in the next paragraph.

Figure 4 shows the learning curve of the metric learning with different number of pairs. A hundred pairs for each class were not sufficient to converge. As more pairs were used for training, both MNIT10 and MAP converged to higher performance. This result was also obtained when more but imbalanced training data was used. Interestingly, the trend of MR1 increased after a certain number of iterations. This is caused by that the metric learning concentrates on the performance for a large majority of query songs, while it fails for very few queries. To support this, we first calculated the median instead of the arithmetic mean rank, and noticed that the correct cover had the highest similarity in most queries (i.e. median $= 1$) for every number of pairs and iteration. Nevertheless, since it is not suitable to show that the performance is getting improved with more iterations, the 90th percentile of rank of the first correctly identified cover (P_{90}R1) is shown instead in the figure.

In summary, our experiments confirm that the use

[1] `http://www.music-ir.org/mirex/wiki/2016:Audio_Cover_Song_Identification`

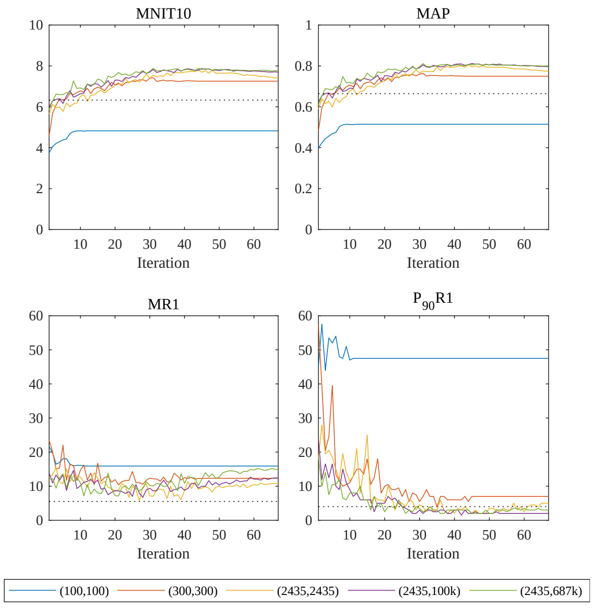


Figure 4. Learning curve of the metric learning with different number of pairs (similar, dissimilar). The black dotted line indicates each metric resulted from the original distance matrix.

of KPCA and metric learning on the SiMPle algorithm achieves the highest performance in a general situation. Although MR1 was increased by metric learning, it was explained by the second experiment showing that the trained metric failed only for a very small number of queries, while it was optimized for the most of queries. Since metric learning takes longer computation time and its performance improvement was not prominent as much as KPCA, it is possible to expect a good performance gain using empirically optimized parameters of KPCA for a fixed dataset. However, considering that scalability is an important issue in cover song identification, metric learning cannot be excluded especially for large-scale collections.

In the new representation through KPCA, each dimension represents the distance from each core song. This implies that core songs with diverse styles of music allows dimensions to be nearly orthogonal, and may yield better performance. In the metric learning, on the other hand, higher performance could be achieved with a sufficient number of similar and dissimilar pairs for training. It is not easy to satisfy both of the above conditions simultaneously, because collection of songs with various styles includes songs that are not very popular and rarely covered. Therefore, when a high recall is required (to avoid very low identification performance for very few queries), it is expected that it can be more important to have many similar pairs than various styles.

5. CONCLUSIONS

In this paper, we have presented a novel approach to improve the performance of existing algorithms for cover song identification. Our approach exploits an external set of core songs so that all the given songs are newly represented by the distance between each core song. Through the distance metric learning after embedding of songs using kernel PCA, the original performance of the state-of-the-art method was improved by more than 20%.

With different features and distance measures, the proposed method can be easily applied to similarity analysis of other tag-based data such as genre, mood, and style. We plan to further explore our approach to many other MIR tasks, and seek for proper criteria to choose the core set from large-scale collections. A sufficient number of well-organized core songs and efficient computation for metric learning will be also studied in the next step.

6. ACKNOWLEDGEMENTS

This research project was supported by Ministry of Culture, Sports and Tourism (MCST) and from Korea Copyright Commission in 2017. [Development of predictive detection technology for the search for the related works and the prevention of copyright infringement]

7. REFERENCES

[1] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, pages 117–120. IEEE, 2011.

[2] Ning Chen, J Stephen Downie, Haidong Xiao, Yu Zhu, and Jie Zhu. Modified perceptual linear prediction liftered cepstrum (mplplc) model for pop cover song recognition. In *International Society for Music Information Retrieval Conference*, pages 598–604, 2015.

[3] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[4] Daniel PW Ellis and Graham E Poliner. Identifying cover songs' with chroma features and dynamic programming beat tracking. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1429. IEEE, 2007.

[5] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(1):67–72, 1975.

[6] Kyogu Lee. Identifying cover songs from audio using harmonic representation. *MIREX task on Audio Cover Song Identification*, 2006.

[7] Abdullah Mueen, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2015. Available: `http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html`.

[8] Meinard Muller and Sebastian Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):649–662, 2010.

[9] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *International Society for Music Information Retrieval Conference*, volume 2005, page 6th, 2005.

[10] Cory S Myers and Lawrence R Rabiner. A comparative study of several dynamic time-warping algorithms for connected-word recognition. *Bell System Technical Journal*, 60(7):1389–1409, 1981.

[11] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[12] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588, 1997.

[13] Joan Serra, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.

[14] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008.

[15] Diego F Silva, Chin-Chin M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, Eamonn Keogh, et al. Simple: assessing music similarity using subsequences joins. In *International Society for Music Information Retrieval Conference*, 2016.

[16] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

EXAMINING MUSICAL MEANING IN SIMILARITY THRESHOLDS

Katherine M. Kinnaird
Brown University
katherine_kinnaird@brown.edu

ABSTRACT

Many approaches to Music Information Retrieval tasks rely on correctly determining if two segments of a given musical recording are repeats of each other. Repetitions in recordings are rarely exact, and identifying the appropriate threshold for these pairwise decisions is crucial for tuning MIR algorithms. However, current approaches for determining and reporting this threshold parameter are devoid of contextual meaning and interpretations, which makes comparing previous results difficult and which requires access to specific datasets. This paper highlights weaknesses in current approaches to choosing similarity thresholds, provides a framework using the *proportion of orthogonal musical change* to tie thresholds back to feature spaces with the cosine dissimilarity measure, and introduces new research possibilities given a music-centered approach for selecting similarity thresholds.

1. INTRODUCTION

Since Foote introduced the self-similarity matrix as a technique for visualizing and representing audio data [7], matrix representations have been widely used to represent music-based data, such as songs or musical scores, when addressing different kinds of tasks in Music Information Retrieval [6, 11, 13, 17]. Recordings of music often contain slight variations between repeated sections either due to artistic interpretations or noise introduced by the recording environment. Addressing these MIR tasks often requires grouping time steps together using a threshold on the self-(dis)similarity matrix representation to determine which pairs of time steps are similar enough to be classified as repetitions of each other. There are two issues at play when choosing this similarity threshold: 1) selecting the best value given the task and data, and 2) using the value with the best musical interpretation.

Similarity thresholds are currently determined in ways that prioritize computational successes and ignore tangible musical interpretations. These thresholds are usually dependent on the data at hand and reported as a selection method (say a fixed percentage) instead of as a particular threshold value. These data-dependent thresholds, reported as methods, require access to common datasets in order to compare previous and current research. Furthermore, many of the processes for determining this crucial threshold do not have a mechanism for connecting that threshold back to the original feature space. For example, current methods give little understanding to what a "small-value" cosine dissimilarity measurement corresponds to in terms of musical sounds such as notes and chords.

Instead of only justifying similarity thresholds based on statistical theory or computational success, we argue musical meaning should be included in the selection and discussion of a similarity threshold. In Section 2, examples based on a jazz lead sheet offer motivation for similarity thresholds with musical context. In Section 3, we model a framework for tying a chosen threshold to a particular feature space via the concept of the maximum proportion of orthogonal musical change. In Section 4, we introduce how music-centered thresholds can enhance MIR research.

2. MOTIVATION WITH EXAMPLES

In MIR literature, there are a variety of methods for setting the similarity threshold used to decide when sections of a song are similar. Current methods have been based on statistical ideas combined with concise algorithmic explanations. In [1, 11, 15], for a given recording of a performance of a piece of music, the threshold was specified so that a fixed percentage of a matrix representation (either self-similarity matrix or self-dissimilarity matrix - SDM) would be selected. The method in [18–20] sets the meaning of "similar" for each time step by first looking for the κ nearest neighbors of a given time step and then by enforcing a mutual condition; that is that time steps i and j are determined to be similar if both time step i is a κ nearest neighbor of time step j and vice versa. In [3], the threshold was set using statistical techniques on a set of sample data. In [8–10], Goto determined a threshold using the automatic threshold selection method developed by Otsu [16] which selects a threshold using statistics of the grey-level histogram of a particular image. In the case of Goto's work [8–10], the image is a matrix representation for a song.

While the above methods are efficient and have satisfying connections to our intuition about similarity, a crucial weakness of these methods is a lack of a musical connection for the similarity threshold. For example, the fixed percentage thresholds in [1, 11, 15] are easy to set and offer clear methods for reproducing those workflows, but no musical intuition is offered for these methods. Underlying fixed percentage thresholds is the assumption that all music has the same proportion of similarity, which certainly would not be the case in a collection with both classical and jazz recordings. The method in [18–20] to some degree addresses the flaws in this assumption, but this method leaves unanswered what it means musically to be mutual κ nearest neighbors.

The following four examples based off a jazz lead sheet use the bottom-10% paradigm for similarity threshold selection and highlight some of the issues with this method. The first example is just the chords by beat as described in the lead sheet. The second example adds absolute Gaussian noise, while the third example adds notes in a restricted manner, seeking to mimic the spontaneous composition of jazz music. The final example adds both absolute Gaussian noise and restricted "note" noise. These examples are constructed from a human coded .jazz file [1] of *Aisha* by McCoy Tyner from 1961 found in the *iRb Corpus in **jazz format* dataset [2]. Beat tracking was not used since these examples are based on a version of the piece's lead sheet. Each time step represents 8 continuous beats by concatenating adjacent 8 feature vectors (one per beat).

For each example, the distribution of dissimilarity values and the thresholded SDM are shown. All results are from single runs of the associated random processes, but similar results occur with repeated trials. For the thresholded SDM, the original SDM values are retained to further highlight contrast between examples.

Example 1 - Jazz Lead Sheet

This example is the ground truth for the true repeated structure of the lead sheet. We assume that there is neither noise nor spontaneous composition on the track.

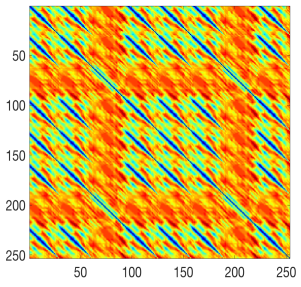
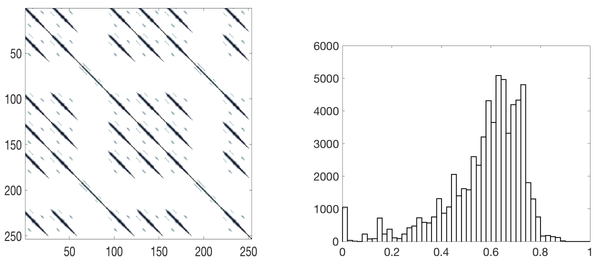


Figure 1. Complete SDM for *Aisha* lead sheet. Values near 0 are dark.

Using the bottom-10% paradigm, the threshold T is 0.375, meaning that two feature vectors with the angle between them no greater than 51.318 degrees will be deemed similar enough to be repeats of each other. This is quite a generous threshold; for example, a feature vector representing a C chord (held for 8 beats) and a feature vector representing C-minor chord (also held for 8 beats) would be deemed as repeats of each other.

[1] The .jazz file was converted to a .txt file using code by Yuri Broze [2]. Chromagrams were then extracted using a new converter file, available at `https://github.com/kmkinnaird/MusicalThresh`

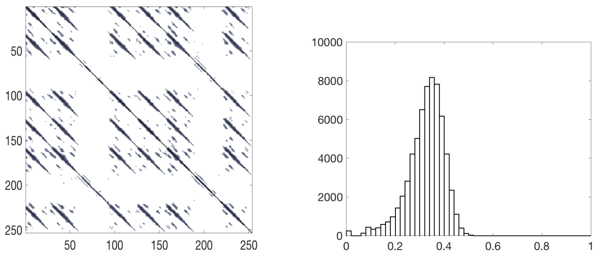


(a) Thresholded SDM under bottom-10% paradigm (b) Histogram of all dissimilarity values

Figure 2. *Aisha* lead sheet without additions

Example 2 - Jazz Lead Sheet with Gaussian Noise

In this example, we add proxy for general noise (such as feedback in the recording environment) to the lead sheet. To each note-beat entry of the chroma matrix for the lead sheet, we add the absolute value of a random sample from the Gaussian centered at 0 with standard deviation 0.5.



(a) Thresholded SDM under bottom-10% paradigm (b) Histogram of all dissimilarity values

Figure 3. *Aisha* lead sheet with added track noise

In addition to most of the similarity from Example 1, additional segments were classified as repeats using the bottom-10% paradigm, meaning that "similarity" is being created under this threshold selection method. However, this example's threshold value is lower, so two audio shingles must be more similar to be considered repeats than in Example 1. The threshold T is approximately 0.232, meaning that two feature vectors with the angle between them no greater than 39.818 degrees will be deemed similar enough to be repeats of each other. This shifted (and possibly contradictory) definition of similarity may be appropriate given the data but there is no musical interpretation of the threshold to support this choice. The lower threshold does reflect the compression of the distribution of dissimilarity values, shown in Figure 3(b).

Example 3 - Jazz Lead Sheet with "Note" Noise

In this third example, we add a proxy for spontaneous composition. This added "note" noise is restricted to the notes within the chord specified on the lead sheet and has its note weight randomly selected from the distribution of note values, shown in Figure 4.

The threshold T for this example is approximately 0.417, meaning that two feature vectors with the angle between them no greater than 54.357 degrees will be deemed

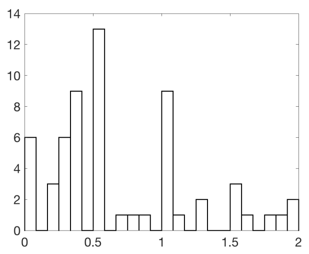
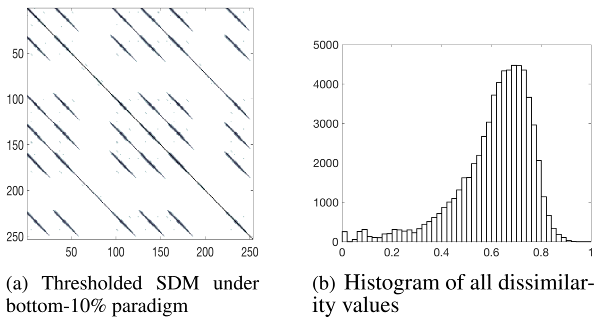


Figure 4. Histogram of note values for "note" noise

similar enough to be repeats of each other. As expected given this example's construction, this threshold is similar to the one in Example 1. However, while much of the similarity from Example 1 was found using the bottom-10% paradigm, it is clear that not all of it was. As with Example 2, this threshold may be appropriate, but there is no musical interpretation to support this choice.

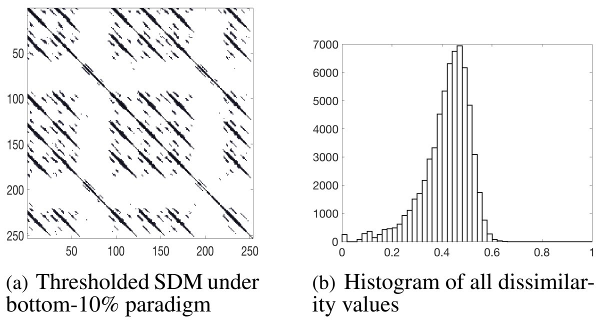


(a) Thresholded SDM under bottom-10% paradigm (b) Histogram of all dissimilarity values

Figure 5. *Aisha* lead sheet with added "note" noise

Example 4 - Jazz Lead Sheet with "Note" Noise and with Gaussian Noise

In this example, we add proxies for both track noise (as in Example 2) and "note" noise (as in Example 3). Since we are assuming that there is both spontaneous composition and additional noise on the track, it is tempting to simply add the thresholds from Examples 2 and 3. However, we cannot, given the construction of the proxies and that cosine dissimilarity measure does not observe the triangle inequality.



(a) Thresholded SDM under bottom-10% paradigm (b) Histogram of all dissimilarity values

Figure 6. Track and "note" noise added to *Aisha* lead sheet

Similar to Example 2, we have a possibly contradictory definition of similarity. In this example, the bottom-10% paradigm captures most of the similarity from Example 1 but also incorrectly matches additional repeated "similarity." However, the value of this threshold T is lower, at approximately 0.293, which translates to an angle no greater than 45.026 degrees between two feature vectors deemed similar enough to be repeats.

Comparing Four Examples

These four examples highlight some of the weaknesses in the commonly used fixed percentage threshold selection paradigm. First, the generous thresholds in Examples 1 and 3 allow for major and minor chords (such as C-major and C-minor) to be deemed as repeats of each other. However, the histogram from Example 3 is quite similar to Example 1, which signals that an appropriate choice of threshold for a lead sheet would also be appropriate to apply to a lead sheet with spontaneous composition.

Second, when a proxy for random track noise is introduced, as in Examples 2 and 4, major and minor chords would no longer be matched. However, a passing glance on the resulting thresholded SDMs in Examples 2 and 4 show sections of the lead sheet designated as repeats when they perhaps should not be. Additionally, the histograms for Examples 2 and 4 are much more compressed than those in Examples 1 and 3, which further signals a need in incorporate musical context into the selection of similarity thresholds.

Even though these four examples are based on a lead sheet, of which three employ random processes as proxies for track noise and spontaneous compositions, these controlled and constructed examples demonstrate the need for careful examination of the meaning and limitations of thresholds used in MIR tasks and approaches.

3. RELATING T TO MAXIMUM PROPORTION OF ORTHOGONAL MUSICAL CHANGE

In this section, we establish a framework for linking a similarity threshold T to the space of audio shingles composed of chroma feature vectors under the cosine dissimilarity measure. We define the *proportion of orthogonal musical change* (or POMC) for this feature space and prove a relationship between a given threshold T to POMC. Although we ground our discussion in one particular feature space, a similar procedure can be used to tie similarity thresholds to any feature space using the cosine dissimilarity measure.

3.1 Preliminary Definitions and Notation

We create overlapping *audio shingles* from k concatenated feature vectors, where k is a fixed integer [3–5]. For a timestep i, the chroma feature vector χ_i is the column vector of 12 non-negative entries, where each entry corresponds to one of the Western pitch classes $\{C, C\#, ..., B\}$ encoding the amount of that pitch class in the i^{th} observation [14].

For time-step i, the audio shingle of length k, incorporating local information, is the column vector α_i:

$$\alpha_i = \left[\chi_i^t, \chi_{(i+1)}^t, \chi_{(i+2)}^t, \cdots, \chi_{(i+k-1)}^t\right]^t \quad (1)$$

Each audio shingle is an element of $\mathbb{R}_{\geq 0}^{(k\times 12)}$, the non-negative closed orthant of $\mathbb{R}^{(k\times 12)}$ and can be regarded as vectors that start at the origin. Let $\theta_{\alpha_i,\alpha_j}$ be the angle between α_i and α_j. Since $\alpha_i, \alpha_j \in \mathbb{R}_{\geq 0}^{(k\times 12)}$, then $\theta_{\alpha_i,\alpha_j} \in [0, \frac{\pi}{2}]$. The pairwise cosine dissimilarity between two audio shingles α_i and α_j is defined as:

$$\mathcal{D}_{i,j} = 1 - \cos\theta_{\alpha_i,\alpha_j} \quad (2)$$

It is natural to ask: *Given the value $\mathcal{D}_{i,j}$, what are the musical differences between those two time steps?* We introduce the notion of *proportion of orthogonal musical change* (POMC), or rather the amount an audio shingle α_i must change orthogonally (before scaling) in order to become α_j. POMC encodes of how much one audio shingle can be comprised of elements perpendicular to another audio shingle before we say these two audio shingles are no longer considered to be "similar" of one another.

Consider Figure 7; the vector $\vec{\gamma}$ is orthogonal to α_i and when added to α_i will meet α_j. We can scale the vector $(\alpha_i + \vec{\gamma})$ to match α_j. Similarly $\vec{\phi}$ is orthogonal to α_j and when added to α_j meets α_i. We can scale $(\alpha_j + \vec{\phi})$ to match α_i. The length of $\vec{\gamma}$ is $||\alpha_i|| \cdot \tan\theta_{\alpha_i,\alpha_j}$, and the length of $\vec{\phi}$ is $||\alpha_j||\cdot\tan\theta_{\alpha_i,\alpha_j}$. So $\tan\theta_{\alpha_i,\alpha_j}$ is the amount of orthogonal change for α_i to become a scalar multiple of α_j and vice versa.

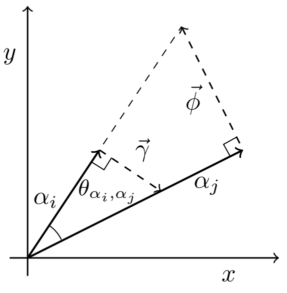


Figure 7. Visualization of orthogonal musical change of α_i onto α_j and of α_j onto α_i, represented respectively by the vectors $\vec{\gamma}$ and $\vec{\phi}$.

Definition 3.1. For a pair of audio shingles α_i and α_j, the *proportion of orthogonal musical change* (POMC) is given by $\tan\theta_{a_i,a_j}$.

3.2 Maximum POMC Given T

Suppose that we have one audio shingle, denoted $\vec{\xi}$ terminating at point ξ, and that we want to classify all audio shingles that are repetitions of $\vec{\xi}$. Let T be the threshold determining whether pairs of audio shingles are close enough to be repeats. We define θ_T as $\cos^{-1}(1-T)$.

Let Ξ be the set of audio shingles that are less than T cosine dissimilar from $\vec{\xi}$. So $\vec{v} \in \Xi$, iff $1 - \cos\theta_{\vec{v},\vec{\xi}} \leq T$, for $\theta_{\vec{v},\vec{\xi}}$. Additionally, for each vector $\vec{v} \in \Xi$, we have:

$$\cos\theta_{\vec{v},\vec{\xi}} \geq 1 - T = \cos\theta_T \quad (3)$$

Definition 3.2. Given T, the *maximum POMC*, denoted ρ, is $\tan(\theta_T)$, where $\theta_T = \cos^{-1}(1-T)$.

We begin establishing the comparison between the audio shingles in Ξ and $\vec{\xi}$ using just POMC. We first note that the set of audio shingles orthogonal to $\vec{\xi}$ is comprised of the audio shingles representing silence (i.e. those without any notes) and the audio shingles that do not have notes in common time with $\vec{\xi}$. For example, if $\vec{\xi}$ represented a C chord followed by a F chord, then an audio shingle that is orthogonal to it could be one representing a C# chord followed by an E chord. Given the importance of note and chord order in music generally, the audio shingle representing an F note followed by a C chord is orthogonal to a second representing a C chord followed by an F note. Neither of the above pairs would be mistaken as similar, and so we restrict $\theta_T \in [0, \frac{\pi}{2})$, since including $\theta_T = \frac{\pi}{2}$ would imply that orthogonal pairs of audio shingles are similar.

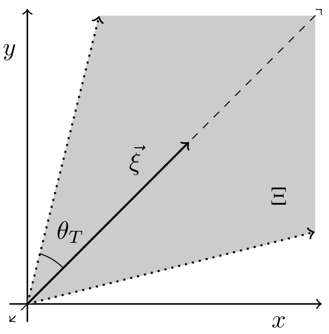


Figure 8. Visualization of Ξ, the set of audio shingles that are less than T cosine dissimilar from $\vec{\xi}$. The gray area flanked in dotted arrows is the set Ξ. The dashed line continuing from $\vec{\xi}$ is the subspace defined by $\vec{\xi}$.

More often, we want to compare pairs like a C chord followed by a second C chord with a C chord followed by a C7 chord, and determine if these two audio shingles are close enough to be deemed similar. These two audio shingles are the same save for the B♭ note in the second chord. Clearly a lone B♭ note is orthogonal to a C chord but is not orthogonal to the C7 chord. However, the C7 chord can be decomposed into the sum of a C chord and a B♭ note. In other words, the C7 chord is the C chord plus a vector orthogonal to it. Such a decomposition is at the heart of the concept of POMC.

Returning to our general case with audio shingle $\vec{\xi}$, we make the following definitions generalizing the above comparison of the C and C7 chords:

Definition 3.3. Let $\xi^{\perp}$ denote the hyperplane that is orthogonal to the vector $\vec{\xi}$ with the point $\xi \in \xi^{\perp}$.

We note that $\xi^{\perp}$ does not require that vectors in $\xi^{\perp}$ to be within Ξ. The following definition adds this restriction:

Definition 3.4. Let V_+ be the set of vectors originating at the point ξ and terminating at a point in $\xi^\perp$ such that for $\vec{v}_+ \in V_+$, we have that the cosine of the angle between $(\vec{\xi} + \vec{v}_+)$ and $\vec{\xi}$ is greater than or equal to $\cos\theta_T$.

For any vector $\vec{v}_+ \in V_+$, we have that the angle between $\vec{\xi}$ and $\vec{v}_+$ is the right angle in a right triangle with one leg along $\vec{\xi}$ with length $||\vec{\xi}||_2$ and with another leg along $\vec{v}_+$ with length $||\vec{v}_+||_2$. The tangent of the angle between $(\vec{\xi} + \vec{v}_+)$ and $\vec{\xi}$ is equal to $\frac{||\vec{v}_+||_2}{||\vec{\xi}||_2}$, which must be less than or equal to $\tan\theta_T$. So $||\vec{v}_+||_2 \leq ||\vec{\xi}||_2 \cdot \tan(\theta_T)$.

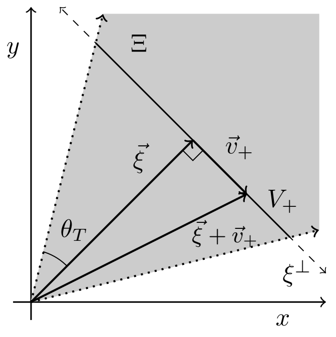


Figure 9. Visualization of the right triangle formed by $\vec{\xi}$ and $\vec{v}_+ \in V_+$ in $\mathbb{R}^2$. The solid line perpendicular to $\vec{\xi}$ represents V_+ and is $2||\vec{\xi}||_2 \cdot \tan(\theta_T)$ long. The dashed line continuing from V_+ combined with V_+ represents $\xi^\perp$.

The set V_+ represents the set of audio shingles that are created by adding to $\vec{\xi}$ an orthogonal vector of length no longer than $||\vec{\xi}||_2 \cdot \tan(\theta_T)$. For example, if $T = 0.1$ and if $\vec{\xi}$ represented one C chord, then the audio shingle representing a C7-chord would terminate in the hyperplane $\xi^\perp$, but would not be included in Ξ; however, if $T = 0.14$, then it would be included in Ξ.

3.3 Decomposition of Elements in Ξ

Thus far we have established the relationship between our audio shingle $\vec{\xi}$ and the audio shingles ending in $\xi^\perp \cap \Xi$. We are interested in understanding the relationship of T with all of Ξ. We offer the following decomposition for the elements of Ξ, which connects the definitions of the previous section to Definition 3.1 shown in Figure 7.

Proposition 3.1. *The set Ξ is equal to the set $\mathcal{S}$, given by*

$$\mathcal{S} = \left\{ w(\vec{\xi} + \vec{v}_+) | \vec{v}_+ \in V_+, w \in \mathbb{R}_{\geq 0} \right\}.$$

Proof: First, we show that $\mathcal{S} \subseteq \Xi$. We begin by letting $\vec{v}$ be an element of $\mathcal{S}$. So $\vec{v} = w(\vec{\xi} + \vec{v}_+)$ for some $\vec{v}_+ \in V_+$ and some $w \in \mathbb{R}_{\geq 0}$. Let β denote the angle between $\vec{v}$ and $\vec{\xi}$, which is also the angle between $(\vec{\xi} + \vec{v}_+)$ and $\vec{\xi}$ since $\vec{v}$ and $(\vec{\xi} + \vec{v}_+)$ are scalar multiples of each other. By the definition of V_+, we have that $\cos\beta \geq \cos\theta_T$. So $\vec{v} \in \Xi$ and thus $\mathcal{S} \subseteq \Xi$.

Next, we show that $\Xi \subseteq \mathcal{S}$. Let $\vec{v} \in \Xi$ and let $\theta_{\vec{v},\vec{\xi}}$ be the angle between $\vec{v}$ and $\vec{\xi}$. Then:

$$1 - \cos\left(\theta_{\vec{v},\vec{\xi}}\right) \leq T \tag{4}$$

Let v_+ be the point where $\vec{v}$ intersects $\xi^\perp$ (noting that it may be necessary to continue in the direction of $\vec{v}$ to intersect with $\xi^\perp$). We have a right triangle with one leg in the direction of $\vec{\xi}$ with length $||\vec{\xi}||_2$ and another leg from $\vec{\xi}$ to v_+ that is length $||v_+ - \xi||_2$. Let $\vec{v}_+$ be the vector from $\vec{\xi}$ to v_+. By definition of v_+, then $||\vec{v}_+||_2 = ||v_+ - \xi||_2$ and thus:

$$\tan\left(\theta_{\vec{v},\vec{\xi}}\right) = \frac{||v_+ - \xi||_2}{||\vec{\xi}||_2}. \tag{5}$$

Leveraging Eqn (5), we have that

$$\begin{aligned} ||\vec{v}_+||_2 &= ||v_+ - \xi||_2 = ||v_+ - \xi||_2 \cdot \frac{||\vec{\xi}||_2}{||\vec{\xi}||_2} \\ &= ||\vec{\xi}||_2 \cdot \tan\left(\theta_{v,\vec{\xi}}\right) \leq ||\vec{\xi}||_2 \cdot \tan(\theta_T). \end{aligned}$$

The last inequality is due to the angle between two vectors in Ξ is less than θ_T, that $\theta_T \in [0, \frac{\pi}{2})$, and that $\tan(x)$ is a monotonically increasing function on the interval $[0, \frac{\pi}{2})$. So $\vec{v}_+ \in V_+$. Let w be the positive scalar that we multiply $(\vec{\xi} + \vec{v}_+)$ by to get $\vec{v}$. Then $\vec{v} = w(\vec{\xi} + \vec{v}_+)$ for some $\vec{v}_+ \in V_+$ and some $w \in \mathbb{R}_{\geq 0}$. So $\vec{v} \in \mathcal{S}$ and $\Xi \subseteq \mathcal{S}$. $\square$

This proposition gives us a decomposition for all audio shingles that are less than T cosine dissimilar from $\vec{\xi}$, regardless of how T is set. Using standard orthogonal projections, we can decompose any audio shingle into the form $w(\vec{\xi} + \vec{\alpha})$, where $\vec{\alpha}$ is audio shingle orthogonal to $\vec{\xi}$. To check if $\vec{\alpha} \in V_+$, we compute $||\vec{\alpha}||_2$ and see if $||\vec{\alpha}||_2 \leq ||\vec{\xi}||_2 \cdot \tan(\theta_T)$. Proposition 3.1 gives contextual meaning to our thresholds, that is the maximum proportion of orthogonal notes allowed between two audio shingles of at most T cosine dissimilarity.

3.4 Relating Choice of T to Audio Shingles via Maximum POMC

The above decomposition for vectors within T cosine dissimilarity measure from $\vec{\xi}$ provides us an avenue for relating our chosen thresholds directly to musical building blocks such as notes and chords, when they are represented as chroma feature vectors. This means that we can set a threshold by directly encoding acceptable musical variation for a small segment instead of setting the threshold using parameters free from musical context, such as a fixed percentage of entries from a matrix representation or a fixed-number of nearest neighbors.

We can set a threshold in one of three ways: 1) choosing T using existing methods, 2) setting the largest allowable θ_T between two audio shingles classified as similar enough, or 3) by setting ρ, the maximum POMC. Since T, θ_T, and ρ are functions of each other, fixing one inherently fixes the other two, and so we have an interpretation for that threshold in the space of audio shingles (under the cosine dissimilarity measure), returning musical context to what we mean by "similar structure."

- If we fix T, then we have $\theta_T = \cos^{-1}(1 - T)$ and

$$\rho = \frac{\sqrt{1 - (1 - T)^2}}{(1 - T)}$$

- If we fix θ_T, then $T = 1 - \cos\theta_T$ and $\rho = \tan\theta_T$.
- If we instead fix ρ, then we have $\theta_T = \tan^{-1}(\rho)$ and

$$T = 1 - \frac{1}{\sqrt{\rho^2 + 1}}$$

3.5 Returning to Motivating Examples

In Section 2, we described the thresholds for each example in terms of θ_T, which is still unsatisfactory in terms of musical intuition. Now we will interpret each T using ρ.

The thresholds in Examples 1 and 3 in Section 2 have similar interpretations, which makes sense given their constructions. In Example 1, we have $T = 0.375$. So $\rho = 1.249$, meaning that for each note in a given audio shingle $\vec{\xi}$, we can add an orthogonal vector of notes with 1.249 times the magnitude of $\vec{\xi}$ and have the result be considered similar to $\vec{\xi}$. This is quite a generous threshold. For example, an audio shingle representing a C chord whole note is considered similar to a second shingle representing a C chord whole note plus a D-minor6 chord whole note and a B♭ dotted-half note. Example 3 has a similarly generous threshold with $T = 0.417$. So $\rho = 1.395$, and we can add a few more orthogonal notes to $\vec{\xi}$ than in Example 1 and still have the result be considered similar to $\vec{\xi}$.

Examples 2 and 4 in Section 2 both include the incorporation of Gaussian noise, and their associated thresholds have similar interpretations. In Example 2, we have $T = 0.232$; so $\rho = 0.834$. In Example 4, we have $T = 0.293$; so $\rho = 1.001$. These thresholds are less generous than those in Examples 1 and 3. In Example 4, an audio shingle representing a C chord whole note is considered similar to a second shingle representing a C chord whole note plus a D-minor chord.

While the above interpretations offer a musical context for our similarity thresholds, these interpretations only regard the worst case (and less likely) scenario for comparing a given audio shingle $\vec{\xi}$ to another one; that is comparing $\vec{\xi}$ to one comprised of $\vec{\xi}$ added to an audio shingle orthogonal to $\vec{\xi}$. In addition to this interpretation, we would also advocate that when setting the similarity threshold, researchers also explore comparisons of $\vec{\xi}$ to audio shingles comprised of $\vec{\xi}$ with audio shingles that are not orthogonal to $\vec{\xi}$.

4. EXPANDING USES OF MAXIMUM POMC

Building off the examples in Section 2 and the methodology in Section 3, we propose research directions that could benefit from using a musically relevant threshold.

Within the song comparison tasks, we can use the maximum POMC to explore less well defined variants of the version detection task. For example, we can explore how much spontaneous composition is added to a jazz lead sheet, while also detecting the repeated sections in the lead sheet given a maximum POMC. In another direction, we could use the relationship between a threshold and maximum POMC to create a lower bound threshold for detecting recordings using auto-tune compared to those without.

Maximum POMC can be used beyond the song comparison tasks. We can leverage the maximum POMC to perform comparisons between genres, perhaps, by quantifying the amount of expected structure in a song from one genre, and comparing that to the expected value of another genre. Using ideas from topological data analysis, we can create diagrams quantifying the amount of structure in a given piece as we increase the maximum POMC. We could also use a dynamically set maximum POMC in generative music tasks to enforce musical style constraints given the target genre for the generated musical work.

5. CONCLUSION

Previous work in MIR determined and reported similarity thresholds as a specific method for a specific dataset preprocessed in a specific manner for a specific task, and thus it is hard to compare previous results. However we can more easily compare future work on both new and current song datasets if we choose a similarity threshold for our matrix representations that includes a tangible interpretation within the feature space.

This paper offered three contributions to the study of similarity thresholds used in MIR on the self-similarity (or dissimilarity) matrices, like those introduced in [7]. First, we demonstrated weaknesses in the current fixed percentage paradigm, using four examples based off one jazz lead sheet to show inconsistencies between the interpretations of the musical differences between sections of music that are regarded as similar.

Next we demonstrated that it is possible to link a threshold to the feature space of the original data, by providing a theoretical framework relating a given threshold to the space of audio shingles comprised of chroma vectors under the cosine dissimilarity measure. Crucial to this framework is the notion of *proportion of orthogonal musical change* (POMC), introduced here. This paper provides an avenue for interpreting and exploring the musical context of similarity thresholds (regardless of how they are determined) for self-dissimilarity matrices built from the space of audio shingles through the maximum POMC. Since the theoretical work in this paper only relied on facts of the cosine dissimilarity measure, the present framework could easily be adjusted to accommodate another feature space using the cosine dissimilarity measure.

Finally we briefly proposed new MIR research directions where contextually meaningful thresholds could provide insight. We also discussed how a contextually meaningful threshold could enhance current research directions.

Setting the similarity threshold can take into account both success on a particular task, given particular data, as well as a tangible musical interpretation of that threshold. Understanding that continuing to use current methods for determining the similarity threshold may be the best for continued computational success, this paper advocates for the inclusion of musical context into, at least, the discussion of the similarity threshold, if not the selection.

Acknowledgements

Part of this work is a portion of the author's doctoral thesis [12], which was partially funded by the GK-12 Program at Dartmouth College (NSF award #0947790). The author thanks Yuri Broze for his assistance installing his code [2] to create the .jazz files. The author also thanks Scott Pauls, Michael Casey, Dan Ellis, and Jessica Thompson for their feedback on the early versions of this work.

6. REFERENCES

[1] J. Bello. Measuring structural similarity in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2013–2025, 2011.

[2] Y. Broze and D. Shanahan. The iRb Corpus in `**jazz` format. `http://musiccog.ohio-state.edu/home/index.php/iRb_Jazz_Corpus`, 2012. [Online; accessed 28-September-2016].

[3] M. Casey, C. Rhodes, and M. Slaney. Analysis of minimum distances in high-dimensional musical spaces. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(5):1015 – 1028, 2008.

[4] M. Casey and M. Slaney. Song intersection by approximate nearest neighbor search. *Proc. of* 7th *ISMIR Conference*, pages 144–149, 2006.

[5] M. Casey and M. Slaney. Fast recognition of remixed audio. *2007 IEEE International Conference on Audio, Speech and Signal Processing*, pages IV – 1425 – IV–1428, 2007.

[6] M. Cooper and J. Foote. Summarizing popular music via structural similarity analysis. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 127 –130, 2003.

[7] J. Foote. Visualizing music and audio using self-similarity. *Proc. ACM Multimedia 99*, pages 77–80, 1999.

[8] M. Goto. A chorus-section detecting method for musical audio signals. *Proc. of ICASSP*, 2003.

[9] M. Goto. SmartMusicKIOSK: Music listening station with chorus-search function. *Proc. of* 16th *ACM Symposium on User Interface Software and Technology (UIST 2003)*, pages 31–40, 2003.

[10] 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*, 14(5):1783–1794, 2006.

[11] P. Grosche, J. Serrà, M. Müller, and J.Ll. Arcos. Structure-based audio fingerprinting for music retrieval. *Proc. of* 13th *ISMIR Conference*, pages 55–60, 2012.

[12] K. M. Kinnaird. *Aligned Hierarchies for Sequential Data*. PhD thesis, Dartmouth College, 2014.

[13] M. Müller. *Information Retrieval for Music and Motion*. Springer Verlag, 2007.

[14] M. Müller and S. Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. *Proc. of* 12th *ISMIR Conference*, pages 215–220, 2011.

[15] M. Müller, P. Grosche, and N. Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. *Proc. of* 12th *ISMIR Conference*, pages 615–620, 2011.

[16] N. Otsu. A threshold selection method from gray-level histograms. *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 Conference*, SMC-9(1):62–66, 1979.

[17] J. Paulus and A. Klapuri. Music structure analysis using probabilistic fitness measure and a greedy search algorithm. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(6):1159–1170, 2009.

[18] J. Serrà, M. Müller, P. Grosche, and J.Ll. Arcos. Unsupervised detection of music boundaries by time series structure features. *Proc. of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[19] J. Serrà, M. Müller, P. Grosche, and J.Ll. Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *IEEE Transactions on Multimedia*, 16(5), 2014.

[20] J. Serrà, X. Serra, and R.G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(093017), 2009.

EXPLORING TONAL-DRAMATIC RELATIONSHIPS IN RICHARD WAGNER'S RING CYCLE

Frank Zalkow, Christof Weiß, Meinard Müller
International Audio Laboratories Erlangen, Germany
{frank.zalkow,christof.weiss,meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Richard Wagner's cycle *Der Ring des Nibelungen*, consisting of four music dramas, constitutes a comprehensive work of high importance for Western music history. In this paper, we indicate how MIR methods can be applied to explore this large-scale work with respect to tonal properties. Our investigations are based on a data set that contains 16 audio recordings of the entire *Ring* as well as extensive annotations including measure positions, singer activities, and leitmotif regions. As a basis for the tonal analysis, we make use of common audio features, which capture local chord and scale information. Employing a cross-version approach, we show that global histogram representations can reflect certain tonal relationships in a robust way. Based on our annotations, a musicologist may easily select and compare passages associated with dramatic aspects, for example, the appearance of specific characters or the presence of particular leitmotifs. Highlighting and investigating such passages may provide insights into the role of tonality for the dramatic conception of Wagner's *Ring*. By giving various concrete examples, we indicate how our approach may open up new ways for exploring large musical corpora in an intuitive and interactive way.

1. INTRODUCTION

Originating in late 16th-century Florence, opera evolved as a central art form of Western culture [1]. Intended as a return to ancient Greek dramatic style, the idea of accompanied singing (*monody*) laid the ground work for two central singing styles of traditional opera: speech-like *recitatives*, which serve as a means for developing the plot, and *arias*, which emphasize the characters' feelings through cantabile melodic lines. For centuries, the structure of opera was determined by alternating such individual pieces of music, which is also known as *number opera*. In the mid-19th century, Richard Wagner developed a novel approach to operatic composition. According to his theoretical writings such as *Oper und Drama* [15], the "drama of the future" should integrate all forms of art ("Gesamtkunstwerk"). He broke with the conventions of number opera in favor of a unity of prose and music with a steady musical flow, which is often referred to as through-composed style or "endless melody" since it lacks both interruptions and exact repetitions. A central aspect of Wagner's operatic style is the frequent use of leitmotifs—short musical ideas associated with a character, a place, an item, or with emotional categories, among others.

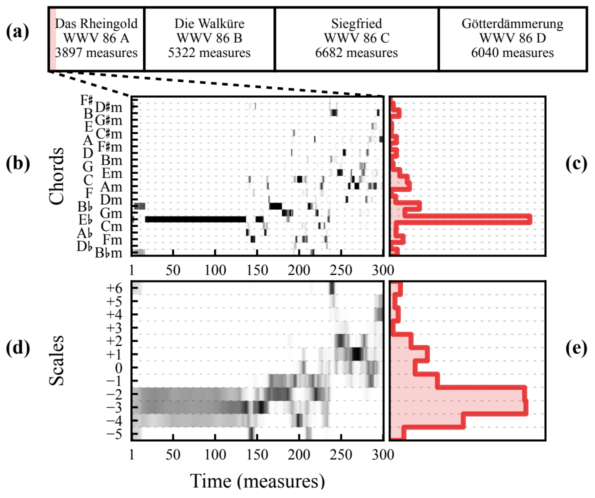


Figure 1. Overview of Wagner's *Ring* and schematic description of the histogram extraction. **(a)** The four parts of the *Ring* with catalogue number and length in measures. The measures under consideration are highlighted in color. **(b)** A local chord representation with time given in measures. **(c)** A histogram summarizing the chord representation. **(d)** A local scale representation. **(e)** A histogram summarizing the scale representation.

One of the most impressive realizations of these ideas is the tetralogy *Der Ring des Nibelungen*, an extensive work cycle of four music dramas created between 1848 and 1874. In Figure 1a, we show an overview of the *Ring*'s parts. A typical performance lasts 14–15 hours in total, which is demanding for listeners as well as performers. Furthermore, the through-composed form may appear less structured to the naive listener than a traditional number opera. Therefore, navigation and visualization tools are particularly useful for exploring this large-scale work. In this paper, we present such visualizations and demonstrate their benefit for musicological research.

The *Ring* has already obtained some attention in the field of Music Information Retrieval (MIR). Page et al. [9] present a toolkit for annotating musical performances in a case study based on the *Ring*. In other studies [17, 21], Wagner's tetralogy serves as a basis for investigating the reliability of measure annotations. Concerning leitmotifs, Müllensiefen et al. [7] consider the human memory recall task. They found that the distance of chroma features relates to the perceived novelty of a leitmotif.

In this paper, we approach the *Ring* from the perspective of tonal analysis. To this end, we perform experiments on the basis of common tonal features extracted from different recordings. These audio features capture the local presence of chord [12] and scale structures [18]. Figure 1 illustrates such tonal representations with respect to major and minor triads (Figure 1b) and to diatonic scales [1] (Figure 1d) for the beginning of the first part, *Das Rheingold*. Dark values indicate a higher salience of the respective tonal structures. The prelude of this piece strongly relies on a single E♭ major triad, which corresponds to homogeneous regions in both feature sequences. In general, Wagner's music is known to present a rich vocabulary of different chord and scale types. However, though being an oversimplification, in this case study we consider only the major and minor chords as well as diatonic scales. These simple tonal structures still explain a relevant span of harmonically stable passages [19]. As a central technique in this paper, we summarize such chord and scale feature sequences in histogram representations as shown in Figures 1c/e. Visualizing these histograms may illustrate global trends in the tonal conception of the music. For computing the histograms, we select and compare passages associated with dramatic aspects, for example, the activity of certain characters or the presence of certain leitmotifs. This way, we show the ability of our visualizations to explore interesting tendencies and to study relationships between tonal and dramatic aspects within the *Ring* cycle.

In the field of MIR, histogram-based features have been extensively used for tasks such as genre recognition [13], tuning estimation [5], and other classification tasks [10, 14]. Moreover, the visualization of music pieces is an important topic in MIR. Wu and Bello [20] present an approach for visualizing musical structure. Sapp's scape plot representations [11] have come out useful for illustrating harmony analysis results in a hierarchical way. In [3], Gómez and Bonada demonstrate several visualization techniques concerning tonal aspects of musical pieces.

Typically, tonal analysis is performed on the basis of musical scores. In Section 2, we discuss why an analysis based on audio recordings may be beneficial for the *Ring* scenario. Having several recorded performances (versions) of the *Ring* allows us to employ a cross-version approach in order to stabilize the audio-based tonal representations. Konz et al. [6] show that visualizing the cross-version consistency of analysis results suppresses aspects of particular performances and, thus, emphasizes aspects relevant to the musical work in general.

Based on previously mentioned works, we present the histogram visualizations as a novel way to explore the

[1] We refer to the diatonic scales according to the respective accidentals. For example, +1 corresponds to a scale with 1♯ (G major or E minor scale) whereas −2 indicates a scale with 2♭ (B♭ major or G minor scale).

No.	Conductor	Recording	hh:mm:ss
1	Barenboim	1991–92	14:54:55
2	Boulez	1980–81	13:44:38
3	Böhm	1967–71	13:39:28
4	Furtwängler	1953	15:04:22
5	Haitink	1988–91	14:27:10
6	Janowski	1980–83	14:08:34
7	Karajan	1967–70	14:58:08
8	Keilberth/Furtwängler	1952–54	14:19:56
9	Krauss	1953	14:12:27
10	Levine	1987–89	15:21:52
11	Neuhold	1993–95	14:04:35
12	Sawallisch	1989	14:06:50
13	Solti	1958–65	14:36:58
14	Swarowsky	1968	14:56:34
15	Thielemann	2011	14:31:13
16	Weigle	2010–12	14:48:46

Table 1. Performances of the *Ring* used for this paper. In No. 8, Furtwängler only conducts *Die Walküre* (different from No. 4), the other parts are conducted by Keilberth.

Ring. The main contribution is the application of MIR techniques in an exploratory manner for highlighting interesting trends and relations within this large-scale work cycle. The remainder of the paper is structured as follows. In Section 2 we explain the data set and describe the characteristics of our annotations. Then, we shortly recapitulate the extraction of local chord and scale information and explain the histogram computation (Section 3). In the central Section 4, we discuss these histograms in detail by means of several concrete examples relating to different dramatic aspects of the *Ring*. Section 5 concludes our paper.

2. DATA SET AND ANNOTATIONS

For an automated tonal analysis, we have to rely on a specific representation of a piece of music. Typically, musicologists perform such analyses in a manual fashion on the basis of musical scores. To automate this process, musical scores need to be accessible in a machine-readable form (symbolic data) in high quality, which is a rare case for Western classical music with a large instrumentation. To overcome this problem, Optical Music Recognition (OMR) techniques are usually employed, which cannot provide satisfactory results in many situations [2] so that time-consuming manual correction steps are required. As an alternative, tonal analysis can be performed on the basis of audio recordings, at least to a certain extent [12, 18]. For the *Ring*, for example, a high number of CD recordings are easily available. In our experiments, we use 16 different performances comprising nearly 232 hours of audio, see Table 1. To compare and combine analysis results obtained from different representations, a link between the representations' time axes is beneficial. For instance, we can use the positions of measure boundaries, as specified by the score, in the recordings. To this end, several students with a strong practical experience in Western classical music manually annotated these positions for three performances [17]. By means of synchronization techniques [8, Chap. 3], we jointly transferred the manual annotations from the three performances to all other performances. See [21] for details and an evaluation regarding the joint transfer. The measure positions constitute suitable reference points for a cross-version analysis on the measure level since different performances can be related to each other using a musical time axis. In total, the *Ring* encompasses 21941 measures, including pickup measures.

Moreover, the measure positions enable the transfer of semantic annotations from a musical time axis to the individual performances and vice versa. In our scenario, we are interested in dramatic aspects of the *Ring*'s plot. For example, we annotated the regions where different characters are singing. In particular, we specified the start and end of the singing voice regions as well as the corresponding character according to the verses of the libretto. If a verse is interrupted by a whole measure or more, the annotation is split accordingly. In total, our annotations comprise 6792 singing voice regions.

Beyond this, the presence of certain leitmotifs is of high interest for studying the dramatic conception. Motivated by this, a musicologist annotated the occurrences of the leitmotifs by listening to a recording and analyzing the sheet music. The definition of leitmotifs used for this work relies on a guide by Julius Burghold from 1913 [16], which serves as a reference both for the names and the musical shape of the motifs. The annotations comprise start and end positions as well as the corresponding name for each occurrence of a leitmotif. For example, within *Siegfried* 2632 leitmotif occurrence regions have been identified.

3. HISTOGRAM COMPUTATION

In this section, we summarize the computation of local chord and scale representations, describe our histogram approach, and show why a cross-version strategy is beneficial. To capture tonal characteristics in audio recordings, we first compute normalized chroma features [8, Chap. 3], which represent the energy within the twelve chromatic pitch classes over time. Employing a common template matching strategy, we locally compare the chroma feature sequence with binary templates corresponding to chords [12] or scales [18]. By means of normalizing, we can interpret the results as probability for the occurrences of the particular chords or scales. Since these tonal structures relate to a time span of several seconds, we suitably smooth the chromagram before applying template matching. To obtain musically meaningful windows, we make use of the measure annotations to obtain performance-independent window sizes $w \in \mathbb{N}$ specified in measures, rather than seconds. Our experiments showed that $w = 4$ for the chord analysis and $w = 12$ for the scale analysis provides meaningful visualizations. From a traditional music theory perspective, a window size of 4 measures for analyzing chords does not make sense. Indeed, chords lasting for such a long time rarely occur in Wagner's music. However, such a parameter setting leads to visualizations, which appear more structured, e.g. emphasizing tonic chords. One reason is that many chords in a progression share common notes and, thus, often stabilize the result for the respective tonic chord. Since we use a centered window view and a hop size of one measure, we obtain one feature vector for each measure. For emphasizing the locally salient structures, we apply an exponential post-processing step similar to the softmax function.

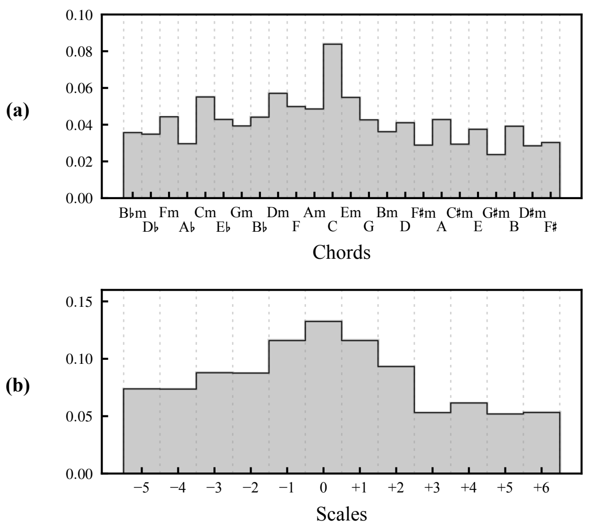


Figure 2. Analysis for the complete *Ring* cycle. **(a)** Chord histogram. **(b)** Scale histogram.

With this strategy, we compute an analysis matrix $\mathcal{A} \in \mathbb{R}^{D \times M}$ for a given performance, where $M \in \mathbb{N}$ denotes the number of measures and $D \in \mathbb{N}$ the feature dimension (with $D = 24$ for chords and $D = 12$ for scales). Figures 1b/d show such matrices. Based on this matrix $\mathcal{A}$, we calculate a histogram $\mathbf{h} \in \mathbb{R}^D$ by averaging over all measures:

$$\mathbf{h}(d) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{A}(d, m) \qquad (1)$$

with $d \in [1 : D] := \{1, 2, \ldots, D\}$. Figure 2 shows histograms for the complete *Ring*. The bar heights correspond to the presence of the chords or scales averaged over all measures. The distributions are rather flat, which indicates that Wagner seems to use the full range of chords and keys for tonally shaping his tetralogy. Nevertheless, we observe a stronger presence of the C major chord as well as the scales 0, +1, and −1 indicating that tonal regions with few accidentals seem to be slightly more prominent. Furthermore, we find a small trend towards flat key regions (left half) compared to sharp key regions (right half).

At this point, we may wonder about the reliability of these results and the histograms' dependency on a specific version. For example, Figure 3a shows two chord histograms, which are computed for two different performances. Even though the global trends seem to be consistent among the two versions, one can observe some deviations. For example, the E minor triad seems to be one of the most prominent chords in the Boulez version (blue), whereas this chord is less important in the Furtwängler version (red). Such performance-specific characteristics may come into play for a variety of reasons. For example, performances may exhibit a different dynamic balance of the instruments and singers. Furthermore, different recording conditions may suppress or enhance certain frequencies.

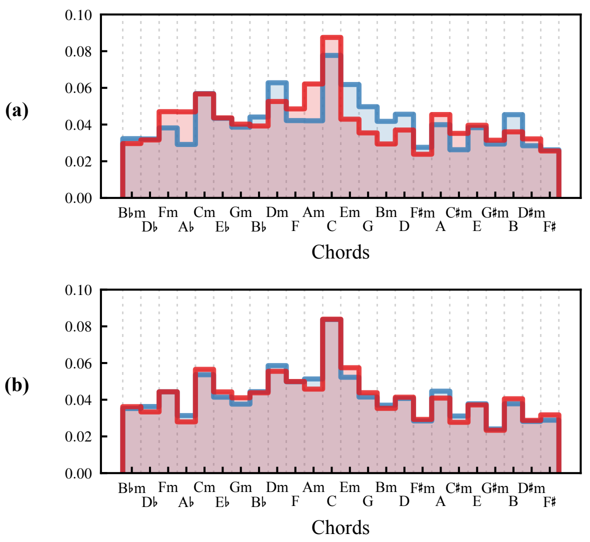


Figure 3. Comparison of single-version histograms and cross-version histograms for the complete *Ring*. **(a)** Chord histograms for the performances conducted by Boulez (No. 2) in blue and Furtwängler (No. 4) in red. **(b)** Chord histograms with cross-version approach. The blue and the red histogram each correspond to eight different versions.

To attenuate the version-specific aspects, we introduce a cross-version approach similar to [6]. Having $P \in \mathbb{N}$ performances, we compute an analysis matrix $\mathcal{A}_p$ for each of these versions $p \in [1:P]$. From these matrices, we derive a cross-version analysis matrix simply by averaging over all performances

$$\mathcal{A}^{\mathrm{cv}}(d,m) = \frac{1}{P}\sum_{p=1}^{P}\mathcal{A}_p(d,m) \tag{2}$$

with $d \in [1:D]$ and $m \in [1:M]$. Finally, we average over all measures of $\mathcal{A}^{\mathrm{cv}}$ to obtain a histogram as in Eq. 1.

Figure 3b shows two histograms computed with our cross-version approach, each for $P = 8$ different performances. [2] Note that the two cross-version histograms are more similar to each other than the two single-version histograms in Figure 3a. This indicates that the cross-version approach stabilizes the results. Furthermore, characteristic trends and peaks of the histograms are retained, which shows that the averaging procedure does not smooth out interesting details. In general, the cross-version approach enhances work-related aspects and suppresses performance-specific artifacts. In the following, all histograms are computed in a cross-version fashion using $P = 16$ performances. Also, the histograms in Figure 2 were computed in this way.

4. EXPLORATION

We now want to show the potential of the introduced histograms for exploring relationships between dramatic aspects of the *Ring* and its tonal organization. To this end, we

[2] Performances Nos. 1, 2, 5, 9, 11–13, 15 correspond to blue and performances Nos. 3, 4, 6–8, 10, 14, 16 correspond to red.

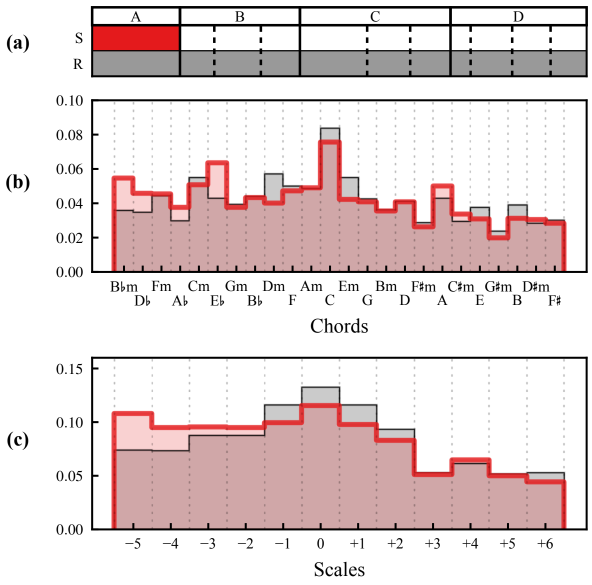


Figure 4. Comparison of *Das Rheingold* to the complete *Ring*. **(a)** Activations for the *Ring* (reference, 21941 measures, gray) and *Das Rheingold* (selection, 3897 measures, red). Dotted lines correspond to the beginning of a new act. **(b)** Chord histogram. **(c)** Scale histogram.

apply a selection procedure with respect to different criteria. As examples of such criteria, we consider the role of a single part with respect to the full cycle, the behavior of singing and instrumental regions, as well as the activity of specific characters. Furthermore, we take the occurrence of certain leitmotifs into account. For all examples, we select the measures or passages fulfilling these criteria while discarding the others before computing the histograms.

As a first scenario, we looked at the role of individual parts of the *Ring*. For example, we compared the first part, *Das Rheingold*, to the complete cycle. In Figure 4a, we show a compact overview that illustrates the investigated passages with the complete cycle as reference (R) in gray and the first part as selection (S) in red. In the following, we refer to such illustrations as activation diagrams. Using this color scheme, Figure 4b shows two chord histograms with the reference histogram in gray and the selection histogram in red. In these histograms, we observe rather flat distributions. Regarding the individual chords, we observe a high presence of the C major chord in *Das Rheingold* (selection). This coincides with the general distribution in the full *Ring* (reference). In contrast, the enhanced presence of the E♭ major chord in the selection deviates from the shape of the reference histogram. An important reason for this peak may be the prelude comprising 136 measures with a constant harmony, an E♭ major triad. Figure 4c shows the corresponding histograms for the scales. The comparison of these histograms indicates a trend towards scales with flats (♭) in their key signatures (–2 to –5) in the selection. These observations suggest that the individual parts of the *Ring* may indeed exhibit a characteristic tonal shape.

As a second example, we examine the characteristics of passages involving instrumental passages and singing

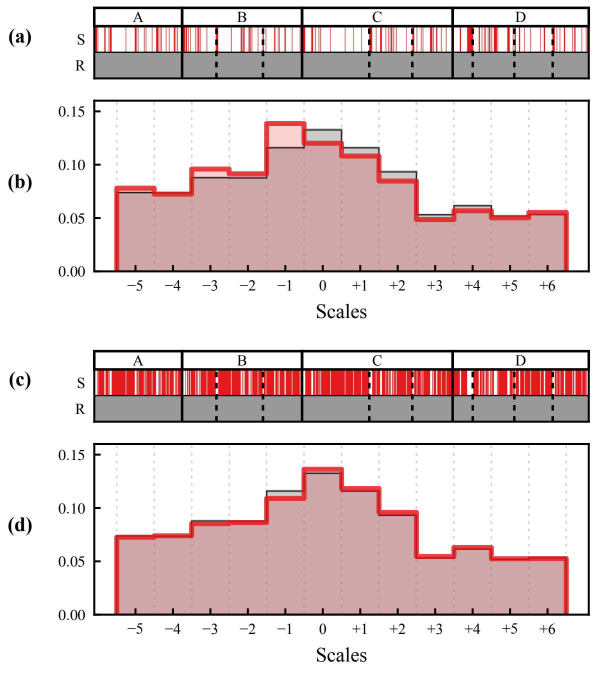


Figure 5. Comparison of instrumental passages to all others. **(a)** Activations for the *Ring* (reference, 21941 measures, gray) and the instrumental passages (selection, 5103 measures, red). **(b)** Scale histogram. **(c)** Activations for the *Ring* (reference, 21941 measures, gray) and passages involving singing (selection, 16838 measures, red). **(d)** Scale histogram.

activity. In contrast to the previous experiment, our selections now comprise several local passages instead of one contiguous region. To obtain two disjoint selections, we assign each measure to either the instrumental or the singing selection where measures with partial singing were assigned to the singing selection. Figures 5a/c display the corresponding activation diagrams. The aggregated lengths of both subsets are in a proportion of roughly 1 : 3 (5103 vs. 16838 measures). In the scale histograms in Figures 5b/d, the selections show a high similarity to the full cycle's histogram even though the respective passages do not overlap. This indicates that, globally speaking, there is no substantial tonal difference between these selections. One may wonder if this similarity is a trivial observation. As we will show in our next example, we observe different shapes when we investigate the singing activity of specific characters.

As the third scenario, we examine the behavior for two groups of characters, *gods* and *mortals*, which constitute central categories in the *Ring*'s plot.[3] The activation diagram in Figure 6a shows the singing activities of gods (in blue) and mortals (in red). Instead using a reference, like the complete *Ring* as before, we now compare two selections based on different selection criteria. In *Das Rheingold*, only gods are active whereas in the final *Götterdämmerung*, gods are barely active (only the Norns in the prologue). Thus, we observe some kind of global trend from gods towards mortals over the course of the *Ring*. Figures 6b/c show the corresponding chord and scale histograms. The mortals' scale histogram exhibits a slight trend towards scales with few accidentals. In contrast, the gods' histogram shows a somewhat higher presence of outlying scales. For musicologists, such observations could be a starting point for relating tonal characteristics to the interpretation of the drama. As an exemplary hypothesis, the prominence of far-off scales might be associated with the gods living far-off our human world.

As a final selection criterion, we focus on the occurrence of leitmotifs, which constitute a central dramatic element in the *Ring*. Even though Wagner did not invent this technique and never used the term "leitmotif" personally, the extensive usage of such motifs makes the *Ring* a prominent example of a realization of this concept. We now indicate how one may explore the tonal characteristics during the occurrences of certain leitmotifs. As a first example, we consider the "Valhalla motif," which refers to the castle of the gods and is frequently used over the course of the tetralogy. Figures 7a/b show an activation diagram and a chord histogram with *Das Rheingold* as reference and all regions in this part with the Valhalla motif as selection. For this motif, we notice a high peak for the D♭ major chord, in contrast to the flat shape of the reference histogram. When we examine this motif over the course of *Die Walküre*, we observe a different trend (see Figures 7c/d). For this part, the histogram exhibits a high peak for the E major chord whereas the D♭ major chord has only a slight peak. These

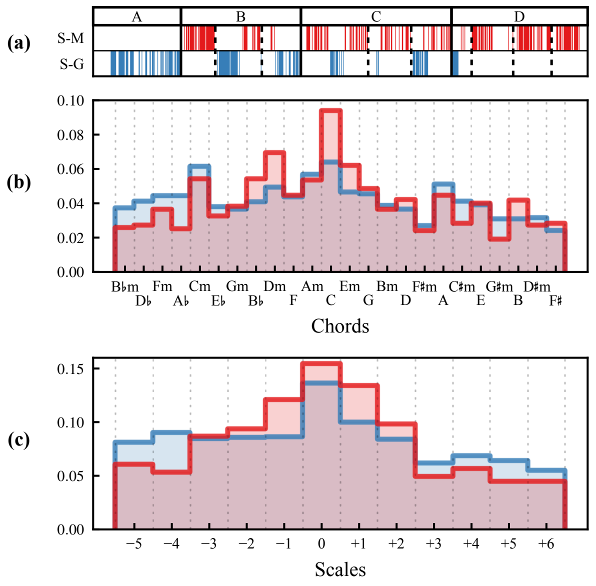
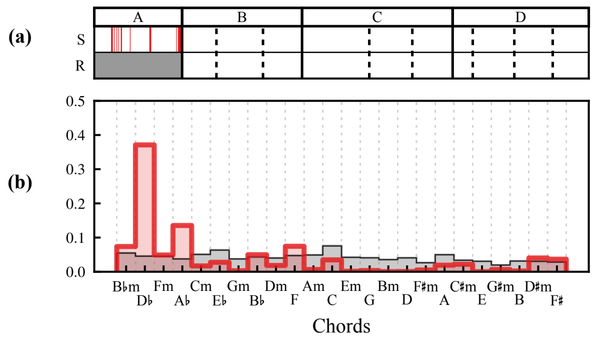


Figure 6. Comparison of gods and mortals singing. **(a)** Activations for passages with mortals singing (S-M, 6409 measures, red) and passages with gods singing (S-G, 4203 measures, blue). **(b)** Chord histogram. **(c)** Scale histogram.

[3] For this exemplary scenario, we classified as gods: Wotan, Fricka, Freia, Donner, Froh, Erda, Loge, and the Norns. Characters classified as mortals are Siegmund, Sieglinde, Siegfried, Hunding, Gunther, Gutrune, Hagen, as well as the male and female choirs. Even though there are border cases such as the demigod Loge or Siegmund and Sieglinde, Wotans children, we consider this a meaningful categorization. We do not take into account other categories such as the Valkyries, or the Nibelungs.



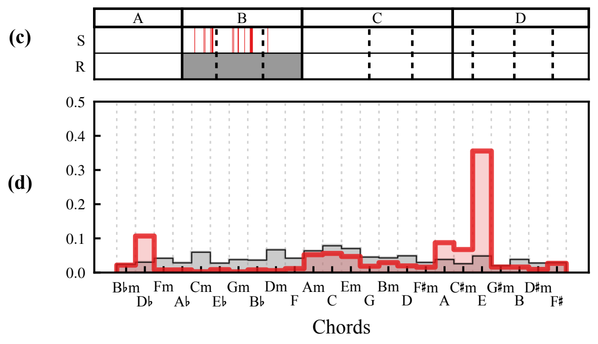
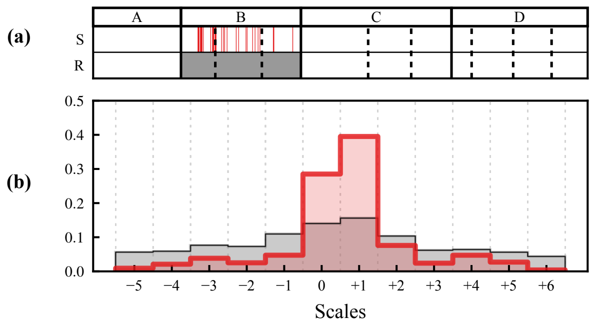


Figure 7. Passages containing the Valhalla motif. **(a)** Activations for *Das Rheingold* (reference, 3897 measures, gray) and passages containing the Valhalla motif therein (selection, 176 measures, red). **(b)** Chord histogram. **(c)** Activations for *Die Walküre* (reference, 5322 measures, gray) and passages containing the Valhalla motif therein (selection, 124 measures, red). **(d)** Chord histogram.



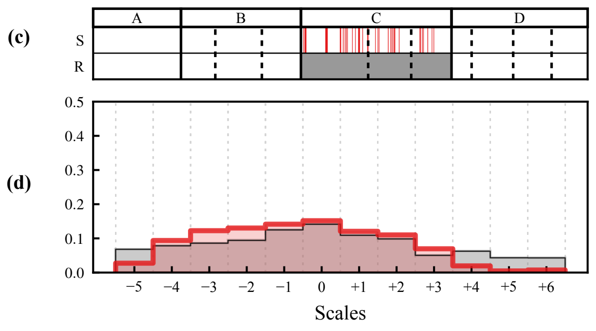


Figure 8. Passages containing the sword motif. **(a)** Activations for *Die Walküre* (reference, 5322 measures, gray) and passages containing the sword motif therein (selection, 200 measures, red). **(b)** Scale histogram. **(c)** Activations for *Siegfried* (reference, 6682 measures, gray) and passages containing the sword motif therein (selection, 203 measures, red). **(d)** Scale histogram.

observations indicate that the Valhalla motif tends to appear in a specific tonal context. However, the corresponding chord strongly differs between the parts of the *Ring*. The correlation of the Valhalla motif to the D♭ major and E major chords, respectively, is a known fact in musicology [4, p. 172]. It is promising that automated methods can confirm such observations.

As a second leitmotif example, we investigate the occurrences of the “sword motif.” Figure 8 shows activation diagrams and scale histograms for this motif within *Die Walküre* and *Siegfried*. In *Die Walküre*, the motif has a clear tendency towards the region of 0 (C major/A minor) and +1 (G major/E minor). The situation is completely different in *Siegfried*, where the scale distribution is rather flat, with a slight trend towards flat scale regions. One reason might be the integration of this motif into a more complex tonal conception in *Siegfried* compared to *Die Walküre*.

5. CONCLUSIONS AND OUTLOOK

In this paper we demonstrated how existing MIR techniques such as tonal audio features and global histograms can be applied in a complex music scenario. Regarding audio-based analysis, we showed that a cross-version approach is able to enhance work-related properties while suppressing performance-specific details and, therefore, stabilizes the analysis results compared to a single-version approach. For exploring relationships between dramatic aspects and the tonal organization of Wagner’s *Ring* cycle, we presented compact visualizations of tonal features based on global histograms with several selection criteria. We showed that these visualizations can provide interesting insights into large-scale works such as Wagner’s tetralogy. Investigating a small selection of examples, we found that the *Ring*’s parts may each exhibit an individual tonal shape. Furthermore, the histograms indicated that character groups can have different tonal preferences. Finally, we showed that leitmotifs can have specific tonal connotations. In general, using global histograms exhibits some limitations since we cannot address many fine-granular issues with this method. Nevertheless, we showed the benefit of such visualizations for highlighting interesting trends and relations. Beyond the analyses shown in this paper, many more filtering criteria could be of interest in this complex scenario. Concerning subsections of the *Ring*, individual acts or scenes could be analyzed. Regarding the musical parts, different character groups or even individual singers could be considered as well as the use of certain instruments or instrument families. Finally, the enormous number of leitmotifs with several thousand occurrences and their complex relationships lead to a vast amount of possible selections. Investigating the *Ring* with respect to such aspects could allow musicologists to confirm or adjust their hypotheses or be inspired to create new ones and, thus, might have potential for musicological research.

Acknowledgments: We thank Julia Zalkow, Vlora Arifi-Müller, and all students for their assistance in generating the annotations and preparing the data. This work has been supported by the German Research Foundation (DFG MU 2686/7-1). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

6. REFERENCES

[1] Howard Mayer Brown, Ellen Rosand, Reinhard Strohm, Roger Parker, Arnold Whittall, Roger Savage, and Barry Millington. Opera. In Stanley Sadie, editor, *The New Grove Dictionary of Music and Musicians*, pages 416–471. Macmillian Publishers, London, UK, 2nd edition, 2001.

[2] Donald Byrd and Jakob G. Simonsen. Towards a standard testbed for optical music recognition: Definitions, metrics, and page images. *Journal of New Music Research*, 44(3):169–195, 2015.

[3] Emilia Gómez and Jordi Bonada. Tonality visualization of polyphonic audio. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 57–60, Barcelona, Spain, 2005.

[4] Tobias Janz. *Klangdramaturgie: Studien zur theatralen Orchesterkomposition in Wagners „Ring des Nibelungen"*, volume 2 of *Wagner in der Diskussion*. Königshausen & Neumann, Würzburg, Germany 2006.

[5] Gopala Krishna Koduri, Joan Serrà, and Xavier Serra. Characterization of intonation in carnatic music by parametrizing pitch histograms. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 199–204, Porto, Portugal, 2012.

[6] Verena Konz, Meinard Müller, and Rainer Kleinertz. A cross-version chord labelling approach for exploring harmonic structures—a case study on Beethoven's Appassionata. *Journal of New Music Research*, 42(1):61–77, 2013.

[7] Daniel Müllensiefen, David Baker, Christophe Rhodes, Tim Crawford, and Laurence Dreyfus. Recognition of leitmotives in Richard Wagner's music: An item response theory approach. In *Analysis of Large and Complex Data*, pages 473–483. Springer, Cham, Switzerland, 2016.

[8] Meinard Müller. *Fundamentals of Music Processing*. Springer, Cham, Switzerland, 2015.

[9] Kevin Page, Terhi Nurmikko-Fuller, Carolin Rindfleisch, Richard Lewis, Laurence Dreyfus, and David De Roure. A Toolkit for Live Annotation of Opera Performance: Experiences Capturing Wagner's Ring Cycle. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 211–217, Málaga, Spain, 2015.

[10] Hendrik Purwins, Benjamin Blankertz, Klaus Obermayer, and Guido Dornhege. Scale degree profiles from audio investigated with machine learning. In *Proceedings of the 116th Audio Engineering Society (AES) Convention*, Berlin, Germany, 2004.

[11] Craig Stuart Sapp. Visual hierarchical key analysis. *ACM Computers in Entertainment*, 3(4):1–19, 2005.

[12] Alexander Sheh and Daniel P.W. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, pages 185–191, Baltimore, USA, 2003.

[13] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[14] George Tzanetakis, Andrey Ermolinskyi, and Perry Cook. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2):143–152, 2003.

[15] Richard Wagner. *Oper und Drama*. Klaus Kropfinger (Ed.). Reclam, Ditzingen, Germany, 1986.

[16] Richard Wagner. *Der Ring des Nibelungen. Vollständiger Text mit Notentafeln der Leitmotive*. Julius Burghold (Ed.). Schott Music, Mainz, Germany, 2013.

[17] Christof Weiß, Vlora Arifi-Müller, Thomas Prätzlich, Rainer Kleinertz, and Meinard Müller. Analyzing measure annotations for western classical music recordings. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 517–523, New York, USA, 2016.

[18] Christof Weiß and Julian Habryka. Chroma-based scale matching for audio tonality analysis. In *Proceedings of the 9th Conference on Interdisciplinary Musicology (CIM)*, pages 168–173, Berlin, Germany, 2014.

[19] Christof Weiß, Rainer Kleinertz and Meinard Müller. Möglichkeiten der computergestützten Erkennung und Visualisierung harmonischer Strukturen – eine Fallstudie zu Richard Wagners *Die Walküre*. In *Bericht zur Jahrestagung der Gesellschaft für Musikforschung (GfM)*, Mainz, Germany, 2016.

[20] Ho-Hsiang Wu and Juan P. Bello. Audio-based music visualization for music structure analysis. In *Proceedings of the Sound and Music Computing Conference (SMC)*, pages 496–501, Barcelona, Spain, 2010.

[21] Frank Zalkow, Christof Weiß, Thomas Prätzlich, Vlora Arifi-Müller, and Meinard Müller. A multi-version approach for transferring measure annotations between music recordings. In *Proceedings of the AES International Conference on Semantic Audio*, pages 148–155, Erlangen, Germany, 2017.

FEATURE DISCOVERY FOR SEQUENTIAL PREDICTION OF MONOPHONIC MUSIC

Jonas Langhabel[1] **Robert Lieck**[2,3] **Marc Toussaint**[2] **Martin Rohrmeier**[3,4]
[1] Department of Computer Science, TU Berlin, Germany
[2] Machine Learning and Robotics Lab, University of Stuttgart, Germany
[3] Systematic Musicology and Music Cognition, TU Dresden, Germany
[4] Digital and Cognitive Musicology Lab, EPFL, Switzerland
langhabel@campus.tu-berlin.de, robert.lieck@ipvs.uni-stuttgart.de,
marc.toussaint@ipvs.uni-stuttgart.de, martin.rohrmeier@epfl.ch

ABSTRACT

Learning a model for sequential prediction of symbolic music remains an open challenge. An important special case is the prediction of pitch sequences based on a corpus of monophonic music. We contribute to this line of research in two respects: (1) Our models improve the state-of-the-art performance. (2) Our method affords learning interpretable models by discovering an explicit set of relevant features. We discover features using the *Pulse* learning framework, which repetitively suggests new candidate features using a generative operation and selects features while optimizing the underlying model. Defining a domain-specific generative operation allows to combine multiple music-theoretically motivated features in a unified model and to control their interaction on a fine-grained level. We evaluate our models on a set of benchmark corpora of monophonic chorales and folk songs, outperforming previous work. Finally, we discuss the characteristics of the discovered features from a musicological perspective, giving concrete examples.

1. INTRODUCTION

Predictive processing and the formation of expectancies are core capacities of human cognition, that are closely tied to the perception and interaction with our environment and to survival and fitness in an evolutionary perspective. Apart from its role in most cognitive domains, predictive processing has also been understood to play a fundamental role in music perception [22, 29]. The formation of musical expectancies is essential for goal directed processes at different musical time-scales, for musical interaction and synchronization as well as for the play with emotional effects in music [10, 17], and particularly, musical tension [8, 13]. Musical expectancy has also been understood to be culture- and style-dependent and to be grounded in musical knowledge that is acquired through processes of implicit or statistical learning [10, 31, 32]. Modeling of human predictive processing in music is thus fundamental for computational cognitive models of music as well as for models of musical interaction or generation.

Musical expectancy has been studied in terms of melody, harmony and rhythm. While the task involves the prediction of the next note, onset, chord or combinations thereof given the past events in the sequence, there are many different types of underlying models one can posit to compute predictions and event probabilities. While the setting of predicting the next event is difficult to define accurately and to tackle in the general case of polyphonic music, many past approaches have simplified the problem to tackle a single stream of events, such as melodic notes or chord events. Because this context is similar to the linguistic case, one frequent approach has been to take into account language models as commonly used in computational linguistics, particularly, n-gram models and derived models, which are discussed in Sec. 1.1. More recently, connectionist models have also become increasingly popular, as discussed in Sec. 1.3.

Apart from n-gram and connectionist models musical expectancy and melodic structure have been modelled by other kinds of approaches that we do not discuss in detail. These involve, most notably, hidden Markov models for melodic structure (e.g. [15, 16]) and dynamic Bayesian networks (e.g. [19, 20, 27]). More generally, a large variety of latent structure models beyond Markovian approaches may be adequate to characterize prediction and to compute sequential event probabilities.

1.1 n-gram Models

Markovian and similar approaches have been applied since decades for the modeling of music (see [22] for a review). n-gram models track the number of times a particular contiguous sub-sequence of events occurs in the data. In simple n-gram models the predictive distribution is computed by normalizing these frequency counts for a fixed context length n. Such n-gram models have been applied particularly in the modelling of melody [5, 21, 24] and harmony

[26, 30, 33, 36, 37], classification [4, 9], or applications such as style description or identification [18, 28].

A major problem with this approach is that short context lengths are unable to capture longer patterns while longer context lengths overfit on the training data by assigning zero probability to unseen sequences. Two common methods to overcome this problem are (1) *escaping* strategies, which explicitly assign non-zero weights to unseen sequences and (2) *smoothing* methods, which combine multiple n-gram models of different, possibly unbounded context length (see [24] for an extensive overview).

Extending common n-gram models, Conklin and Witten [5] proposed the notion of multiple viewpoint systems, which combine n-gram models over different basic and derived features in order to improve melodic prediction taking into account correlations between different features. Pearce [21] extended this idea forming the basis of IDyOM, a cognitive model of melodic prediction and generation, which was evaluated with human psychological data [23].

1.2 Long-Term and Short-Term Model

Conklin and Witten [5] proposed the distinction between a long-term model (LTM) and a short-term model (STM). The LTM is trained on a corpus of data and is supposed to capture piece-independent characteristics of the corresponding style, epoch, or genre. The STM, on the other hand, is supposed to capture properties of a single piece like motives or repetitions and is trained online at prediction time for each piece separately. LTM and STM are combined at prediction time (see Sec. 2.6 for details).

1.3 RTDRBM (Connectionist Models)

Recently, Cherla et al. [3] used a *recurrent temporal discriminative restricted Boltzmann machine* (RTDRBM) as LTM, improving over the to-date best performing n-gram LTMs. RTDRBM are state-full connectionist models similar to recurrent neural networks (RNNs), which have the potential to capture long-term dependencies in time series data. This renders them particularly suited for the LTM.

2. THE *PULSE* FRAMEWORK FOR MUSIC

We employ the *Pulse* learning framework [14] for discovering relevant musical features. *Pulse* performs a guided search through an infinitely large feature space and thereby allows to discover features in spaces that are too large for classical feature selection approaches. In doing so, *Pulse* iteratively performs (forward) feature expansion and (backward) feature selection, resulting in a framework similar to evolutionary algorithms. We will first describe the general principles of discovering features with *Pulse* in Sec. 2.1 before going into details about our specific implementation.

2.1 Discovering Features with *Pulse*

Pulse addresses the ubiquitous machine learning problem that, on the one hand, we need to include task-specific prior knowledge to efficiently solve a learning task but, on the other hand, explicitly specifying a set of features might neither be possible nor desirable for a number of reasons: (a) We may lack the explicit knowledge required to specify good features. (b) The specified features may be too specific and "overfit" on a single problem instance. (c) Explicitly specifying features is tedious work to be done by experts, which we might want to automate.

More precisely, instead of explicitly specifying features, in *Pulse* we specify a generative operation N^+ that suggests new candidate features based on the current feature set. N^+ may inject new features as well as mutate and recombine existing features, analogous to an evolutionary algorithm. After expanding the feature set by including all candidate features suggested by the N^+ operation, *Pulse* shrinks the feature set by optimizing the underlying model and selecting features based on the model performance. Again this is akin to evolutionary algorithms with the difference that *Pulse* defines an objective based on the whole feature set and features are thus not scored individually but selected based on how much they contribute to the fitness of the whole population. For learning an optimal set of features and parameters, *Pulse* repetitively expands the feature set by applying N^+ and selects a subset of features by optimizing the model parameters Θ and removing features with zero weight.

As the underlying model, *Pulse* uses a conditional random field [12], which defines a conditional probability distribution $p(x|y)$ as

$$p(x|y) = \frac{1}{Z(y)} \exp \sum_{f \in \mathcal{F}} \theta_f f(x, y) \tag{1}$$

$$Z(y) = \sum_{x' \in \mathcal{X}} \exp \sum_{f \in \mathcal{F}} \theta_f f(x', y) \ , \tag{2}$$

where $y \in \mathcal{Y}$ is known at prediction time, $x \in \mathcal{X}$ is to be predicted, $\mathcal{F}$ is the set of features with weights $\Theta = \{\theta_f \in \mathbb{R} \mid f \in \mathcal{F}\}$, and the partition function $Z(y)$ ensures correct normalization of the conditional distribution. The features $f \in \mathcal{F}$ may be arbitrary real-valued functions of x and y, $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. When modeling sequential data, $x \in \mathcal{X}$ is the next event, $y \in \mathcal{Y} \equiv \mathcal{X}^*$ is the sequence of past events, and $\mathcal{X}$ is called the symbol space or the alphabet. The parameters are optimized by performing (stochastic) gradient descent on the negative log-likelihood of the data

$$\ell(\Theta; D) = -\sum_{(x,y) \in D} \log p(x|y) + \rho(\Theta) \ , \tag{3}$$

where $\rho(\Theta)$ comprises any regularization terms, most notably an L_1-regularization to enforce a sparse feature set. Note that since ρ implements a prior over the model parameters, specifying additional regularization terms is another means to inject task-specific knowledge in addition to N^+ (also see Sec. 2.5). For optimization we use *AdaGrad* [7] combined with the approach described by Tsuruoka, Jun'ichi Tsujii, and Ananiadou [35] for implementing the L_1-regularization. We will interchangeably speak of maximizing the data likelihood or minimizing the model cross-entropy as both objectives are equivalent.

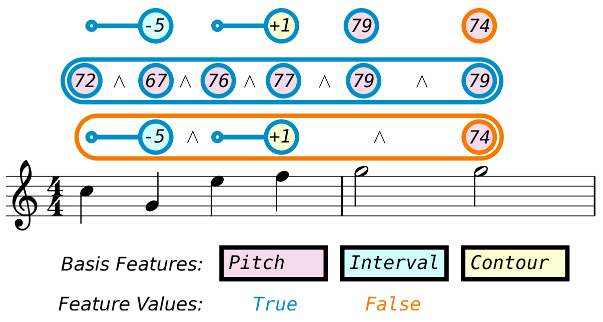


Figure 1. Visualization of generalized n-gram features constructed from three different viewpoints with basis features (top row), a classical n-gram feature (middle row), and a generalized n-gram feature (bottom row). See text for detailed explanation.

We use *Pulse* to discover two kinds of features: Viewpoint features (Sec. 2.2), which generalize the concept of classical n-grams, and Anchor features (Sec. 2.3), which incorporate the concepts of tonic and mode (key) that is common in tonal music.

2.2 Viewpoint Features

Viewpoint or n-gram features indicate the presence of a specific sequence of events. Compared to classical n-grams, the approach described in the following defines a more general set of features, namely ones that also include partial sequences (i.e. sequences with "holes") and sequences that mix different viewpoints. We sometimes differentiate between these generalized and classical non-generalized n-gram features.

Any viewpoint ξ implies a particular alphabet $\mathcal{X}_\xi$. We define an associated set of basis features B_ξ such that feature $b_{(x,t)}(s) \in B_\xi$ indicates whether symbol $x \in \mathcal{X}_\xi$ occurred at time $T-t$ in the given sequence $s \in \mathcal{X}_\xi^*$ of length T. The feature $b_{(x,t)}$ thus looks t steps into the past, indicating the occurrence of event x at that time, with $b_{(x,0)}$ referring to the next event to be predicted. In the upper part of Tab. 1 we list all viewpoints that we used for constructing n-gram features.

Picking up on the construction method suggested in [14] we construct more complex features via logical conjunction of multiple basis features. Features constructed in this way are more general than classical n-grams in two respects: (1) They do not necessarily contain a contiguous sequence of basis features, which allows to generalize over events at a specific time by leaving a "hole". (2) They may be composed of basis features derived from different alphabets/viewpoints, which allows to define a specific combination of viewpoints at one point in a sequence and ignore some of the viewpoints at others. In Sec. 2.4 we describe the specific generative N^+ operations that we use in detail.

An illustration of possible features constructed in this way is given in Fig. 1. Here, we used pitch, interval and contour (P, I, C in Tab. 1) as viewpoints. The last tone in the sequence, the rightmost G5, is to be predicted and has time index $t = 0$. As indicated by the colored border, features evaluate to `true` (or 1) if they match the data and to `false` (or 0) otherwise. In the top row, the basis features $b_{(I=-5,t=4)}$, $b_{(C=+1,t=2)}$, $b_{(P=79,t=1)}$ and $b_{(P=74,t=0)}$ are depicted. Note that interval and contour features not only depend on the pitch at time t but additionally on the previous pitch at time $t-1$. By concatenating basis features from a single viewpoint we can construct classical n-gram features, as illustrated for a pitch n-gram feature in the middle row, which indicates the pitch sequence $72, 67, 76, 77, 79, 79$. If we combine basis features from different viewpoints in a non-contiguous way, we end up with a generalized n-gram feature, as shown in the bottom row. This feature indicates a sequence that starts with a 5-semitone step down, followed by an arbitrary tone from which it rises by an arbitrary interval, again followed by an arbitrary tone, and finally terminates on a D5. As the actual sequence terminates on a G5 this feature evaluates to `false`/0 in this specific case.

2.3 Anchor Features

Anchor features allow to incorporate the concept of tonic and mode, that is key, into our model. They essentially are interval features where the value is not defined with respect to the previous tone but with respect to an anchor tone that may be computed based on any information available at prediction time.

We use three kinds of anchor features that introduce an increasing amount of prior knowledge about tonal music, as listed in the bottom part of Tab. 1. The F$_i$ features use the i^{th} tone of the current piece as reference tone, which is trivial to compute, does not change during the piece and ignores the mode. In many cases the tonic is among the first tones of a piece. A more sophisticated approach is to estimate the tonic based on all tones heard so far using the key-finding algorithm by Krumhansl [11] with parameters from [34]. This is realized in the T features, which may thus change during a piece (even though a change usually only occurs within the first couple of tones) but still ignore the mode. As the employed key-finding algorithm also estimates whether the piece is in major or minor mode it is straightforward to include this distinction, which is realized in the K features. It is interesting to note that in contrast to n-gram features, which have an arbitrary yet fixed length, anchor features incorporate information from the entire history dating back to the very first tone in a piece.

Just as with viewpoint features it is possible to form logical conjunctions of anchor features (anchored generalized n-gram features), however, in this work we confine ourselves to including only single (unigram) anchor features.

2.4 N^+ Operation

In this section we describe the different generative N^+ operations we use to search through the space of generalized n-gram features. The role of the N^+ operation is to suggest new candidate features that are included in the feature set if they improve the model (see Sec. 2.1). Our N^+ operations will inject new basis features (unigrams) and sug-

	Abbrev.	Name	Value Range	Description
Viewpoint Features	P	pitch	$\mathcal{P}$	MIDI pitch of the current note
	I	interval	$\mathcal{I}$	pitch difference between current and previous note
	C	contour	$\{-1, 0, 1\}$	sign of the interval
	X	extended contour	$\{-2, -1, 0, 1, 2\}$	like C but ± 2 for intervals larger than ± 5 steps
Anchor Features	F_i	i^{th} in piece	$\mathcal{I}$	pitch differences to the i^{th} tone in the current piece
	T	tonic	$\{0, \ldots, 11\}$	octave invariant pitch difference to the tonic
	K	key	$\{\mathrm{maj}, \mathrm{min}\} \times \{0, \ldots, 11\}$	like T but separate for major and minor keys

Table 1. Set of employed basis features. $\mathcal{P}$ corresponds to the prediction alphabet, that is, the set of possible MIDI pitches. $\mathcal{I} = \{a - b \,|\, a, b \in \mathcal{P}\}$ is the set of all possible intervals in $\mathcal{P}$.

gest new candidates by taking existing features and adding a new basis feature via logical conjunction. In general, we will apply a combination of multiple N^+ operations to learn our models.

More precisely, we write ξ to refer to an N^+ operation that adds all basis features $b_{(x,0)} \in B_\xi$ of the corresponding viewpoint for time zero to the feature set. Whether the symbol ξ refers to the viewpoint or the corresponding N^+ operation should be clear from context or is explicitly stated otherwise. Likewise, the operators F_i, T, and K add the corresponding anchor features to the feature set. We write ξ^* for an N^+ operation that expands existing features by adding another basis feature from the corresponding viewpoint. The ξ^* operation ignores features corresponding to viewpoints other than ξ. When applying the ξ^* operation we implicitly assume that the ξ operation is also applied (after ξ^*) without explicitly stating it.

Our ξ^* operations come in two versions, as *backwards expansion* for the LTM and as *forwards expansion* for the STM.

For *backwards expansion* (LTM), in the i^{th} iteration of feature expansion, ξ^* expands all existing features with all basis features $b_{(x,i)} \in B_\xi$ with time i. That is, if we were not to remove any features from the set, after n iterations of *backwards expansion*, ξ^* would have constructed all possible generalized n-gram features (with and without "holes") for alphabet $\mathcal{X}_\xi$.

For *forwards expansion* (STM), ξ^* first shifts all existing features by one time step to the past and then expands them with all basis features $b_{(x,0)} \in B_\mathcal{X}$ for time zero. If we were not to remove any features, after n iterations of *forwards expansion*, ξ^* would have constructed all possible *non-generalized* n-gram features (only those without "holes") for alphabet $\mathcal{X}_\xi$.

Backwards and forwards expansion take into account the different learning scenarios for LTM and STM. For the LTM all data is known from the beginning and backwards expansion successively suggests n-gram features of increasing context length until the model stops improving. In contrast, for the STM new data keeps coming in and we construct n-gram features on-the-fly by performing forward expansion once per time step (see Sec. 3). This ensures that (1) short n-gram features can be rebuilt from scratch to account for new data and (2) if an existing n-gram feature captures a motive in the piece, all possible continuations are considered as new candidate features in the next time step.

2.5 Regularization

The purpose of the regularization terms $\rho(\Theta)$ in the *Pulse* objective is twofold: (1) It limits growth of the feature set via L_1-regularization. (2) It implements a prior/bias, which shapes the model characteristics and is a means to prevent overfitting. We use a regularization of the form

$$\rho(\Theta) = \sum_{f \in \mathcal{F}} \left[|\theta_f| \rho_{L_1}(f) + |\theta_f|^2 \rho_{L_2}(f) \right], \quad (4)$$

where $\rho_{L_1}(f)$ and $\rho_{L_2}(f)$ compute the L_1 and L_2 regularization independently for each feature f. For the L_1-regularization in our LTM we follow the rationale that the further back a note lies in time, the less impact it has on the prediction of the current note. This means that longer context lengths risk to overfit on the training data and should be regularized more strongly. We did not observe a significant improvement from applying an additional L_2-regularization in the LTM and use

$$\text{LTM:} \quad \begin{aligned} \rho_{L_1}(f) &= \lambda_1 \, e^{\tau_f / \epsilon} & (5) \\ \rho_{L_2}(f) &= 0 \,, & (6) \end{aligned}$$

where τ_f is the temporal extent of feature f (i.e. the maximum time index of the basis features), ϵ determines how quickly the regularization kicks in for increasing temporal extent, and λ_1 determines the overall regularization strength. For the STM we use

$$\text{STM:} \quad \begin{aligned} \rho_{L_1}(f) &= \lambda_1 \, e^{-t/r_1} \, e^{\tau_f / \epsilon} & (7) \\ \rho_{L_2}(f) &= \lambda_2 \, e^{-t/r_2} \,, & (8) \end{aligned}$$

which implements the same idea with two crucial modifications: (1) The overall regularization strength decays exponentially as more data becomes available, where $r_{1/2}$ are the decay rates and t is the current time index in the song during online training of the STM. (2) We use an additional L_2-regularization, which impedes sparsity but was found to improve the STM performance especially in the initial phase.

The structure of these regularization functions was the result of preliminary runs. Parameters are chosen as described in Sec. 3.

2.6 Combining LTM and STM

LTM and STM are combined by computing a weighted arithmetic mean of their predictive distributions [25]

$$p(x|y) \propto \frac{\sum_{m \in M} w_m p_m(x|y)}{\sum_{m \in M} w_m} \tag{9}$$

where M is the set of available models, which in principle may contain more than just two models. The weights are computed based on how "certain" a given model is, as measured by the entropy of its predictive distribution

$$w_m = \left[\frac{-\sum_{x \in \mathcal{X}} \log p_m(x|y)}{\log |\mathcal{X}|} \right]^{-b}, \tag{10}$$

where division by $\log |\mathcal{X}|$ (the maximum possible entropy) ensures that weights are in $[0, 1]$, and $b \geq 0$ is a bias parameter that allows to shift weights towards models with lower entropy.

3. EXPERIMENTS

We evaluate our models on a corpus of eight symbolic music datasets, as used in [1–3, 24]. The corpus consists of 1009 monophonic folk melodies from different countries and styles as well as the soprano lines of 185 Bach chorales. The data is parsed from the ***kern* format using the Python toolkit *music21* [6]. As input for our model, the melodies are represented as monophonic chromatic pitch sequences with ties being merged. Performance is indicated by the model *cross-entropy* measured in bits. We perform 10-fold cross-validation on each corpus separately using the same folds as [2, 24]. The overall model performance is computed by first computing the cross-entropy for the test set in each cross-validation fold, then averaging over the folds within one corpus, and finally averaging over the different corpora (this is the same approach as in previous work).

Optimization of the feature weights is done using *AdaGrad* [7] with a constant learning learning rate of $\eta = 1$ and initial gradient squared accumulators of $g = 10^{-10}$.

LTM: We evaluate our LTM using the different N^+ operations listed in Tab. 2. Our best performing LTM is compared to the state-of-the-art (see Tab. 3). The feature set is expanded until less than 1% of the features change. The LTM hyperparameter ϵ was fixed to $\epsilon = 1/\ln(2) \approx 1.44$ in preliminary runs while λ_1 is optimized for every cross-validation fold by leaving out 10% of the training data as validation set and performing a Gaussian process based optimization for $\lambda_1 \in [10^{-9}, 10^{-6}]$ using the framework Scikit-Optimize [1].

STM: For the STM we combine the N^+ operations P, I* and F$_1$. The feature set is expanded once per time step followed by an optimization of the feature weights until convergence. The hyperparameters were set to the following values based on preliminary runs: $\lambda_1 = 10^{-5}$, $r_1 = 100$, $\epsilon = 1/\ln(1.2) \approx 5.48$, $\lambda_2 = 10^{-2}$, $r_2 = 8$.

[1] https://scikit-optimize.github.io

Pulse-LTM				
P	I*	C	–	2.701
		X*	–	2.692
		C*	–	2.692
			F$_1$	2.620
			F$_1$F$_2$F$_3$	2.602
			T	2.586
			K	**2.547**

Table 2. Performance of *Pulse*-LTM for different configurations.

	LTM	STM	Hybrid
Pulse	**2.547**	**3.094**	**2.395**
RTDRBM [3]	2.712	3.363	2.421
n-gram [24]	2.878	3.139	2.479

Table 3. Comparison of best performing *Pulse*, RTDRBM, and n-gram models.

Hybrid: For the hybrid model we combine our (PI*C*K)-LTM with our (PI*F$_1$)-STM. We also test the combination of our LTM with an (C*I) n-gram STM model from the IDyOM-framework [21]. The bias parameter b was determined over the grid $b \in \{0, 1, 2, 3, 4, 5, 6, 16, 32\}$ on the training set of each cross-validation fold.

4. RESULTS

The chief results are that

1. Our *Pulse*-LTM outperforms the current state-of-the-art, RTDRBM [2].
2. Our *Pulse*-STM outperforms the current state-of-the-art, X*UI n-gram [24].
3. Our LTM/STM-hybrid model outperforms the current state-of-the-art, RTDRBM/n-gram [3].
4. The discovered features and learned weights provide musically interpretable insights into the model.

We will now discuss these results in more detail.

4.1 LTM Configurations

In Tab. 2 we list the results for our different LTM configurations. In preliminary runs we identified PI*C to be the minimum setup for outperforming previous work. Performance is improved by expanding contour features (C $\rightarrow$ C*) in addition to intervals, enabling the model to learn melody contours in addition to transposition invariant motifs. Interestingly, the distinction of small and large intervals using extended contour features, X*, which is considered relevant in music theory, did not result in further improvement.

As expected, incorporating an increasing amount of prior knowledge about tonic and key via F$_1$, F$_1$F$_2$F$_3$, T, and K, respectively, led to significant improvements. Our best LTM configuration, PI*C*K, significantly improves over the current state-of-the-art, with the performance gain being of the same magnitude as was achieved by the current state-of-the-art RTDRBM [2] versus n-grams [24].

It is interesting to note that the algorithm for computing T and K involves a combination of music-theoretical insights and empirical tone profiles. An important future research question is how this can be generalized and made accessible to learning from corpus data.

4.2 STM Performance

In Tab. 3 we compare the PI*F_1 *Pulse*-STM with the best RTDRBM [3] and n-gram STM [24]. To our knowledge we present the first model that outperforms the well-established n-gram STM for the task of sequential prediction of symbolic monophonic music. We assume that an even better performance is achievable by performing a more thorough (yet very expensive) optimization of the STM hyperparameters.

4.3 Hybrid

Combining the *Pulse*-LTM with the *Pulse*-STM gives a performance boost, outperforming current state-of-the-art hybrids (see Tab. 3). The combination of our *Pulse*-LTM with an C*I n-gram STM from the IDyOM-framework yields an interesting result: While the n-gram STM alone performs worse (3.152 bits) than our *Pulse* STM the corresponding hybrid displays a better performance of 2.365 bits. We conjecture that the n-gram STM has complementary properties to our *Pulse*-based model and therefore is able to contribute valuable missing information. The best performing LTM/STM-hybrid on this corpus thus is the combination of our PI*C*K *Pulse*-LTM and the C*I n-gram STM.

4.4 Discussion of Features

While an extensive discussion of all features for the different corpora is beyond the scope of this paper, we show a qualitative plot of the weights for a subset of features learned by our PI*C*K *Pulse*-LTM from the Bach chorale corpus in Fig. 2. First, we see that the pitch features (P) describe a general preference for tones in the middle register. For the interval features (I) we restrict ourselves to length-one features, which show a preference of small (especially descending) steps over large steps. This is in accordance with general music-theoretic principles of voice leading. Note that a tritone step (±6 semitones) is particularly discouraged. The anchor features (K) model separate tone profiles for major (M) and minor (m) modes. We empirically confirm a preference for relevant in-scale tones: tonic (0), major third (4)/minor third (3) and fifth (7), whereas the out-of-scale tones minor second (1), minor third(3)/major third (4), tritone (6), and minor seventh (10)/major seventh (11) are discouraged.

During training of the model, a total of 5851 features was temporally included from which 322 remained in the final model. This underlines the relevance of performing both feature expansion *and* selection, which allows *Pulse* to scale to very large feature spaces.

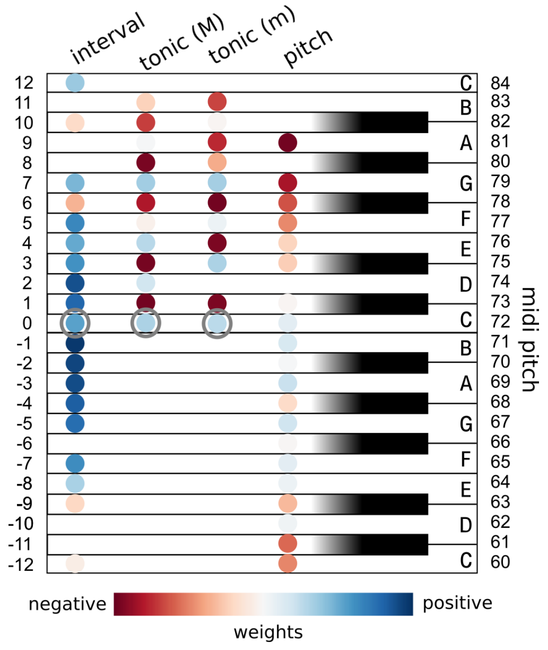


Figure 2. Qualitative plot of the feature weights for P and I features of length one as well as K features, learned based on the Bach chorales. For I and K features middle C is chosen as reference tone as marked by the circles.

5. CONCLUSION

We applied the *Pulse* framework to the problem of learning a model for sequential prediction of symbolic monophonic music. Our models outperform the current state-of-the-art for long-term, short-term and hybrid models on a standard benchmark corpus of folk melodies and Bach chorales. At the same time our approach affords interpretable models that use an explicit set of musically relevant features. The size of the processed feature spaces are challenging for classical feature expansion methods and our method has the potential to scale to even larger spaces. This becomes particularly relevant for an application to polyphonic music and modeling of harmony as well as for including more complex viewpoints.

This is the first application of the *Pulse* framework for modelling music, which provides excellent results and opens up a number of possible directions for further investigation. We therefore consider *Pulse* to be a promising framework for the development of a unified architecture for modelling music.

6. REFERENCES

[1] Srikanth Cherla et al. “A Distributed Model For Multiple-Viewpoint Melodic Prediction”. In: *International Society for Music Information Retrieval (ISMIR)*. 2013, pp. 15–20.

[2] Srikanth Cherla et al. “Discriminative learning and inference in the Recurrent Temporal RBM for melody modelling”. In: *Neural Networks (IJCNN), International Joint Conference on*. IEEE. 2015, pp. 1–8.

[3] Srikanth Cherla et al. “Hybrid Long-and Short-Term Models of Folk Melodies.” In: *International Society for Music Information Retrieval*. 2015, pp. 584–590.

[4] Darrell Conklin. “Multiple viewpoint systems for music classification”. In: *Journal of New Music Research* 42.1 (2013), pp. 19–26.

[5] Darrell Conklin and Ian H. Witten. “Multiple Viewpoint Systems for Music Prediction”. In: *Journal of New Music Research* 24.1 (1995), pp. 51–73.

[6] Michael Scott Cuthbert and Christopher Ariza. “music21: A toolkit for computer-aided musicology and symbolic music data”. In: *International Society for Music Information Retrieval (ISMIR)*. 2010, pp. 637–642.

[7] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[8] Morwaread M Farbood. “A parametric, temporal model of musical tension”. In: *Music Perception: An Interdisciplinary Journal* 29.4 (2012), pp. 387–428.

[9] Ruben Hillewaere, Bernard Manderick, and Darrell Conklin. “Global Feature Versus Event Models for Folk Song Classification.” In: *International Society for Music Information Retrieval (ISMIR)*. 2009.

[10] David Huron. *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, Massachusetts: MIT Press, 2006.

[11] Carol L. Krumhansl. *Cognitive Foundations of Musical Pitch*. Oxford Psychology 17. New York / Oxford: Oxford University Press, 1990.

[12] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289.

[13] Moritz Lehne et al. “The influence of different structural features on felt musical tension in two piano pieces by Mozart and Mendelssohn”. In: *Music Perception: An Interdisciplinary Journal* 31.2 (2013), pp. 171–185.

[14] Robert Lieck and Marc Toussaint. “Temporally Extended Features in Model-Based Reinforcement Learning with Partial Observability”. en. In: *Neurocomputing* 192 (2016), pp. 49–60.

[15] Panayotis Mavromatis. “A hidden Markov model of melody production in Greek church chant”. In: *Computing in Musicology* 14 (2005), pp. 93–112.

[16] Panayotis Mavromatis. “Minimum description length modelling of musical structure”. In: *Journal of Mathematics and Music* 3.3 (2009), pp. 117–136.

[17] L B Meyer. *Emotion and Meaning in Music*. London: University of Chicago Press, 1956.

[18] M. Ogihara and T Li. “N-Gram chord profiles for composer style identification”. In: *International Society for Music Information Retrieval (ISMIR)*. 2008, pp. 671–676.

[19] Jean-Francois Paiement, Samy Bengio, and Douglas Eck. “Probabilistic models for melodic prediction”. In: *Artificial Intelligence* 173.14 (2009), pp. 1266–1274.

[20] Jean-Francois Paiement, Yves Grandvalet, and Samy Bengio. “Predictive models for music”. In: *Connection Science* 21.2-3 (2009), pp. 253–272.

[21] Marcus T. Pearce. “The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition”. PhD thesis. Department of Computing, City University, London, UK, 2005.

[22] Marcus T Pearce and Geraint A Wiggins. “Auditory expectation: The information dynamics of music perception and cognition”. In: *Topics in cognitive science* 4.4 (2012), pp. 625–652.

[23] Marcus T Pearce and Geraint A Wiggins. “Expectation in melody: The influence of context and learning”. In: *Music Perception: An Interdisciplinary Journal* 23.5 (2006), pp. 377–405.

[24] Marcus T. Pearce and Geraint A. Wiggins. “Improved Methods for Statistical Modelling of Monophonic Music”. In: *Journal of New Music Research* 33.4 (2004), pp. 367–385.

[25] Marcus Pearce, Darrell Conklin, and Geraint Wiggins. “Methods for Combining Statistical Models of Music”. In: *International Symposium on Computer Music Modeling and Retrieval*. Springer, 2004, pp. 295–312.

[26] Dan Ponsford, Geraint Wiggins, and Chris Mellish. “Statistical learning of harmonic movement”. In: *Journal of New Music Research* 28.2 (1999), pp. 150–177.

[27] Stanislaw Raczynski et al. “Multiple pitch transcription using DBN-based musicological models”. In: *International Society for Music Information Retrieval (ISMIR)*. 2010, pp. 363–368.

[28] M. Rohrmeier and I. Cross. “Statistical Properties of Harmony in Bach’s Chorales”. In: *10th International Conference on Music Perception and Cognition (ICMPC)*. 2008, pp. 619–627.

[29] Martin A Rohrmeier and Stefan Koelsch. "Predictive information processing in music cognition. A critical review". In: *International Journal of Psychophysiology* 83.2 (2012), pp. 164–175.

[30] Martin Rohrmeier and Thore Graepel. "Comparing feature-based models of harmony". In: *Proceedings of the 9th International Symposium on Computer Music Modelling and Retrieval*. Citeseer. 2012, pp. 357–370.

[31] Martin Rohrmeier and Patrick Rebuschat. "Implicit learning and acquisition of music". In: *Topics in cognitive science* 4.4 (2012), pp. 525–553.

[32] Jenny R Saffran et al. "Statistical learning of tone sequences by human infants and adults". In: *Cognition* 70.1 (1999), pp. 27–52.

[33] R. Scholz, E. Vincent, and F. Bimbot. "Robust modelling of musical chord sequences using probabilistic n-grams". In: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2009, pp. 53–56.

[34] David Temperley. "What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered". In: *Music Perception: An Interdisciplinary Journal* 17.1 (1999), pp. 65–100.

[35] Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. "Stochastic Gradient Descent Training for L1-Regularized Log-Linear Models with Cumulative Penalty". In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 477–485.

[36] Raymond P Whorley and Darrell Conklin. "Music generation from statistical models of harmony". In: *Journal of New Music Research* 45.2 (2016), pp. 160–183.

[37] R. Whorley et al. "Multiple Viewpoint Systems: Time Complexity and the Construction of Domains for Complex Musical Viewpoints in the Harmonisation Problem". In: *Journal of New Music Research* 42 (2013), pp. 237–266.

GENERATING NONTRIVIAL MELODIES FOR MUSIC AS A SERVICE

Yifei Teng
U. of Illinois, Dept. of ECE
teng9@illinois.edu

Anny Zhao
U. of Illinois, Dept. of ECE
anzhao2@illinois.edu

Camille Goudeseune
U. of Illinois, Beckman Inst.
cog@illinois.edu

ABSTRACT

We present a hybrid neural network and rule-based system that generates pop music. Music produced by pure rule-based systems often sounds mechanical. Music produced by machine learning sounds better, but still lacks hierarchical temporal structure. We restore temporal hierarchy by augmenting machine learning with a temporal production grammar, which generates the music's overall structure and chord progressions. A compatible melody is then generated by a conditional variational recurrent autoencoder.

The autoencoder is trained with eight-measure segments from a corpus of 10,000 MIDI files, each of which has had its melody track and chord progressions identified heuristically.

The autoencoder maps melody into a multi-dimensional feature space, conditioned by the underlying chord progression. A melody is then generated by feeding a random sample from that space to the autoencoder's decoder, along with the chord progression generated by the grammar. The autoencoder can make musically plausible variations on an existing melody, suitable for recurring motifs. It can also reharmonize a melody to a new chord progression, keeping the rhythm and contour.

The generated music compares favorably with that generated by other academic and commercial software designed for the music-as-a-service industry.

1. INTRODUCTION

Computer-generated music has started to expand from its pure artistic and academic roots into commerce. Companies such as Jukedeck and Amper offer so-called music as a service, by analogy with software as a service. However, their melodies, when present at all, often just arpeggiate the underlying chord.

We extend this approach by generating music with both chord progressions and interesting, nontrivial melodies. We expand a song structure such as $AA'BA$ into a harmonic plan, and then add a melody compatible with this structure and harmony. This compatibility uses a chord-melody relationship found by applying machine learning to a corpus of MIDI transcriptions of pop music (Figure 1).

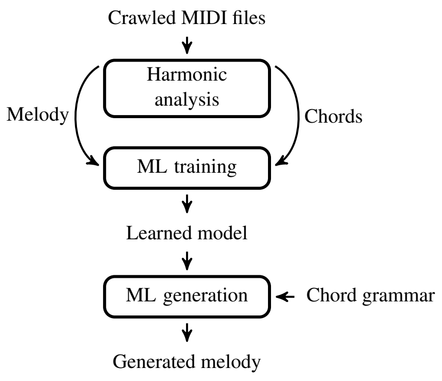


Figure 1: Machine learning (ML) workflow for generating music from a MIDI corpus.

Prior research is discussed in section 2. Harmonic analysis is detailed in sections 3 and 4. Hierarchy generation and melody generation are described in section 5.

2. RELATED WORK

Recent approaches to machine composition use neural networks (NNs), hoping to approximate how humans compose. Chu et al [5] generate a melody with a hierarchical NN that encodes a composition strategy for pop music, and then accompany the melody with chords and percussion. However, this music lacks hierarchical temporal structure. Boulanger-Lewandowski et al [3] investigate hierarchical temporal dependencies and long-term polyphonic structure. Inspired by how an opening theme often recurs at a song's end, they detect patterns with a recurrent temporal restricted Boltzmann machine (RTRBM). This can represent more complicated temporal distributions of notes. Similarly, Huang and Wu [10] generate structured music with a 2-layer Long Short Term Memory (LSTM) network. Although the resulting music often sounds plausible, it cannot produce clearly repeated melodic themes, just like a Markov resynthesis of the text of the famous poem "Jabberwocky" is unlikely to replicate the identical opening and closing stanzas of the original. Despite the LSTM network's theoretical capability of long-term memory, it fails to generalize to arbitrary time lengths [8], and its generated melodies remain unimaginative.

In these approaches, tonic chords dominate, and melody is little more than arpeggiation. To avoid this banality, we work in reverse. We first create structure and chords, and then fit melody to that. This mimics how classical western Roman-numeral harmony is taught to beginners: only after one has the underlying chord sequence, can one explain the melody in terms of chord tones, passing tones, appoggiaturas, and so on.

3. MELODY IDENTIFICATION

For pop music, a catchy and memorable melody is crucial. To generate melodies that sound less robotic than those generated by other algorithms, we use machine learning. To create a learning database, we started with a corpus of 10,000 MIDI files [16], from which we extracted useful training data (melodies that sound vivid or fun). In particular, the training data was eight-measure excerpts labelled as melody and chords. We thus had to identify which of a MIDI file's several tracks contained the melody. To do so, we assigned each track the sum of a rubric score and an entropy score. Whichever track scored highest was declared to be the melody. (Ties between high-scoring tracks were broken arbitrarily, because they were usually due to several tracks having identical notes, differing only in which instrument played them.)

3.1 Rubric Score

Our rubric considered attributes such as instrumentation, note density, and pitch range.

We first considered a track's instrument name (MIDI meta-event `FF 04`). Certain instruments are more common for melody, such as violin or flute. Others are more likely to be applied as accompaniment or long sustained notes, such as low brass. A third category is likely used as unpitched percussion. The instrument's category then adjusted the rubric's score.

We also considered the track's note density, how often at least one note is sounding (between corresponding MIDI note-on and note-off events), as a fraction of the track's full duration. A track scored higher if this was between 0.4 and 0.8, a typical value for pop melodies.

Finally we considered pitch range, because we observed that pop melodies often lie between C3 and C5. The score was higher for a pitch range between C3 and C6, to exclude bass tracks from consideration.

The values for these attributes were chosen based on manual inspection of 100 files in the corpus.

3.2 Entropy Score

We empirically observed that melody tracks often have a greater variety of pitches than other tracks. Thus, to quantify how varied, complex, and dynamic a track was, we calculated each track's entropy

$$H(X) = -\sum_{i=1}^{12} P(x_i) \log P(x_i) \tag{1}$$

where x_i represents the event that a particular note in the octave is i semitones from the pitch C, and $P(x_i)$ represents that event's probability. Higher entropy corresponds to a greater number of distinct pitches.

3.3 Evaluation

To measure how well this scoring identified melody tracks, we manually tagged the melody track of 160 randomly selected MIDI files. Comparing the scored prediction to this ground truth showed that the error rate was 15%.

4. CHORD DETECTION

To identify the chords in a MIDI file, we considered three aspects of how pop music differs from genres like classical music. First, chord inversions (where the lowest note is not the chord's root) are rare. When a new chord is presented, it is often in root position: most pop songs have a clear melody line and bass line [14], and the onset of a new chord is marked with the chord's root in that bass line. Second, chords may contain extensions (seventh), substitutions (flattened fifth), doublings, drop voicings (changing which octave a pitch sounds in), and omissions (third or fifth). Although such modifications complicate the task of functional harmony analysis, this is not a concern for our application. Third, new chord onsets are often at the start of a measure; rarely are there more than two chords per measure. Combining these observations led us to the following chord detection algorithm.

We first partition the song into segments with constant time signatures. (these are explicitly stated as MIDI meta messages). Then each segment is evenly divided into bins, where we try to match the entire bin to a chord. Because chords have different durations, we try different bin lengths: half a measure, one measure, and two measures. Then for each bin, containing all the notes sounding during that time interval, we add all these notes to a set that is matched against a fixed collection of chords, based on how close the pitches are, with a cost function:

Chord Detection: COST

```
1: function BESTCHORDINBIN($Pitches$)
2:     $Root$ ← Lowest note starting before first upbeat
3:     $Chords$ ← All chords, as array of intervals
4: return $\text{argmin}_{C \in Chords}\{\text{COST}(Pitches, C, Root)\}$
5: function COST($Pitches, Chord, Root$)
6:     $PitchCost \leftarrow 0$
7:     for $P \in Pitches$ do
8:         $interval$ ← No. semitones of $P$ from $Root$
9:         $d \leftarrow \min_{voice \in Chord}\{dist(interval, voice)\}$
10:        $PitchCost \leftarrow PitchCost + d$
11:    $ChordCost \leftarrow 0$
12:    for $voice \in Chord$ do
13:        $d \leftarrow \min_{P \in Pitches}\{dist(P - Root, voice)\}$
14:        $ChordCost \leftarrow ChordCost + d$
15: return $PitchCost + ChordCost$
```

Distance in semitones	Compatibility distance
0	0
1	6
2	2
3	2
4	2
5	1
6	4
7	1

Table 1: Interval compatibility.

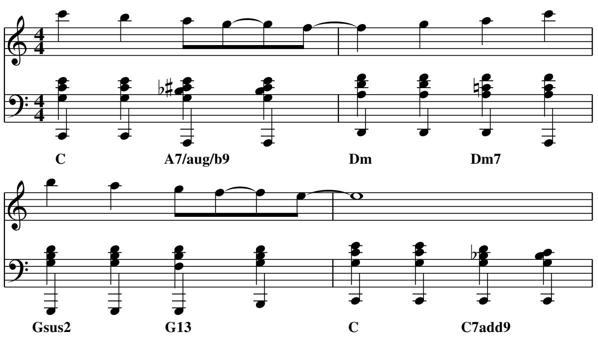


Figure 2: Example of chord detection.

Each chord's cost is the sum of the distance of the nearest interval in the chord (from the root) to each interval in the input pitches, and the distance of the nearest interval in the input pitches (from its "root") to each interval in the chord, based on some definition of distance. The cost function then returns the lowest-cost chord.

Defining the distance in terms of mere pitch difference in semitones would be simple, but performs poorly. For example, matching the pitch set $[C, E, G]$ to the chord $[C, E\flat, G\flat]$ would yield a cost of two, which is far too low. Instead, our distance function reflects how compatible intervals are. The unison is the most compatible, with distance zero; fourths and fifths are next, with distance one (Table 1). This conveniently handles omitted-fifth chords, because the chord's root matches the omitted fifth with a distance of only one.

Figure 2 demonstrates chord detection on the song *Fly me to the moon*. The bin size is half a measure, yielding 8 identified chords. The $A7/aug/\flat 9$ chord resulted from the accompaniment notes $A, G, B\flat$(flat ninth)$, C\sharp, E$, and the melody notes A, G, F(augmented fifth).

5. WRITING MUSIC

To output pieces with audibly hierarchical structure, we start with the harmonic structure produced by a temporal generative grammar. Then an autoencoder recurrent neural network (RNN) generates a melody compatible with this harmonic scaffold. The RNN learns to play using the chord's notes, with occasional surprising non-chord tone decorations such as passing tones and appoggiaturas.

5.1 Generating Melody

We first search for a representation of the melody using ML. This is traditionally done by an autoencoder, a pair of NNs that maps high-dimensional input data to and from a lower-dimensional space. Although this dimensionality reduction can eliminate perfect mappings, this turns out not to be a problem because the subspace of "pleasant" music within all possible musics is sufficiently small. Thus, the autoencoder can extract the pleasant content and map only that into the representation space.

It is tempting to feed a random point from the representation space to the autoencoder's decoder, and observe how much sense it makes of that point. However, because one cannot control the shape of the distribution of melody representations, one cannot guarantee that a given point from the representation space would be similar to those seen by the decoder during training. Thus, the vanilla autoencoder architecture [2] is not viable as a generative model. We propose the following improvements for generating melodies:

1. *Condition the NN on the chord progression.* The chord progression is provided to the NN at every level, so when reproducing a melody, the decoder has access to both the representation and the chord progression. This is useful because a melody has rhythmic information, intervallic content, and contour. The decoder can ignore the separately provided harmonic information, and use only the melody's other aspects. This also lets the representation remain constant while altering the chord progression, so the NN can adapt a melody to a changed chord progression, such as what happens when a key changes from minor to major.

2. *Add a stochastic layer.* Autoencoders which learn a stochastic representation are called variational autoencoders, and perform well in generative modelling of images [11]. The representation is not deterministic. We assume a particular (Gaussian) distribution in the representation space, and then train the NN to transform this distribution to match the distribution of input melodies in their high dimensional space. This ensures that we can take a random sampling of the representation space following its associated probability distribution, then feed it through the decoder and expect a melody similar to the set of musically sensible melodies.

3. *Use recurrent connections.* Pop music has many time-invariant elements, especially at time scales below a few beats. A recurrent NN shares the same processing infrastructure for note sequences starting at different times, and thereby accelerates learning.

4. *Normalize all other notes relative to the tonic.* Pop music is also largely pitch invariant, insofar as a song transposed by a few semitones still sounds perceptually similar. The NN ignores the song's key and considers the tonic pitch to be abstract, as far as pitches in melody and chords are concerned.

	-16	…	16	Silent	Attack	x8
	⋮	⋮	⋮	⋮	⋮	
16x	I	…	VI	VII	Silent	
	Pwr	Maj	Min	Dim	Aug	
	⋮	⋮	⋮	⋮	⋮	
	Major	Dorian	…	Locrian	Jazz Minor	

Table 2: An encoding of 8 measures (see section 5.1.1).

5.1.1 *Implementation*

The input melody is quantized to sixteenth notes. Only sections with an unchanging duple or quadruple meter are kept. The melody is converted to a series of one-hot vectors, whose slots represent offsets from the tonic in the range of -16 to 16 semitones, with one more slot representing silence. There is also an attack channel, where a value of 1 indicates that the note is being rearticulated at the current time step. The encoding for chords supports up to two chords per measure, and uses a one-hot vector for scale degrees and separate boolean channels for chord qualities (Table 2). (Note that because this encoding uses just seven Roman-numeral symbols, it does not try to represent chords outside the current mode. Before training, we removed from the corpus the few songs that contained significant occurrences of this.) We use the basic triad form for each chord identified using techniques from section 4, marking compatible chord qualities. For example, G^7 is encoded by marking a 1 in the Maj and Pwr columns. (The chord quality encoding could be extended to seventh and ninth chords.) The table's gray rows are data the network is conditioned on, while the other rows are input data that the network tries to reproduce. For an 8-measure example, the input and output vector size is $35 \times 8 \times 16 = 4480$, and the conditional vector size is $8 \times 16 + 5 \times 16 + 8 = 216$.

The network has 24 recurrent layers, 12 each for the encoder and decoder (Figure 3). Drawing on ideas of deep residual learning from computer vision [9], we make additional connections from the input to every third hidden layer. To improve learning, the network accesses both the original melody and the transformed results from previous layers during processing. The conditional part (chords and mode) is also provided to the network at every recurrent layer, as extra incoming connections.

The network is implemented in Tensorflow, a machine learning library for rapid prototyping and production training [1]. It was trained for four days on an Nvidia Tesla K80 GPU. We used Gated Recurrent Units [4] to build the bidirectional recurrent layer and Exponential Linear Units [7] as activation functions. These significantly accelerate training while simplifying the network [6, 7]. Figure 4 shows the training error (the sum of model reproduction errors) and the difference of the latent distribution from a unit Gaussian distribution, as measured by Kullback-Leibler divergence [12]. The network's input data (available at `https://goo.gl/VezNNA`) is a set of MIDI

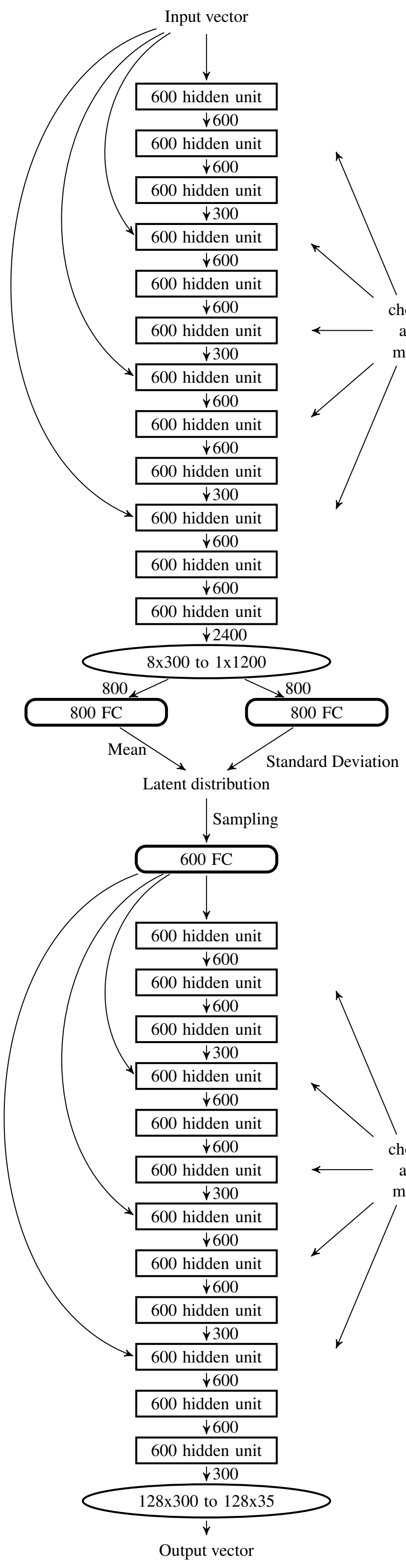


Figure 3: Network architecture. Rectangles are bidirectional recurrent neural network cells. Ellipses are strided time-convolution cells. Rounded rectangles are fully connected (FC) layers. Numbered arrows indicate a connection's dimension.

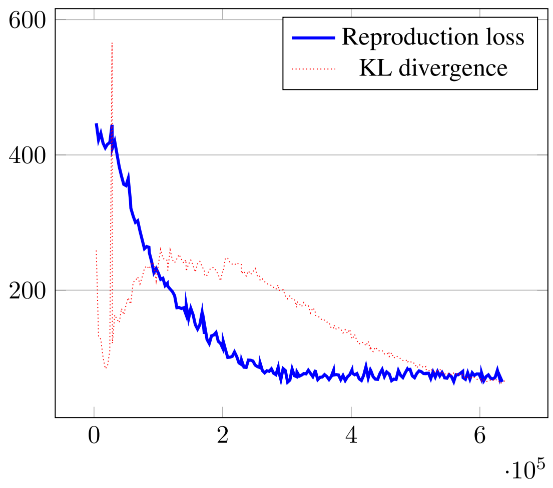


Figure 4: Training error and Kullback-Leibler divergence of the NN. The horizontal axis indicates how many training segments have elapsed ($\times 10^5$). Initial outliers have been removed.

songs from various online sources. Our harmonic analysis converted this to 1.9×10^6 measures of melodies and corresponding chords. We implemented KL warm up, because that is crucial to learning for a variational autoencoder [15]. But instead of linearly scaling the KL term for this, we found that a sigmoid reduced the network's reproduction loss.

5.2 Generating Hierarchy and Chords

Hierarchy and chords are generated simultaneously, using a temporal generative grammar [13], modified to suit the harmonies of pop music, and extended to enable repeated motifs with variations. The original temporal generative grammar has notions of sharing by binding a section to a symbol. For example, the rule

$$\text{let } x = \text{I in I } M5(x) \text{ I } M5(x) \text{ I}, \qquad (2)$$

where $M5$ indicates modulating to the 5th degree, would expand to five sections, with the second and fourth identical because x is reused. We extend this by having symbols x carry along a number: $x_1, x_2, \ldots$. Different subscripts of the same symbol still expand to the same chord progression, but denote slightly different latent representations when generating corresponding melodies for those sections. The latent representations corresponding to $x_{i>1}$ are derived from that of x_1 by adding random Gaussian perturbations. This yields variations on the original melody.

5.3 Training Examples in the Representation Space

We randomly chose 130 songs from the training set, fed them through the network, and performed t-SNE analysis on the resulting 130 locations in the representation space. Although a melody maps to a distribution in the representation space, Figure 5 plots only each distribution's mean, for visual clarity. This t-SNE analysis effectively reduces the 800-dimensional representation

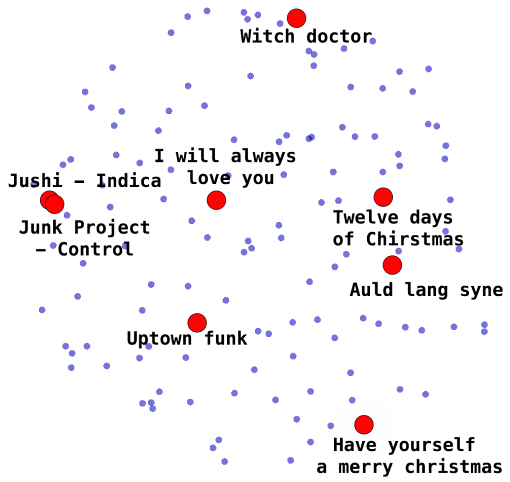


Figure 5: Example melodies in a t-SNE plot of the representation space.

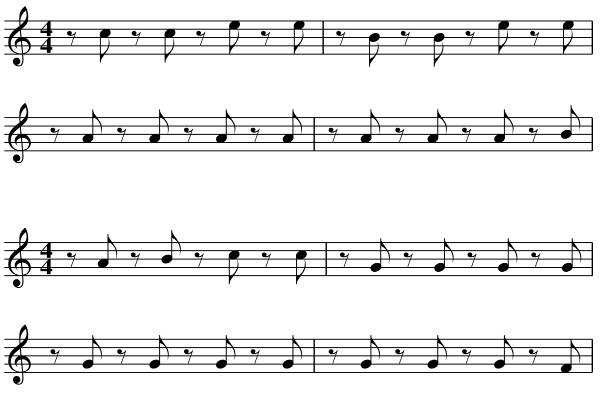


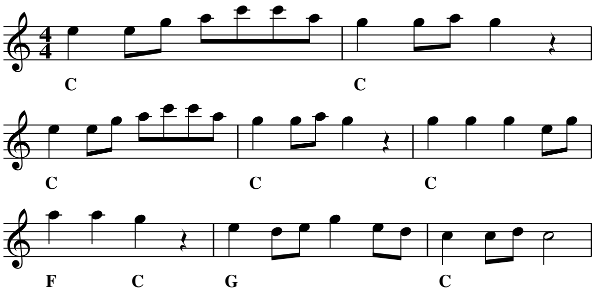
Figure 6: Four-bar excerpts from the songs *Indica* (top) and *Control* (bottom).

space into a low-dimensional human-readable format [17]. (A larger interactive visualization of 1,680 songs is at `https://composing.ai/tsne`.)

Two songs that are both in the techno genre, *Indica* by Jushi and *Control* by Junk Project, are indeed very near in the t-SNE plot, almost overlapping. Excerpts from them show that both have a staccato rhythm with notes landing on the upbeat, and have similar contours (Figure 6).

5.4 Reharmonizing Melody

We hypothesized that, when building the neural network architecture, providing the chord progression to both the encoder and the decoder would not preserve that information in the representation space, thus saving space for rhythmic nuances and contour. To test this hypothesis, we gave the network songs disjoint from the training set and collected their representations. We then fed these representations along with a new chord progression to the network. We hoped that it would respond by generat-



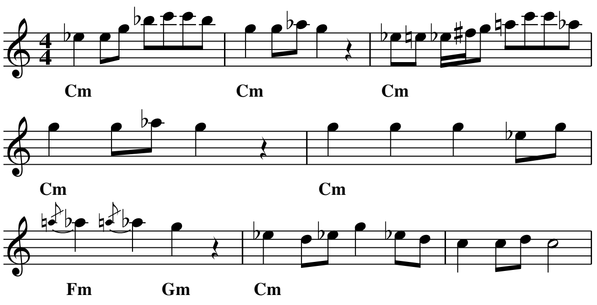


Figure 7: The song *Jasmine Flower* with original chords (top), and adapted to a new chord progression (bottom).

ing a melody that was harmonically compatible with the new chord progression, while still resembling the original melody. We demonstrate this with the Chinese folk song *Jasmine Flower*, in a genre unfamiliar to the NN (Figure 7). Note that we supplied the chords in Figure 7 (bottom), for which the NN filled in the melody. The network flattened the E, A, and B, by observing that the chord progression looked minor. This is typically how a human would perform the reharmonization, demonstrating the network's comprehension of how melody and harmony interact.

Although the NN struggled to reproduce the melody, it provided interesting modifications. The grace notes in measure 6 could be due to similar ones in the training set, or due to vacillation between the A♮ from the representation and the A♭ from the chord conditioning.

5.5 Examples of Generated Melodies

Because an entire multi-section composition cannot fit here, we merely show excerpts from two shorter examples.

Figure 8 and Figure 9 demonstrate melodies generated from points in the representation space that are not near any particular previously known melody. Structure is evident in Figure 8: measures 1–3 present a short phrase, and measure 4 leads to the next four measures, which recapitulate the first three measures with elaborate variation. Figure 9 shows an energetic melody where the grammar only produced C minor chords. Although the final two measures wander off, the first six have consistent style and natural contour.

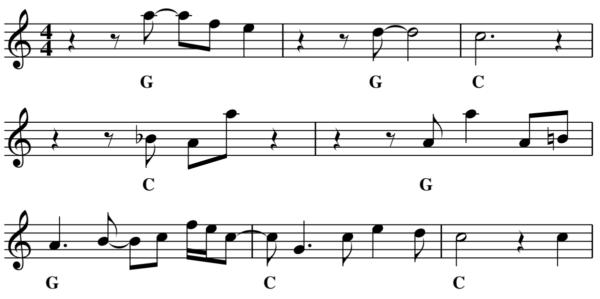


Figure 8: Generated melody for a grammar-generated chord progression.

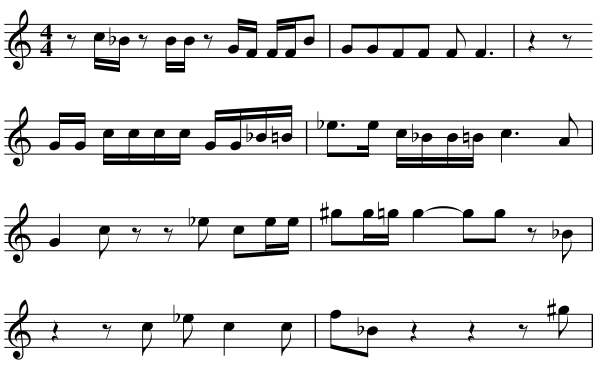


Figure 9: Generated melody for an extended C minor chord.

6. CONCLUSION AND FUTURE WORK

We have combined generative grammars for structure and harmony with a NN, trained on a large corpus, to emit melodies compatible with a given chord progression. This system generates compositions in a pop music style whose melody, harmony, motivic development, and hierarchical structure all fit the genre.

This system is currently limited by assuming that the input data's chords are in root position. More sophisticated chord detection would still let it exploit the relative harmonic rigidity of popular music. Also, by investigating the representation found by the NN, meaning could be assigned to some of its 800 dimensions, such as intensity, consonance, and contour. This would let us boost or attenuate a given melody along those dimensions.

Acknowledgements. The authors are grateful to Mark Hasegawa-Johnson for overall guidance, and to the anonymous reviewers for insights into both philosophical issues and minute details.

7. REFERENCES

[1] 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. 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 *Proc. USENIX*

Conf. Operating Systems Design and Implementation, pages 265–283. USENIX Association, 2016.

[2] Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[3] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. `arxiv.org/abs/1206.6392`, 2012.

[4] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. `arxiv.org/abs/1406.1078`, 2014.

[5] H. Chu, R. Urtasun, and S. Fidler. Song from PI: a musically plausible network for pop music generation. `arxiv.org/abs/1611.03477`, 2016.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. `arxiv.org/abs/1412.3555`, 2014.

[7] D. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). `arxiv.org/abs/1511.07289`, 2015.

[8] A. Graves, G. Wayne, and I. Danihelka. Neural turing machines. `arxiv.org/abs/1410.5401`, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. `arxiv.org/abs/1512.03385`, 2015.

[10] A. Huang and R. Wu. Deep learning for music. `arxiv.org/abs/1606.04930`, 2016.

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes. `arxiv.org/abs/1312.6114`, 2013.

[12] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[13] D. Quick and P. Hudak. A temporal generative graph grammar for harmonic and metrical structure. In *Proc. International Computer Music Conference*, pages 177–184, 2013.

[14] M. P. Ryynänen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.

[15] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. `arxiv.org/abs/1602.02282`, 2016.

[16] Y. Teng and A. Zhao. Composing.AI. `http://composing.ai/`, 2017.

[17] L. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

GEOGRAPHICAL ORIGIN PREDICTION OF FOLK MUSIC RECORDINGS FROM THE UNITED KINGDOM

Vytaute Kedyte[1] **Maria Panteli**[2] **Tillman Weyde**[1] **Simon Dixon**[2]
[1] Department of Computer Science, City University of London, United Kingdom
[2] Centre for Digital Music, Queen Mary University of London, United Kingdom
{Vytaute.Kedyte, T.E.Weyde}@city.ac.uk, {m.panteli, s.e.dixon}@qmul.ac.uk

ABSTRACT

Field recordings from ethnomusicological research since the beginning of the 20th century are available today in large digitised music archives. The application of music information retrieval and data mining technologies can aid large-scale data processing leading to a better understanding of the history of cultural exchange. In this paper we focus on folk and traditional music from the United Kingdom and study the correlation between spatial origins and musical characteristics. In particular, we investigate whether the geographical location of music recordings can be predicted solely from the content of the audio signal. We build a neural network that takes as input a feature vector capturing musical aspects of the audio signal and predicts the latitude and longitude of the origins of the music recording. We explore the performance of the model for different sets of features and compare the prediction accuracy between geographical regions of the UK. Our model predicts the geographical coordinates of music recordings with an average error of less than 120 km. The model can be used in a similar manner to identify the origins of recordings in large unlabelled music collections and reveal patterns of similarity in music from around the world.

1. INTRODUCTION

Since the beginning of the 20th century ethnomusicological research has contributed significantly to the collection of recorded music from around the world. Collections of field recordings are preserved today in digital archives such as the British Library Sound Archive. The advances of Music Information Retrieval (MIR) technologies make it possible to process large numbers of music recordings. We are interested in applying these computational tools to study a large collection of folk and traditional music from the United Kingdom (UK). We focus on exploring music attributes with respect to geographical regions of the UK and investigate patterns of music similarity.

The comparison of music from different geographical regions has been the topic of several studies from the field of ethnomusicology and in particular the branch of comparative musicology [13]. Savage et al. [17] studied stylistic similarity within music cultures of Taiwan. In particular, they formed music clusters for a collection of 259 traditional songs from twelve indigenous populations of Taiwan and studied the distribution of these clusters across geographical regions of Taiwan. They showed that songs of Taiwan can be grouped into 5 clusters correlated with geographical factors and repertoire diversity. Savage et al. [18] analysed 304 recordings contained in the 'Garland Encyclopedia of World Music' [14] and investigated the distribution of music attributes across music recordings from around the world. They proposed 18 music features that are shared amongst many music cultures of the world and a network of 10 features that often occur together.

The aforementioned studies incorporated knowledge from human experts in order to annotate music characteristics for each recording. While expert knowledge provides reliable and in-depth insights into the music, the amount of human labour involved in the process makes it impractical for large-scale music corpora. Computational tools on the other hand provide an efficient solution to processing large numbers of music recordings. In the field of MIR several studies have used computational tools to study large music corpora. For example, Mauch et al. [10] studied the evolution of popular music in the USA in a collection of approximately 17000 recordings. They concluded that popular music in the US evolved with particular rapidity during three stylistic revolutions, around 1964, 1983 and 1991. With respect to non-Western music repertoires Moelants et al. [12] studied pitch distributions in 901 recordings from Central Africa from the beginning until the end of the 20th century. They observed that recent recordings tend to use more equally-tempered scales than older recordings.

Computational studies have also focused on predicting the geographic location of recordings from their music content. Gomez et al. [3] approached prediction of musical cultures as a classification problem, and classified music tracks into Western and non-Western. They identified correlations between the latitude and tonal features, and the longitude and rhythmic descriptors. Their work illustrates the complexity of using regression to predict the geographical coordinates of music origin. Zhou et al. [23] also approached this as a regression problem, predicting latitudes and longitudes of the capital city of the music's country of origin, for pieces of music from 73 countries. They used K-nearest neighbours and Random Forest regression techniques, and achieved a mean distance error between predicted and target coordinates of 3113 kilometres (km). The advantage of treating geographic origin prediction as a regression problem is that it allows the latitude and longitude correlations found by Gomez et al. [3] to be considered as well as the topology of the Earth. The disadvantage is not accounting for latitudes getting distorted towards the poles, and longitudes diverging at ±180 degrees. Location is usually used as an input feature in regression models, however some studies have explored prediction of geographical origin in a continuous space in the domains of linguistics [2], criminology [22], and genetics [15, 21].

In this paper we study the correlation between spatial origins and musical characteristics of field recordings from the UK. We investigate whether the geographical location of a music recording can be predicted solely based on its audio content. We extract features capturing musical aspects of the audio signal and train a neural network to predict the latitude and longitude of the origins of the recording. We investigate the model's performance for different network architectures and learning parameters. We also compare the performance accuracy for several feature sets as well as the accuracy across different geographical regions of the UK.

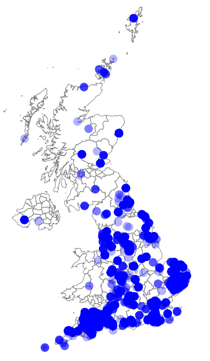
Our developments contribute to the evaluation of existing audio features and their applicability to folk music analysis. Our results provide insights for music patterns across the UK, but the model can be expanded to process music recordings from all around the world. This could contribute to identifying the location of recordings in large unlabelled music collections as well as studying patterns of music similarity in world music.

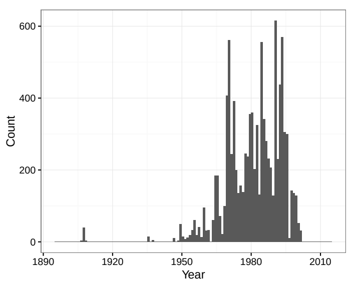
This paper is organised as follows: Section 2 provides an overview of the music collection and Section 3 describes the different sets of audio features considered in this study. Section 4 provides a detailed description of the neural network architecture as well as the training and testing procedures. Section 5 presents the results of the model for different learning parameters, audio features, and geographical areas. We conclude with a discussion and directions for future work.

2. DATASET

Our music dataset is drawn from the World & Traditional music collection of the British Library Sound Archive [1] which includes thousands of music recordings collected over decades of ethnomusicological research. In particular, we use a subset of the World & Traditional music collection curated for the Digital Music Lab project [1]. This subset consists of more than 29000 audio recordings with a large representation (17000) from the UK. We focus solely on recordings from the UK and process information on the recording's location (if available) to extract the latitude and longitude coordinates. We keep only those tracks whose extracted coordinates lie within the spatial boundaries of the UK.

The final dataset consists of a total of 10055 recordings. The recordings span the years between 1904 and 2002 with median year 1983 and standard deviation 12.3 years. See Figure 1 for an overview of the geographical and temporal distribution of the dataset. The origins of the recordings span a range of maximum 1222 km. From the origins of all 10055 recordings we compute the average latitude and average longitude coordinates and estimate the distance between each recording's location and the average latitude, longitude. This results in a mean distance of 167 with standard deviation of 85 km. A similar estimate is computed from recordings in the training set and used as the random baseline for our regression predictions (Section 5).


(a) Geographical spread



(b) Year distribution

Figure 1: Geographical spread and year distribution in our dataset of 10055 traditional music recordings from the UK.

3. AUDIO FEATURES

We aim to process music recordings to extract audio features that capture relevant music characteristics. We use a speech/music segmentation algorithm as a preprocessing step and extract features from the music segments using available VAMP plugins [2]. We post-process the output of the VAMP plugins to compute musical descriptors based on state of the art MIR research. Additional dimensionality reduction and scaling is considered as a final step. The methodology is summarised in Figure 2 and details are explained below.

Several recordings in our dataset consist of compilations of multiple songs or a mixture of speech and music segments. The first step in our methodology is to use a speech/music segmentation algorithm to extract relevant music segments from which the rest of the analysis is derived. We choose the best performing segmentation algorithm [9] based on the results of the Music/Speech Detection task of the MIREX 2015 evaluation [3]. We apply the segmentation algorithm to extract music segments from

[1] http://sounds.bl.uk/World-and-traditional-music

[2] http://www.vamp-plugins.org

[3] http://www.music-ir.org/mirex/wiki/2015:Music/Speech_Classification_and_Detection

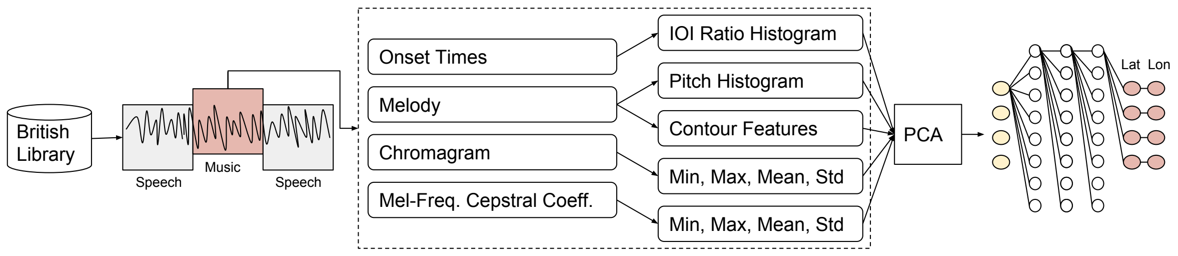


Figure 2: Summary of the methodology: UK folk music recordings are processed with a speech/music segmentation algorithm and VAMP plugins are applied to music segments. Audio features are derived from the output of the VAMP plugins, PCA is applied, and output is fed to a neural network that predicts the latitude and longitude of the recording.

each recording in our dataset. We require a minimum of 10 seconds of music for each recording and discard any recordings with total duration of music segments less than this threshold.

Our analysis aims to capture relevant musical characteristics which are informative for the spatial origins of the music. We focus on aspects of rhythm, melody, timbre, and harmony. We derive audio features from the following VAMP plugins: MELODIA - Melody Extraction [4], Queen Mary - Chromagram [5], Queen Mary - Mel-Frequency Cepstral Coefficients [6], and Queen Mary - Note Onset Detector [7]. We apply these plugins for each recording in our dataset and omit frames that correspond to non-music segments as annotated by the previous step of speech/music segmentation.

The raw output of the VAMP plugins cannot be directly incorporated in our regression model. We post-process the output to low-dimensional and musically meaningful descriptors as explained below.

Rhythm. We post-process the output of the Queen Mary - Note Onset Detector plugin to derive histograms of inter-onset interval (IOI) ratios [4]. Let $O = \{o_1, ..., o_n\}$ denote a sequence of n onset locations (in seconds) as output by the VAMP plugin. The IOIs are defined as $IOI = \{o_{i+1} - o_i\}$ for index $i = 1, ..., n-1$. The IOI ratios are defined as $IOIR = \{\frac{IOI_{j+1}}{IOI_j}\}$ for index $j = 1, ..., n-2$. The IOI ratios denote tempo-independent descriptors because the tempo information carried with the magnitude of IOIs vanishes with the ratio estimation. We compute a histogram for the $IOIR$ values with 100 bins uniformly distributed between $[0, 10)$.

Timbre. We extract summary statistics from the output of the Queen Mary - Mel-Frequency Cepstral Coefficients (MFCC) plugin [8] with the default values of frame and hop size. In particular, we remove the first coefficient (DC component) and extract the min, max, mean, and standard deviation of the remaining 19 MFCCs over time.

Melody. The output of the MELODIA - Melody Extraction plugin denotes the frequency estimates over time of the lead melody. We extract a set of features capturing characteristics of the pitch contour shape and melodic embellishments [16]. In particular, we extract statistics of the pitch range and duration, fit a polynomial curve to model the overall shape and turning points of the contour, and estimate the vibrato range and extent of melodic embellishments. Each recording may consist of multiple shorter pitch contours. We keep the mean and standard deviation of features across all pitch contours extracted from the audio recording. We also post-process the output from MELODIA to compute an octave-wrapped pitch histogram [20] with 1200-cent resolution.

Harmony. The output of the Queen Mary - Chromagram plugin is an octave-wrapped chromagram with 100-cent resolution [5]. We use the default frame and hop size and extract summary statistics denoting the min, max, mean, and standard deviation of chroma vectors over time.

The above process results in a total of 1484 features per recording. Before further processing, the features were standardised with z-scores. Dimensionality reduction was also applied with Principal Component Analysis (PCA) including whitening and keeping enough components to represent 99% of the variance.

4. REGRESSION MODEL

The prediction of spatial coordinates from music data has been treated as a regression problem in previous research using K-nearest neighbours and Random Forest Regression methods [23]. We explore the application of a neural network method. Neural networks have been shown to outperform existing methods in supervised tasks of music similarity [7, 11, 19]. We evaluate the performance of a neural network under different parameters for the regression problem of predicting latitude and longitudes from music features.

A neural network with two continuous value outputs, latitude and longitude predictions, was built in Tensorflow. We used the Adaptive Moment Estimation (Adam) algorithm for optimisation, Rectified Linear Unit (ReLU) as activation function, and drop-out rate of 0.5 for regularisation. The evaluation of the model performance was based on the mean distance error in km, calculated using the Haversine formula [6]. The Haversine distance d between two points in km is given by

[4] `http://mtg.upf.edu/technologies/melodia`
[5] `http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-chromagram`
[6] `http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-mfcc`
[7] `http://vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-onsetdetector`

Parameters	Values
Target Scaling	True or False
Number of hidden layers	$\{3, 4\}$
Cost function	Haversine or MSE
Learning Rate	$\{0.005, 0.01, 0.05\}$
L1 regularisation	$\{0, 0.05, 0.5\}$
L2 regularisation	$\{0, 0.05, 0.5\}$

Table 1: The hyper-parameters and their range of values for optimisation.

$$d = 2r\arcsin([\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos(\phi_1)\cos(\phi_2)\sin^2(\frac{\lambda_2 - \lambda_1}{2})]^{\frac{1}{2}}) \quad (1)$$

where ϕ represents the latitude, λ longitude, and r the radius of the sphere (with r fixed to 6367 km in this study). We further explored the performance of the model under architectures with different numbers of hidden layers, two different cost functions, and a range of regularisation parameters as explained below.

4.1 Parameter Optimisation

A grid-search of model hyper-parameters was performed to identify the combination that achieves best performance in cross-validation. The following hyper-parameters were considered for optimisation: whether or not to scale the targets (i.e., z-score standardisation of the ground truth latitude/longitude coordinates of each recording), the number of hidden layers, two possible cost functions, namely, the Haversine distance in km and the Mean Squared Error (MSE), and a range of values for learning rate, $L1$ and $L2$ regularisation parameters. The parameter optimisation is summarised in Table 1. We tested in total 216 combinations of hyper-parameters and selected the best performing combination to tune parameters and retrain the model for the final results.

4.2 Train-test splits

The training of the model was done in two phases. First the model was trained using the full set of features (Section 3) and the different hyper-parameters as defined in Table 1. The hyper-parameters were tuned based on the optimal performance obtained through cross-validation. In the second phase, the hyper-parameters were fixed to their optimal values and the model was retrained for different sets of features. Each new model's performance was assessed on a test set unique to that model.

In the first training phase, we sampled at random 70% from the total number of 10055 recordings for training. This resulted in a total of 7038 samples in the training set, of which 30% (2111) was set aside for validation. Following PCA, the feature dimensionality of the dataset was 368.

Target Scaling	Hidden Layers	Cost Function	Training Error (km)	Validation Error (km)
True	3	Haversine	**72.68**	**119.36**
True	3	MSE	166.21	166.27
True	4	Haversine	98.03	128.44
True	4	MSE	166.19	166.24
False	3	Haversine	165.34	166.79
False	3	MSE	169.91	169.30
False	4	Haversine	170.91	171.26
False	4	MSE	181.44	180.10

Table 2: Results for parameter optimisation. Learning rate, $L1$, and $L2$ regularisation parameters are fixed to $0.005, 0, 0.5$ respectively. Best performance is obtained when target scaling is combined with 3 hidden layers and Haversine distance as cost function.

We used cross-validation with $K = 5$ folds and tuned parameters based on the mean of the distance error on the validation set (Equation 1). In the second phase we retrained the model for different feature sets. For each feature set, the dataset was split into training (random 70%) and test (remaining 30%) and the performance of the model was assessed on the test set.

5. RESULTS

5.1 Parameter Optimisation

The model that produced the lowest mean error on the validation set (119 km) used the following hyper parameters: target scaling, 3 hidden layers, Haversine distance as cost function, learning rate of 0.005, and $L1$, $L2$ regularisation parameters of 0 and 0.5, respectively. The main hyper-parameters that determined the accuracy of the model were the use of Haversine distance as the cost function, and the application of target scaling. The performance of the model for different parameter values is shown in Table 2.

5.2 Results for different feature sets

The second set of experiments explored the performance of the model when trained for different sets of features. We estimated the random baseline from the origins of recordings in the training set. In particular, we computed the average latitude and average longitude coordinates of recordings and estimated the distance between each recording's location and the average latitude, longitude. Based on this estimate the mean distance error of the baseline approach was 167.4 km. Each model was compared to the baseline approach (i.e., the mean distance error of its test targets) with a Wilcoxon signed-rank test. The performances of the models trained on different sets of features and evaluated on separate test sets were compared with a pairwise Wilcoxon rank sum test (also known as Mann-Whitney) with Bonferroni correction for multiple comparisons. We consider a significance level of $\alpha = 0.05$ and denote the Bonferroni corrected level by $\hat{\alpha}$.

Model No.	Feature Set Name	Error (km)
1	All features	149.8
2	Rhythm: IOIR histogram	160.0
3	Harmony: Chromagram statistics	152.5
4	Timbre: MFCC statistics	129.0
5	Pitch histogram	160.1
6	Contour features mean	159.8
7	Contour features standard deviation	162.3
8	Melody: Pitch hist., contour features	152.6
9	Rhythm and Harmony	149.1
10	Rhythm and Timbre	120.1
11	Rhythm and Melody	150.5
12	Melody and Harmony	139.4
13	Melody and Timbre	117.1
14	Timbre and Harmony	**114.0**
15	Rhythm, Harmony, and Timbre	118.3
16	Rhythm, Harmony, and Melody	142.8
17	Rhythm, Timbre, and Melody	119.8
18	Harmony, Timbre, and Melody	140.3
–	Baseline	167.4

Table 3: The mean distance error (in km) of the test set for 18 models trained on different sets of features.

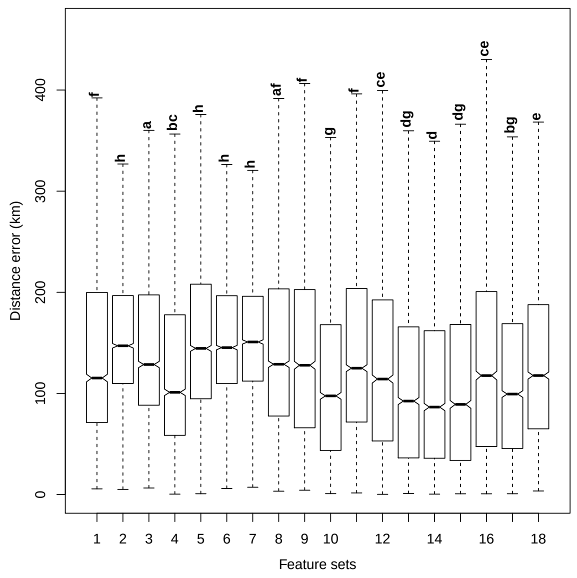
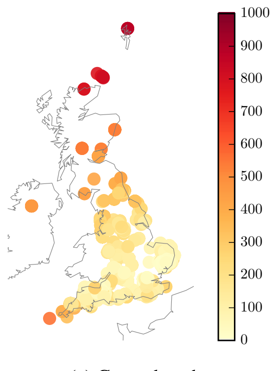
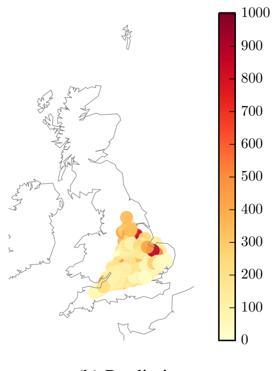


Figure 3: Distance error of predictions for different sets of features (see Table 3 for the feature set used to train each model). Labels $a - l$ in indicate features sets that have non-significantly different results ($p > \hat{\alpha}$) where they share the same letter. For example, feature set 3 shares the label a with feature set 8 but shares no label with any other feature set, indicating that results from model 3 are significantly different from all other models except for model 8.



(a) Ground truth



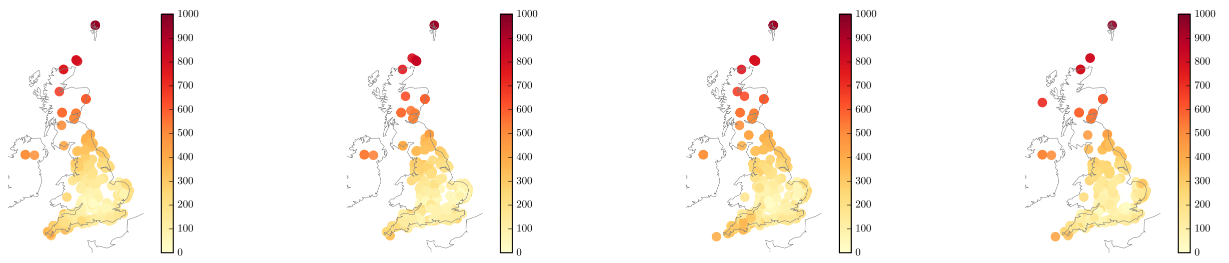
(b) Predictions

Figure 4: (a) Ground truth and (b) predicted music recording origins, coloured by the distance error (in km) for the best performing model (no. 14).

All models achieved results significantly different from the baseline approach ($p < .0001$). The best performance (lowest error of 114.0 km) was achieved when combining the timbral and harmonic descriptors (model 14). This combines the summary statistics of the chromagram and the summary statistics of the MFCCs. The performance of this model was significantly different ($p < \hat{\alpha}$) from all other models except models 13 and 15 trained on melodic and timbral, and rhythmic, harmonic and timbral descriptors, respectively. The model achieved a mean error of 149.8 km on the test set when all features (Section 3) were used. The results from model 3 trained on harmonic descriptors were significantly different from all other models except model 8 trained on melodic features. The model trained on rhythmic descriptors (model 2) is amongst the weakest predictors. However, adding rhythmic features to any of melodic, harmonic, or timbral features, for example models $9, 10, 11$, significantly improves the performance of the model ($p < \hat{\alpha}$ for pairwise comparisons between models 3 and 9, 4 and 10, 8 and 11). Models $5, 6, 7$ trained on pitch histograms, contour features mean, and contour features standard deviation, respectively, are also amongst the weakest predictors but when all these features are combined together as in model 8, the performance is improved. See Table 3 for an overview of the prediction accuracy of models trained on different feature sets. Figure 3 provides a box-plot visualisation of the results from different feature sets and marks statistical significance between results.

5.3 Results for different regions

The last analyses aim to study the prediction accuracy with respect to the geographical origins of recordings. Figure 4 shows the ground truth and predicted coordinates for the best performing model (model no.14 as denoted in Table 3) coloured by the distance error in km. We observe that data points with the lowest predictive accuracy originate from the north-eastern and the south-western areas of the UK (Figure 4a). Predictions are mostly concentrated in the southern part of the UK. Data points predicted towards the



(a) Rhythm (b) Harmony (c) Timbre (d) Melody

Figure 5: Music recording origins coloured by the distance error (in km) for models trained on (a) rhythmic, (b) harmonic, (c) timbral, and (d) melodic features (models no. 2, 3, 4, 8 respectively as defined in Table 3).

eastern areas indicate a larger distance error (Figure 4b).

In Figure 5 we visualise the prediction accuracy of models trained on different feature sets with respect to geography. We observe that for all models the northern areas of the UK (i.e., in the region of Scotland) are predicted with a relatively large distance error (lowest accuracy). For the model trained on timbral features (Figure 5c) we also observe the south west of England predicted with lower accuracy than the models trained on harmonic and melodic features (Figures 5b and 5d).

6. DISCUSSION

Our results provide insights on the contribution of different feature sets and suggest patterns of music similarity across geographical regions. The methodology can be improved in various ways.

The initial corpus of folk and traditional music from the UK consisted of a total of 17000 of which only 10055 were processed in this study. The final dataset had a skewed geographical distribution with over-representation of the south-eastern and south-western UK regions, e.g., Devon and Suffolk, and under-representation of the North-Eastern, North-Western areas, e.g., Scotland and Northern Ireland. Effects from the skewness of the dataset could be observed in the distribution of predicted latitude and longitude coordinates (Figure 4b). A larger and more representative corpus can be used in future work.

We used features derived from the output of VAMP plugins to describe musical content of audio recordings. Some of these plugins were designed for different music styles and their application to folk music might not give robust results. A thorough evaluation of the suitability of the features can give valuable insights for improving their robustness to different corpora such as the one used in this study. We used feature representations averaged over time but in future work preserving temporal information in the features could provide better music content description.

We observed that results from models trained on individual features showed on average larger distance errors. When however combinations of features were considered, the model achieved on average higher accuracies. An exception is the case when all features were considered but the performance of the model had a relatively large distance error. This could be due to limitations of the model especially with regards to over-fitting or the lack of adequate music information captured by the features. Integrating additional audio features could help capture more of the variance of the data and improve the model.

The model was validated for a range of parameters and several approaches were considered to avoid over-fitting. However, evidence of over-fitting could still be observed in the final results. Training with more data could help make the model more generalisable in future work. What is more, oversampling techniques could be explored to overcome the problem of under-represented geographical regions in our dataset.

Neural networks in combination with audio features as proposed in this study, can provide good predictions of the origins of the music. This can aid musicological research as well as improve spatial metadata associated with large music collections.

7. CONCLUSION

We studied a collection of field recordings from the UK and investigated whether the geographical origins of recordings can be predicted from the music attributes of the audio signal. We treated this as a regression problem and trained a neural network to take as input audio features and predict the latitude and longitude of the music's origin. We trained the model under different hyperparameters and tested its performance for different feature sets. Highest accuracy was achieved for the model trained on timbral and harmonic features but no significant differences were found to the same model with rhythm features added or with melody replacing harmony. The southern regions of the UK were predicted with a relatively high accuracy whereas northern regions were predicted with low accuracy. Effects of the skewness of the dataset and the reliability of audio features were discussed. The corpus and methodology can be improved in future work and the applicability of the model could be extended to music from around the world.

8. ACKNOWLEDGEMENTS

MP is supported by a Queen Mary research studentship.

9. REFERENCES

[1] S. Abdallah, E. Benetos, N. Gold, S. Hargreaves, T. Weyde, and D. Wolff. The Digital Music Lab: A Big Data Infrastructure for Digital Musicology. *ACM Journal on Computing and Cultural Heritage*, 10(1), 2017.

[2] J. Eisenstein, B. O'Connor, N.A Smith, and E.P. Xing. A Latent Variable Model for Geographic Lexical Variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1277–1287, 2010.

[3] E. Gómez, M. Haro, and P. Herrera. Music and geography: Content description of musical audio from different parts of the world. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 753–758, 2009.

[4] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Proceedings of the AES 25th International Conference*, pages 196–204, 2004.

[5] C. Harte and M. Sandler. Automatic chord identifcation using a quantised chromagram. In *118th Audio Engineering Society Convention*, 2005.

[6] J. Inman. *Navigation and Nautical Astronomy: For the Use of British Seamen*. F. & J. Rivington, 1849.

[7] I. Karydis, K. Kermanidis, S. Sioutas, and L. Iliadis. Comparing content and context based similarity for musical data. *Neurocomputing*, 107:69–76, 2013.

[8] B. Logan. Mel-Frequency Cepstral Coefficients for Music Modeling. In *Proceedings of the International Symposium on Music Information Retrieval*, 2000.

[9] M. Marolt. Music/speech classification and detection submission for MIREX 2015. In *MIREX*, 2015.

[10] M. Mauch, R. M. MacCallum, M. Levy, and A. M. Leroi. The evolution of popular music: USA 1960-2010. *Royal Society Open Science*, 2(5):150081, 2015.

[11] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 525–530, 2004.

[12] D. Moelants, O. Cornelis, and M. Leman. Exploring African Tone Scales. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 489–494, 2009.

[13] B. Nettl. *The Study of Ethnomusicology: Thirty-one Issues and Concepts*. University of Illinois Press, Urbana and Chicago, 2nd edition, 2005.

[14] B. Nettl, R. M. Stone, J. Porter, and T. Rice, editors. *The Garland Encyclopedia of World Music.* Garland Pub, New York, 1998-2002 edition, 1998.

[15] J. Novembre, K. Bryc, S. Bergmann, A.R. Boyko, C.D. Bustamante, A. Auton, M. Stephens, Z. Kutalik, A. Indap, T. Johnson, M.R. Nelson, and K.S. King. Genes mirror geography within Europe. *Nature*, 456(7218):98–101, 2008.

[16] M. Panteli, R. Bittner, J. P. Bello, and S. Dixon. Towards the characterization of singing styles in world music. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 636–640, 2017.

[17] P. E. Savage and S. Brown. Mapping Music: Cluster Analysis Of Song-Type Frequencies Within and Between Cultures. *Ethnomusicology*, 58(1):133–155, 2014.

[18] 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 of the United States of America*, 112(29):8987–8992, 2015.

[19] D. Turnbull and C. Elkan. Fast recognition of musical genres using RBF networks. *IEEE Transactions on Knowledge and Data Engineering*, 17(4):580–584, 2005.

[20] G. Tzanetakis, A. Ermolinskyi, and P. Cook. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2):143–152, 2003.

[21] W. Yang, J. Novembre, E. Eskin, and E. Halperin. A model-based approach for analysis of spatial structure in genetic data. *Nature Genetics*, 44(6):725–731, 2012.

[22] J. M. Young, L. S. Weyrich, J. Breen, L. M. Macdonald, and A. Cooper. Predicting the origin of soil evidence: High throughput eukaryote sequencing and MIR spectroscopy applied to a crime scene scenario. *Forensic Science International*, 251:22–31, 2015.

[23] F. Zhou, Q. Claire, and R. D. King. Predicting the Geographical Origin of Music. In *IEEE International Conference on Data Mining*, pages 1115–1120, 2014.

IN SEARCH OF THE CONSENSUS AMONG MUSICAL PATTERN DISCOVERY ALGORITHMS

Iris Yuping Ren Utrecht University y.ren@uu.nl
Hendrik Vincent Koops Utrecht University h.v.koops@uu.nl
Anja Volk Utrecht University a.volk@uu.nl
Wouter Swierstra Utrecht University w.s.swierstra@uu.nl

ABSTRACT

Patterns are an essential part of music and there are many different algorithms that aim to discover them. Based on the improvements brought by using data fusion methods to find the consensus of algorithms on other MIR tasks, we hypothesize that fusing the output from musical pattern discovery algorithms will improve the pattern discovery results. In this paper, we explore two methods to combine the pattern output from ten state-of-the-art algorithms using two datasets. Both provide human-annotated patterns as ground truth. We show that finding the consensus among the output of different musical pattern discovery algorithms is challenging for two reasons: First, the number of patterns found by the algorithms exceeds patterns in human annotations by several orders of magnitude, with little agreement on what constitutes a pattern. Second, the algorithms perform inconsistently across different pieces. We show that algorithms lack a consensus with each other. Therefore, it is difficult to harness the collective wisdom of the algorithms to find ground truth patterns. The main contribution of this paper is a meta-analysis of the (dis)similarities among pattern discovery algorithms' output and using the output in two fusion methods. Furthermore, we discuss the implication of our results for the MIREX task.

1. INTRODUCTION

An important property of music is its recurring structures [18]. Musically meaningful repetitions in the form of musical patterns or musical motifs [29] provide one of the most intensely researched aspects both for analyzing individual musical pieces [24] and groups or collections of musical pieces for identifying musical style based on musical patterns [8, 23, 34]. Automatic pattern discovery is an active research area in Music Information Retrieval (MIR) that aims to discover these patterns automatically. Different pattern discovery methods have been introduced, such as string-based approaches [4, 7, 14, 16, 17, 25], geometric approaches [3, 6, 21, 31], data mining approaches [28], and machine learning approaches [26]. Musical pattern discovery algorithms have been used for different applications: for determining similarity between musical pieces [1], for automatic compositions [11], and for describing musical style characteristics [8].

Although many approaches have been developed over the recent decades (for a detailed overview see [12]), musical pattern discovery algorithms face a number of challenges. Music is inherently ambiguous: musicologists often do not agree on what the important musical patterns are in a given piece [5]. This makes it difficult to evaluate the quality of automatically extracted musical patterns. Furthermore, each algorithm is historically tested on unassociated datasets with disparate metrics [12]. One attempt to systematically evaluate the algorithms is the MIREX Discovery of Repeated Themes & Sections task initiated in 2014. In the task, a pattern is defined as a set of time-pitch pairs that occurs at least twice in a piece of music [10]. Although the state-of-the-art algorithms cannot reproduce the human-annotated patterns yet, they perform acceptably well according to the evaluation metrics in this task. However, the algorithms perform inconsistently across different pieces which makes it hard to determine whether there exists a single 'best' performing algorithm.

Another problem is that algorithms tend to find far more patterns than human annotators do [10]. Hence the challenge is to find which potential patterns are musically meaningful. The poor performance of automatically extracted patterns in the compression and classification task on the Dutch Song Database in [1] also shows that pattern discovery is far from being a solved problem in MIR and Computational Music Analysis.

Integrating different algorithms using data fusion has been shown to be a successful approach to improving overall performance in other areas dealing with ambiguous musical data, such as in Automatic Chord Estimation [15]. To address the challenges in musical pattern discovery, we hypothesize that integrating the output of state-of-the-art algorithms to find a consensus among these algorithms will help us to achieve an overall better pattern discovery result. To this end, we explore two fusion methods: a new algorithm, the Pattern Polling Algorithm (PPA), and the Time Indexed Novelty Algorithm (TINA), which is based on commonly used time indexed novelty scores. Using these two methods, we aim to integrate the patterns found by multiple pattern discovery algorithms to a consensus and therefore employ their collective wisdom.

Fusing the patterns produced by individual algorithms is challenging since there are different assumptions, datasets and methods behind the development of each algorithm. By exploring PPA and TINA using the MIREX dataset and the Annotated Corpus from the Dutch Song Database [32], we identify two problems with using these fusion methods. First, because the number of patterns taken as input of the fusion process is several orders of magnitude larger than the human-annotated ground truth patterns and they are disparate in terms of the pattern location, the pattern length, pattern overlap, and pattern coverage of the music pieces, it makes it difficult to find agreements among these patterns. The disagreement reflects the ambiguity of the pattern discovery task and a need for better definitions of musical patterns. Second, the individual algorithms perform inconsistently on different pieces of music. The lack of large musical pattern discovery data sets aggregates the issue of the inconsistency and prevents further improvements on using machine learning algorithms.

In this paper, we make two main **contributions**: First, we undertake a meta (dis)similarity comparison among the output of musical pattern discovery algorithms using two fusion methods, TINA and PPA (Section 2 and Section 3). Second, based on this research, we discuss issues of the MIREX Discovery of Repeated Themes & Sections task and suggest future directions for improving musical pattern discovery research (Section 4).

2. METHODS

In this section, we introduce the two fusion methods of PPA and TINA along with our evaluation methods. We use the MIREX monophonic version of Chopin's Mazurka Op. 24 No. 4 as an example to illustrate the algorithms. The code of the algorithms and supportive explanations can be found in https://github.com/irisyupingren/2017Pattern.

2.1 Algorithms Overview

The two new methods we use to explore musical pattern fusion have different goals. PPA focuses on using the gathered information to extract local pattern features (pattern boundaries), while TINA focuses on globally integrating the output patterns of individual algorithms to a probability distribution (pattern distribution).

We devise PPA based on the fact that all pattern discovery algorithms aim at finding the salient parts in musical compositions. We assume that each algorithm's output can be taken as a vote on whether or not a given time point participates in a salient part of the composition, e.g. is part of a musical pattern. Moreover, we define a salience degree of a time point which corresponds to the number of patterns that the time point participates in. In essence, the PPA is a voting system in which each algorithm votes on the salience degree of a time point based on the discovered patterns. The resulting *polling curve* is then taken as a base to detect pattern beginnings and endings.

TINA is devised based on taking the polling curve and the ground truth patterns and normalize them to a probability distribution. Along with the polling curve, we use the time indexed novelty score [9], which is produced by correlating a checkerboard kernel along the main diagonal of the similarity matrix of pattern votes. The time indexed novelty scores are then taken as a base to compare with the pattern distributions of individual algorithms, the polling curve, and the human-annotated patterns.

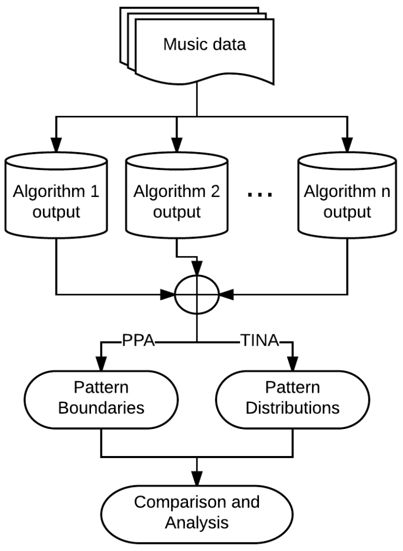


Figure 1. The pipeline of the fusion and evaluation. Same datasets and evaluation methods are used to compare two fusion methods (PPA and TINA) with individual algorithms.

The pipeline of the entire fusion and evaluation process can be found in Figure 1. For a set of music data and musical pattern discovery algorithms, we first determine the musical patterns discovered by each algorithm on each musical piece. Then we use PPA to extract pattern boundaries and use TINA to calculate the pattern distributions. Finally, we analyze the fusion results and the individual algorithms.

2.2 Pattern Polling Algorithm (PPA)

PPA starts with calculating a polling curve by taking into account all musical patterns output of all algorithms. After smoothing the polling curve, the algorithm takes the critical points (i.e. where the derivatives equal to zero) of the curve and the first derivative as the boundaries of the patterns (the beginnings and endings of the patterns). This is because changes in salience values could potentially reveal structural changes in music.

Polling Curve. The polling curve (PC) is created using the output from all individual algorithms. We let each algorithm vote at a given time point to decide whether it is a salient part of the music. To create the voting time points in the music, we use the resolution of one quarter note length. The time points where the algorithms vote are therefore in the vector $T := [0, 1, ..., n]$ with the unit of a quarter note.

The voting is realized by looking up discretized time points in the occurrences of output patterns: if there is an occurrence interval which covers the time point, we count that there is a valid vote. Finally, we add up the voting

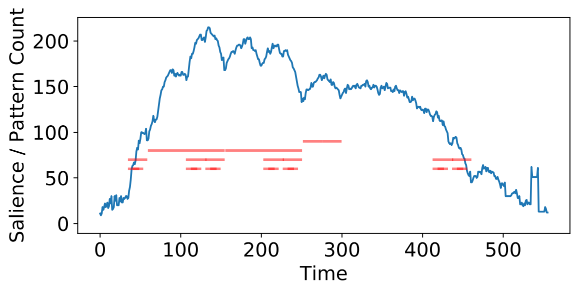


Figure 2. The polling curve of Chopin's Mazurka Op. 24 No. 4 using algorithms from the MIREX task (see Section 3). The horizontal bars show where the ground truth patterns are present. The x-axis represents time in the unit of quarter note and the y-axis represents the salience value, which is the number of pattern counts if each vote carries the same weight. We see promising correspondences between the polling curve and human annotations.

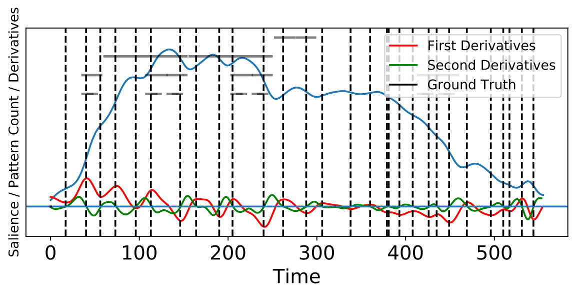


Figure 3. Extracted pattern boundaries using PPA. The dashed vertical lines are the boundaries. Many dashed lines are aligned with the boundaries of human annotations. We also plotted the polling curve, the ground truth, first and second derivatives for reference.

from all the algorithms and produce the polling curve $P(t)$, which is a time series consisting of the salience values at time points in T.

Since PPA uses a combination of algorithms, we should consider which algorithms we want to include or exclude. PPA could be extended if we have extra information on which algorithms should be trusted more and make a portfolio of algorithms as the input. The portfolio is essentially a way of assigning binary weights to the algorithms' votes: when the algorithm is included, its patterns have weight one, and when not included, weight zero. We can also generalize the weight to a continuous value.

To formalize the process of voting:

$$P(t) = \sum_A \sum_P \sum_O I_O^{A,P}(t) \tag{1}$$

where A stands for Algorithm, P stands for Pattern, O stands for occurrence, and $I_O^{A,P}(t)$ is the weighted indicator function of an occurrence in the pattern P in the algorithm A:

$$I_O^{A,P}(t) = \begin{cases} \omega_A & t \in O \subseteq P \subseteq A \\ 0 & t \notin O \subseteq P \subseteq A \end{cases} \tag{2}$$

where ω_A is the weight assigned to algorithm A.

An exemplary polling curve of Chopin's Mazurka Op. 24 No. 4 using several algorithms from the MIREX task is shown in Figure 2. The polling curve provides us with a clue of where there is a salience change in the music. Critical values (i.e. prominent changes) in salience values will be regarded as boundaries in the polling curve times series. In the following subsection, we will explain how to decide what are the prominent changes and how to reduce the possibly irrelevant micro-changes in the polling curve and then find the pattern boundaries.

Smoothing. One common way to reduce the effects of possibly irrelevant micro-changes in time series is smoothing. In our algorithm, we use the Savitzky-Golay filter [30], which is a linear least-square polynomial fitting filter. Each time we apply the smoothing, we reduce some effects of micro-changes, but at the same time, we might also lose potentially valuable details. With different degrees of smoothing, we capture different levels of details in salience's changes. Therefore, we make the degree of smoothness, s, to be one of the two parameters in PPA.

Derivative. After smoothing, to find the prominent changes of the salience in music, we calculate the first and second discrete derivatives of the polling curve and take their critical zero-crossing points as the pattern boundaries. More formally: let $P'(t) = P(t+1) - P(t)$ and let $P''(t) = P'(t+1) - P'(t)$, $t > 0, t \in T$. We are interested in the zero crossing $\bar{t}$ of $P'(t)$ and $P''(t)$ because the zero crossing points $\bar{t}$ represent a change of direction in the polling curve. For example, when $P'(t) < 0$ and $P'(t+1) > 0$, we have a dipping point $P'(\bar{t}) = 0$ in the curve. There are more patterns discovered starting from this point: it is likely to be a beginning of a pattern.

One question remains as for how strong the dipping, tipping, concave and convex in the curve should be so that we pick it as a boundary. Here we introduce the second parameter: a threshold on the steepness of the zero crossing points λ. With different values of λ, we create a set of boundary sets which consist of the time at which zero crossing happens. In Figure 3, an example of the extracted boundaries can be found. We notice that some boundaries line up well with ground truth boundaries. We will evaluate the extracted pattern boundaries in Section 2.4.

2.3 Time Indexed Novelty Algorithm (TINA)

Since PPA extracts local boundaries, we use TINA to assess globally how the extracted patterns are similar to human-annotated patterns. Using the notions provided in Section 2.2, TINA can be described concisely as follows: We use the pattern vote representation in Equation (2) as the input. Formally, the input matrix is $M = (I_O^{A,P_1}(t); I_O^{A,P_2}(t); ..., I_O^{A,P_n}(t))$, where n is the count of output patterns we would like to combine. The main component of TINA is the calculation of the time indexed novelty scores described in [9]. This includes calculating the similarity matrix S of M using the Euclidean distance and then multiplying the diagonal with a checkerboard kernel $K = (1, -1; -1, 1)$, which gives us the novelty curve

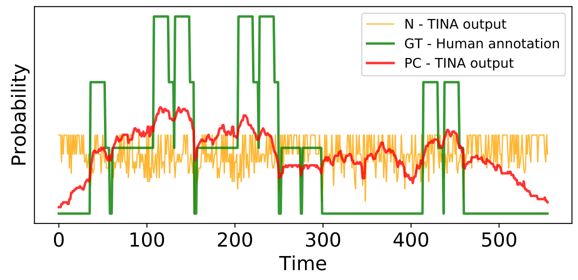


Figure 4. The TINA output novelty curve (N) distribution calculated using patterns from all algorithms, the TINA output polling curve (PC) distribution calculated using patterns from MIREX algorithms (see Section 3) and the ground truth (GT) pattern distribution. The x-axis represents time in the unit of quarter note. Correspondences of the time series can be seen from the three curves.

$N(t)$. The novelty curve represents the changing rate of the pattern vote time series $I_O^{A,P_i}(t)$, serving the same role as the derivatives in PPA. In the end, we obtain a novelty curve for each algorithm and the ensemble of algorithms, depending on which patterns are included in M.

Next, the comparison in TINA requires the input from the human-annotated ground truth patterns and the polling curve. To convert the ground truth into the same time series format as the novelty curve and the polling curve, we construct the polling curve from the ground truth patterns as $GT(t)$. Furthermore, taking the frequentists point of view, we normalize the time series by the sum of the entire time series so that we get the distributions of the novelty curve $\overline{N}(t)$, the polling curve $\overline{P}(t)$ and the ground truth patterns $\overline{GT}(t)$. Similarly, we can also construct the pattern distributions of individual algorithms $\overline{P}_A(t)$.

In Figure 4, we give an example of the novelty curve distribution, the polling curve distribution and the ground truth pattern distribution. In an initial visual inspect, we see some correspondences among the three curves: some fluctuations and the tipping/dipping points of the curves tend to coincide. We will evaluate the distribution similarities globally in the next subsection.

2.4 Evaluation

We use two evaluation methods to assess how similar the human-annotated patterns are to the output boundaries of PPA and the output distributions of TINA.

Pattern Boundaries. To evaluate the extracted pattern boundaries, we use the boundaries of the ground truth patterns. Following the standard MIREX evaluation metrics, we calculate the precision, recall and F1 score of the boundaries with a degree of fuzziness: we look for a match of boundaries with a tolerance of one quarter note length because of the one-quarter-length discretization we used for creating the polling curve.

Pattern Distribution. To evaluate globally how similar the normalized novelty curve and the polling curve are to the ground truth pattern distribution, we calculate the Bhattacharyya coefficients [13] and the Pearson correlation coefficients. Bhattacharyya coefficients measure the amount of overlaps between two distributions and the Pearson correlation coefficients measure the linear correlation of distributions. For the extracted patterns to be similar with the ground truth patterns, we expect high correlation values and high overlap values.

3. RESULTS

In this section, we first introduce the input we use for PPA and TINA and provide a meta-analysis on the individual algorithms. Then we explore the effects of the two parameters s and λ in PPA and the necessity of cross-validation. Using our evaluation metrics, we show the performance of the two fusion algorithms is on average similar to individual algorithms, and we provide analysis as to why the fusion methods do not excel.

3.1 Input: Algorithms and Music Data

We use two sets of algorithms and music data. The first set is from the Annotated Corpus of the Dutch Song Database (MTC-ANN) and the algorithms used in [1], namely PatMinr [17], MotivesExtractor (ME) [25], SIATEC [22], COSIATEC [19], and MGDP [7]. MTC-ANN [32] consists of 360 dutch folk songs in 26 tune families. Because we are interested in finding shared patterns between songs in the same tune family, the pattern discovery algorithms are computed on the concatenation of the songs in the same tune family, and then the patterns discovered on the boundaries of concatenation are filtered out (same as the intra-opus task described in [1]). The 360 individual songs are taken as the input of PPA and TINA.

The second set is from the MIREX Discovery of Repeated Themes & Sections task. For music data, we use a subset of the task's training dataset. The original dataset contains five pieces in both polyphonic and monophonic format. We take three pieces in the monophonic format: Chopin's Mazurka Op. 24 No. 4, Mozart's Piano Sonata K. 282, 2nd movement, and Beethoven Piano Sonata Op. 2 No. 1, 3rd movement. For the sake of the consistency of the task and the compatibility with MTC-ANN, we leave out the two music pieces which are constructed by a concatenation of voices in the piece. The algorithm input consists of all algorithms submitted to the MIREX task during 2014-2016: MotivesExtractor (ME) [25], SIATECCompress-TLP (SIAP), SIATECCompress-TLF1 (SIAF1), SIATECCompress-TLR (SIAR) [20], OL1 & OL2 [17], VM1 & VM2 [33], SYMCHM (SC) [27], along with SIARCT-CFP (SIACFP) [6], the algorithm developed by the task captain. The output patterns of these state-of-the-art algorithms for our example piece are shown in Figure 5. We make several observations:

1. Different algorithms find very different patterns: some tend to find shorter patterns, some longer; some find many patterns while others are more "picky".
2. We have three algorithm families (SIA, VM, and OL) which consist of more than one algorithm. The algorithms from the same algorithm family tend to find sim-

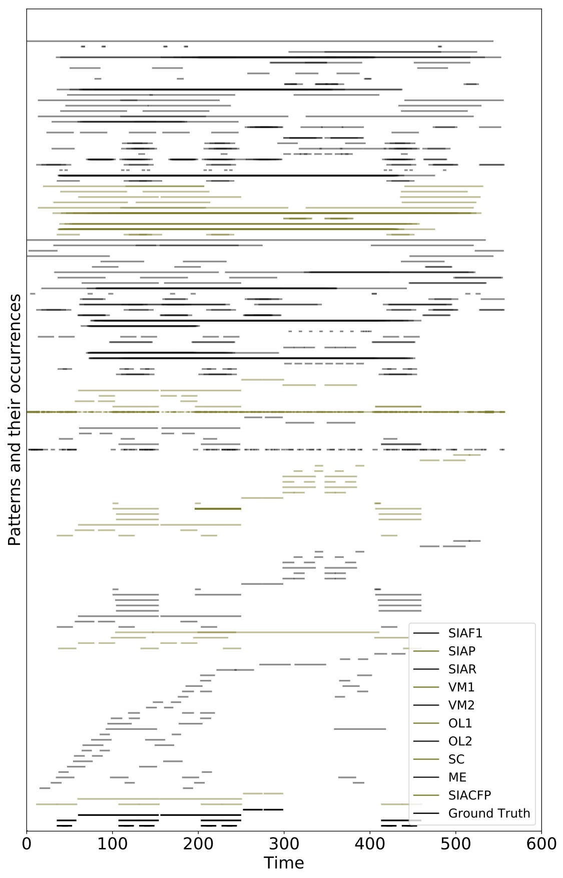


Figure 5. Patterns extracted by all algorithms submitted to the MIREX task 2014-2016 plus SIARCT-CFP on the monophonic Chopin's Mazurka Op. 24 No. 4. The horizontal bars show where the patterns are present. The x-axis represents time in the unit of quarter note. We can see the algorithms find different amount of patterns and patterns of different length, etc.

ilar patterns. Similarities here include the number of patterns discovered, the coverage of the song and the overlaps of the occurrences.

3. The ground truth is sparse in comparison to the patterns discovered by the algorithms.
4. From eyeballing the entire visualization, we see some correspondence and similarities between the algorithms and the ground truth patterns.

3.2 PPA: Parameter Space and Cross Validation

PPA extracts the local boundaries using the output patterns from individual algorithms. We start investigating the effects s and λ in PPA using the MIREX set as input, because a small number of music pieces gives us a clear idea of the relation between the parameters and the performance of PPA. Ideally, if there is a consistent best-performing s and λ across the three pieces for precision, recall and F1 score, it would be possible that the parameters can be generalized. However, we find that no single choice of s and λ performs well across all pieces. Nevertheless, to avoid over-fitting using the ground truth patterns, we perform a three-fold cross-validation using a split of two-pieces training and one piece testing in the MIREX dataset. The results of the MIREX set are shown in Table 1 and the results of MTC-ANN are shown in Table 2.

Algorithm	Precision	Recall	F1
ME	(0.125, 0.086)	(0.184, 0.077)	(0.149, 0.083)
SC	(0.396, 0.022)	(0.419, 0.068)	(0.402, 0.046)
OL1	(0.420, 0.038)	(0.565, 0.044)	(0.462, 0.023)
OL2	(0.422, 0.061)	(0.565, 0.044)	(0.483, 0.054)
SIAF1	(0.139, 0.049)	(0.670, 0.005)	(0.228, 0.041)
SIAR	(0.213, 0.039)	(0.427, 0.000)	(0.279, 0.021)
SIAP	(0.117, 0.043)	(0.596, 0.008)	(0.195, 0.037)
VM1	(0.137, 0.035)	(**1.0**, **0.0**)	(0.240, 0.029)
VM2	(0.206, 0.073)	(0.543, 0.024)	(0.296, 0.060)
SIACFP	(**0.819**, 0.030)	(0.82, 0.064)	(**0.815**, 0.046)
PPA-P	**0.478**	0.206	0.249
PPA-R	0.228	**0.867**	0.35
PPA-F1	0.248	0.738	**0.360**

Table 1. MIREX: (Mean, Variance) of the precision, recall and F1 score of different algorithms at the pattern boundary extraction task. The PPA-P, PPA-R and PPA-F1 are obtained using a 3-fold cross-validation training process optimizing the precision, the recall and the F1 scores. Because we only have one piece in the test set, there is no variance value. Bold numbers are the best results from individual algorithms and PPA.

Algorithm	Precision	Recall	F1
PatMinr	(0.465, 0.054)	(0.957, 0.020)	(0.598, 0.050)
ME	(0.366, 0.103)	(0.353, 0.098)	(0.314, .0879)
COSIATEC	(0.482, 0.049)	(0.774, 0.042)	(0.569, 0.040)
SIATEC	(0.468, 0.046)	(**0.975**, 0.017)	(**0.610**, 0.041)
MGDP	(**0.515**, 0.072)	(0.754, 0.093)	(0.557, 0.065)
PPA-P	(**0.489**, 0.135)	(0.201, 0.023)	(0.264, 0.035)
PPA-R	(0.486, 0.057)	(**0.657**, 0.046)	(**0.534**, 0.044)
PPA-F1	(0.477, 0.054)	(0.652, 0.047)	(0.526, 0.042)

Table 2. MTC-ANN results in the format of Table 1, the only difference being that we use a 10-fold cross-validation. Best results are bold.

In the MIREX set, the best F1 score of PPA ranks the fifth out of ten when using the optimal parameters found by cross-validation. The best F1 score of PPA 0.360 is better than the average of the F1 scores of individual algorithms 0.3549. The SIACFP algorithm performs overall the best on the MIREX set. With small differences, PPA ranks the fourth out of five algorithms in MTC-ANN. However, the best F1 score 0.534 of PPA is better than the average F1 score of four individual algorithms 0.510. Although PatMinr has the best F1 score in this set of music data and algorithms, other algorithms follow very closely and therefore it is hard to determine whether there is a best algorithm in this set of data and algorithms. On both datasets, we observe that PPA performs slightly better than the average of the individual algorithms.

3.3 TINA: Pattern Distributions

From a global point of view, to measure the similarities of novelty distributions, we calculate the polling curve distributions and the pattern distributions of ground truth and individual algorithms using TINA. To evaluate how similar the distributions are, we calculate the Bhattacharyya coef-

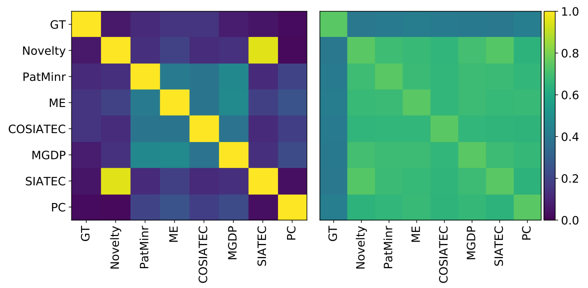


Figure 6. Left: Pairwise Pearson correlation coefficients of the ground truth distribution, individual algorithm distributions, the novelty curve distribution (Novelty) and the polling curve (PC) distribution using the 360 songs in MTC-ANN. All p-values $\ll 0.05$. Right: Pairwise Bhattacharyya coefficients of the same distributions.

ficients and the Pearson coefficients. The pairwise values of the two measurements of the MTC-ANN set of music data and algorithms are shown in Figure 6.

An obvious observation in both figures is the large distance and small correlation between the ground truth pattern distribution and the output of algorithms. Using the Bhattacharyya coefficients, we see that, in comparison to the distribution overlap differences between the ground truth and the individual algorithms, the differences among the individual algorithms and the fusion algorithms are smaller. Using the Pearson correlation coefficients, we see less linear correlation between the polling curve distribution and the ground truth distribution. Other algorithms have similar correlation values except SIATEC and the novelty curve, which have specially high correlation. This means the novelty curve is largely based on the SIATEC algorithm's output, and this is caused by the large number of output patterns generated by the algorithm. Looking at both figures, from a global point of view, output of the algorithms have similarities among themselves, but they show less correlation and similarity compared to the ground truth patterns. Similar observations are made in the MIREX dataset and hence the matrices are not shown here.

3.4 Analysis on the Results

Combining all evaluation results, we identify why the fusion methods do not excel over individual algorithms as the fusion approach applied in [15]. First, the available datasets are small and the ground truth patterns are sparse, which is problematic for training the parameters and evaluating a stable performance. Second, the algorithms disagree with each other on pattern length and pattern overlap etc., which reflects the inherent ambiguity of music and a lack of unified goal/application of the musical pattern discovery task. Third, because there are well-performing algorithms and relatively less well-performing algorithms in the fusing portfolio, fusion results are understandably of average quality since it combines results from all these different sources. In the end, although we observed promising correspondence and consensus among algorithms in Figure 4 and Figure 5, a systematic evaluation reveals that the degree of consensus is not yet enough for helping to find patterns that agree with the annotated patterns.

4. DISCUSSION AND CONCLUSION

In this paper, we attempt to combine the output of musical pattern discovery algorithms to improve musical patterns discovery. We devise a new algorithm, PPA, and apply an established method from the audio music similarity field, TINA, to musical pattern discovery. We test the fusion algorithms on pieces in the MIREX and MTC-ANN datasets. The results show that PPA and TINA on average do not improve the performance significantly. More specifically, the results from PPA show that we can extract local boundaries using a combination of musical pattern discovery algorithms, but we need to select the parameters properly. The results from TINA show that the ground truth probability distributions of musical patterns are different from the ones produced by algorithms. The results of using two datasets show that algorithms perform differently given different pieces and it is sometimes hard to select a single ‘winner’. The reason of the dissatisfying performance of the fusion algorithms lies in a large number of disagreeing patterns and the sparsity of the human-annotated patterns: the salient parts of music identified by the extracted musical patterns do not align with the human annotations. To break the current limitations of applying data fusion in this domain, our work implies a need for an improved dataset and musical pattern discovery task formulation. It is also possible to improve the fusion methods by incorporating and learning more parameters from the data source.

MIREX From using the MIREX dataset in the fusion task, we identify three potential improvements for the task. First, the ground truth data from the MIREX dataset is sparse and consists of only a few pieces. It would be desirable to obtain more annotations from experts. In addition, the current ground truth consists of annotations from different sources, which could be improved by adopting a collaborative ground truth creation process [2]. Second, an open question is whether the patterns of algorithms should be compared to humanly annotated patterns as a way of evaluation, given that musicologists often disagree on the patterns: more aspects of subjectivity should be taken into account. In addition, since we see that pattern discovery algorithms produce very different patterns, one might ask whether different algorithms' output might be useful for different application scenarios. In the future of the MIREX task, instead of measuring the agreement with annotated patterns only, the testing of pattern quality by providing a range of subtasks which employ extracted patterns into various applications, constitutes a promising direction for improving the evaluation of pattern discovery algorithms.

Acknowledgements. We thank all authors of the algorithms for providing their algorithms and output, and the anonymous reviewers for valuable suggestions.

5. REFERENCES

[1] Peter Boot, Anja Volk, and W. Bas de Haas. Evaluating the role of repeated patterns in folk song classification and compression. *Journal of New Music Research*, 45(3):223–238, 2016.

[2] John Ashley Burgoyne, Jonathan Wild, and Ichiro 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*, pages 633–638, 2011.

[3] Chantal Buteau and Guerino Mazzola. Motivic analysis according to Rudolph Reti: Formalization by a topological model. *Journal of Mathematics and Music*, 2(3):117–134, 2008.

[4] Emilios Cambouropoulos. Musical parallelism and melodic segmentation. *Music Perception*, 23(3):249–268, 2006.

[5] Tom Collins. Improved methods for pattern discovery in music, with applications in automated stylistic composition. *PhD thesis. Milton Keynes, UK: Faculty of Mathematics, Computing and Technology, The Open University*, 2011.

[6] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, pages 549–554, 2013.

[7] Darrell Conklin. Discovery of distinctive patterns in music. *Intelligent Data Analysis*, 14(5):547–554, 2010.

[8] Darrell Conklin and Christina Anagnostopoulou. Comparative pattern analysis of cretan folk songs. *Journal of New Music Research*, 40(2):119–125, 2011.

[9] Matthew Cooper and Jonathan Foote. Summarizing popular music via structural similarity analysis. In *Proceedings of the 7th IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 127–130. IEEE, 2003.

[10] Music Information Retrieval Evaluation eXchange (MIREX) 2013. Discovery of Repeated Themes & Sections. `http://www.musicir.org/mirex/wiki/2013`. Accessed: 2017-05-04.

[11] Dorien Herremans and Elaine Chew. Morpheus: Automatic music generation with recurrent pattern constraints and tension profiles. *Technical Report, Queen Mary University of London (2016).*

[12] Berit Janssen, W. Bas de Haas, Anja Volk, and Peter van Kranenburg. Finding repeated patterns in music: State of knowledge, challenges, perspectives. In *Proceedings of the 10th International Symposium on Computer Music Modeling and Retrieval*, pages 277–297. Springer, 2013.

[13] Thomas Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967.

[14] Ian Knopke and Frauke Jürgensen. A system for identifying common melodic phrases in the masses of Palestrina. *Journal of New Music Research*, 38(2):171–181, 2009.

[15] Hendrik Vincent Koops, W. Bas de Haas, Dimitrios Bountouridis, and Anja Volk. Integration and quality assessment of heterogeneous chord sequences using data fusion. In *Proceedings of the 17th International Society for Music Information Retrieval Conference*, pages 178–184, 2016.

[16] Olivier Lartillot. Multi-dimensional motivic pattern extraction founded on adaptive redundancy filtering. *Journal of New Music Research*, 34(4):375–393, 2005.

[17] Olivier Lartillot. PatMinr: In-depth motivic analysis of symbolic monophonic sequences. *Music Information Retrieval Evaluation eXchange (MIREX 2014)*, 2014.

[18] Elizabeth Hellmuth Margulis. *On repeat: How music plays the mind*. Oxford University Press, 2014.

[19] David Meredith. COSIATEC and SIATECCompress: Pattern discovery by geometric compression. In *Music Information Retrieval Evaluation Exchange (MIREX 2013)*, 2013.

[20] David Meredith. Using SIATECCompress to discover repeated themes and sections in polyphonic music. In *Music Information Retrieval Evaluation Exchange (MIREX 2016)*, 2016.

[21] David Meredith, Kjell Lemström, and Geraint A. Wiggins. Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, 31(4):321–345, 2002.

[22] David Meredith, Geraint A. Wiggins, and Kjell Lemström. Pattern induction and matching in polyphonic music and other multidimensional datasets. In *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics*, pages 22–25, 2001.

[23] Leonard B. Meyer. *Style and music: Theory, History, and Ideology*. Chicago: University of Chicago Press, 1989.

[24] Jean-Jacques Nattiez and Jonathan M. Dunsby. Fondements d'une sémiologie de la musique. *Perspectives of New Music*, 15(2):226–233, 1977.

[25] Oriol Nieto and Morwaread M. Farbood. Identifying polyphonic patterns from audio recordings using music segmentation techniques. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 411–416, 2014.

[26] Matevz Pesek, Ales Leonardis, and Matija Marolt. A compositional hierarchical model for music information retrieval. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 131–136, 2014.

[27] Matevz Pesek, Urša Medvešek, Aleš Leonardis, and Matija Marolt. SymCHM: a compositional hierarchical model for pattern discovery in symbolic music representations. *Music Information Retrieval Evaluation Exchange (MIREX 2015)*, 2015.

[28] Iris Yuping Ren. Closed patterns in folk music and other genres. In *Proceedings of the 6th International Workshop on Folk Music Analysis*, 2016.

[29] Rudolph Reti. *The Thematic Process in Music*. New York: Macmillan, 1951.

[30] Ronald W. Schafer. What is a Savitzky-Golay filter? *IEEE Signal Processing Magazine*, 28(4):111–117, 2011.

[31] Wai Man Szeto and Man Hon Wong. A graph-theoretical approach for pattern matching in post-tonal music analysis. *Journal of New Music Research*, 35(4):307–321, 2006.

[32] Peter van Kranenburg, Berit Janssen, and Anja Volk. The Meertens Tune Collections: The Annotated Corpus (MTC-ANN) versions 1.1 and 2.0.1. *Meertens Online Reports*, 2016(1), 2016.

[33] Gissel Velarde and David Meredith. A wavelet-based approach to the discovery of themes and sections in monophonic melodies. *Music Information Retrieval Evaluation Exchange (MIREX 2014)*, 2014.

[34] Anja Volk, W. Bas de Haas, and Peter van Kranenburg. Towards modelling variation in music as foundation for similarity. In *Proceedings of the 12th International Conference on Music Perception and Cognition*, pages 1085–1094, 2012.

INFORMED AUTOMATIC METER ANALYSIS OF MUSIC RECORDINGS

Ajay Srinivasamurthy[*] ajays.murthy@upf.edu

Andre Holzapfel[†] holzap@kth.se

Xavier Serra[*] xavier.serra@upf.edu

[*]Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[†]Media Technology and Interaction Design Department, KTH Royal Institute of Technology, Stockholm, Sweden

ABSTRACT

Automatic meter analysis aims to annotate a recording of a metered piece of music with its metrical structure. This analysis subsumes correct estimation of the type of meter, the tempo, and the alignment of the metrical structure with the music signal. Recently, Bayesian models have been successfully applied to several of meter analysis tasks, but depending on the musical context, meter analysis still poses significant challenges. In this paper, we investigate if there are benefits to automatic meter analysis from additional *a priori* information about the metrical structure of music. We explore informed automatic meter analysis, in which varying levels of prior information about the metrical structure of the music piece is available to analysis algorithms. We formulate different informed meter analysis tasks and discuss their practical applications, with a focus on Indian art music. We then adapt state of the art Bayesian meter analysis methods to these tasks and evaluate them on corpora of Indian art music. The experiments show that the use of additional information aids meter analysis and improves automatic meter analysis performance, with significant gains for analysis of downbeats.

1. INTRODUCTION

Automatic meter analysis of a music recording aims at determining different components of its metrical structure such as the type of meter, the tempo, the beats and downbeats. It is an important Music Information Research (MIR) task that provides useful musically relevant metadata not only for enriched listening, but also for preprocessing of music for several higher level tasks such as section segmentation, structural analysis and defining rhythm similarity measures. Initial approaches to meter analysis explored individual tasks of meter analysis, such as tempo estimation [8, 9], beat tracking [5, 13], time signature estimation [15] and downbeat tracking [10, 14]. Recent approaches consider a joint estimation of several of these components and have successfully applied Bayesian models to jointly estimate beat and downbeats using rhythmic patterns learned from onset detection features [1, 11, 12]. Recent interest has also been to explore neural networks for beat and downbeat tracking with several musically inspired features and network topologies [7]. Despite the recent success, meter analysis still poses significant challenges depending on the musical context [18, 20].

In this paper, we investigate the potential to improve meter analysis methods by providing them with additional prior information about the underlying metrical structure. This is a research problem we define as *informed meter analysis*, referring to a class of analysis tasks that utilize some form of additional information about the underlying metrical structure of the piece. Apart from building meter-aware analysis methods, informed meter analysis is motivated by its potential applications and the need for improved meter analysis performance. It is hypothesized that information available as metadata or obtainable from an expert user can be effectively utilized to significantly improve meter analysis performance. Such informed approaches can help to establish a focus in the space of possible solutions by the incorporation of *a priori* information, supporting meter analysis especially in the context of computationally challenging samples. Some informed meter analysis tasks have been studied before, such as the task of downbeat tracking from a set of known beats [10]. However, there has been no formal treatment of the problem, which is the focus of this paper.

Carnatic and Hindustani music are Indian Art Music (IAM) traditions from Southern and Northern parts of the Indian subcontinent, respectively. Both these musics have a long history of performance and continue to thrive in current sociocultural contexts. While the two musics differ in performance practices, they share similar melodic and rhythmic concepts. The rhythmic framework is based on cyclic metrical structures called the tāḷa in Carnatic music (CM) or tāl in Hindustani music (HM), which provide a broad structure for repetition of music phrases, motifs and improvisations. A cycle of a tāḷa (or tāl) is divided into isochronous beats (called the mātrā in HM), which are grouped into possibly unequal length sections. The beginning of a cycle (the downbeat) is referred to as sama (sam in HM). Given the central importance of tāḷa in defining rhythmic structures, meter analysis in the context of IAM aims to time-align and tag a music recording with tāḷa related events and metadata. Clayton [3] and Sambamoorthy [16] provide an in depth discussion of rhythm in Hindustani and Carnatic music, respectively.

With significant improvisation and expressive timing, a wide range of tempo and cycles as long as a minute, IAM has been shown to pose several challenges to automatic meter analysis [20]. Further, large and continuously growing archives of IAM are available with varying amounts of tāḷa related metadata [17]. In this paper, we use corpora of IAM as a challenging case to explore the potential of informed meter analysis, and include a set of Ballroom dances to enable comparison with other styles.

2. INFORMED METER ANALYSIS

Different kinds of prior information about the underlying metrical structure can be made available to analysis algorithms. In the following subsections, we describe specific informed analysis tasks and emphasize different practical scenarios for each task. At the outset, we assume that some basic information about the music piece is available for all informed analysis tasks. We assume that the music tradition is known, and that the rhythm class (tāḷa) of the piece is from a set of known (from musicological literature) tāḷas. Further, we assume we know the range of tempo generally used in a music culture. A piece of IAM is performed in a single tāḷa (rare exceptions exist, but outside the scope of regular performance practice) and most commercial releases are segmented so that an audio recording is a single piece. However, there are cases when an entire concert or parts of concert with multiple pieces (and hence possibly different tāḷas) are stored in a single audio recording. We assume that such a recording has been segmented into individual pieces of music with a single tāḷa. The case of change of tāḷas within a recording is not addressed.

Finally, for better readability, we use the commonly used terminology of tempo, beats and downbeats in the paper, while we carefully note that the equivalence of these terms across different music cultures cannot be assumed.

2.1 Meter Inference (`Inference`)

Meter inference aims for a complete meter analysis of a recording starting with no prior information. Given an audio music recording, meter inference task aims to estimate the rhythm class (or meter type or tāḷa), time-varying tempo, beats and downbeats. Meter inference in IAM aims to recognize the tāḷa/tāl, estimate the time varying tempo (measured as the inter beat interval), the beat and the sama/sam (downbeat) locations. It is the least informed and most difficult task owing to the large range of tempi and different tāḷas. While meter inference is the only applicable task with unlabeled collections of music, it is often the case that some tāḷa related information is available or can be inferred, e.g. from the editorial metadata of a music piece. Most of commercially released music in both Carnatic and Hindustani music has the name of the tāḷa in editorial metadata. Even within a live concert, the musician often announces the tāḷa of a piece and hence tāḷa recognition is a redundant task. However, meter inference can be used as a baseline task to understand the complexity of uninformed meter analysis.

2.2 Meter Tracking (`Track`)

Given an audio music recording and its rhythm class (or meter type or tāḷa), meter tracking aims to estimate the time varying tempo, beat and downbeat locations. Meter tracking in IAM aims to track the time varying tempo, beats and the sama from an audio music recording, given the tāḷa. Assuming that the tāḷa, and hence the metrical structure is known in advance is a fair and practical assumption making meter tracking the most relevant meter analysis task for IAM.

2.3 Informed Meter Tracking

Informed meter tracking is meter tracking in which some additional information apart from the tāḷa is available. The additional information could be in the form of a tempo range, a few instances of beats and downbeats annotated, or even partially tracked metrical cycles. The additional metadata could come from manual annotation or as an output of other automatic algorithms, e.g. the median tempo of a piece can be obtained from a standalone tempo estimation algorithm, or some melodic analysis algorithms might output (with a high probability) some beats/downbeats as a byproduct.

From a practical standpoint, while it is prohibitively resource intensive to manually annotate all the beats and downbeats of a large music collection, it might be possible to seed the meter tracking algorithms with the first few beats and downbeats. For a musician or even an expert listener, it would be easy to tap some instances of the beat and sama/downbeats, which could then be used to automatically track meter in the whole recording. In specific, we explore three variants of informed meter tracking, with varying levels of available information:

Sama-informed meter tracking (`SI-Track`) task in which a few instances of sama/downbeat of the piece are provided as an additional input to the meter tracking algorithm. An example downbeat is expected to help the algorithm to better align the audio to the underlying meter. We only explore the use of first downbeat of the piece, without any knowledge of tempo.

Tempo-informed meter tracking (`TI-Track`) task in which the median tempo (or a narrow range of tempi) of the piece is provided as an additional input to the meter tracking algorithm. Providing the median tempo is hypothesized to help reduce metrical level errors - tracking the metrical cycles at the correct metrical level instead of tracking half and double cycles. The median tempo can be obtained manually or through other automatic tempo estimation algorithms [8, 22].

Sama-Tempo-informed meter tracking (`STI-Track`) task in which the median tempo and a few downbeat locations in the excerpt are provided as additional inputs to the meter tracking algorithm. We only explore the use of median tempo value and the first downbeat of the music piece provided to the meter tracking algorithm.

The informed meter tracking tasks formulated in this section are relevant and designed to require minimal human effort to provide the necessary additional information. In a best case scenario, the most informed `STI-Track` task can be applied to a music piece by listening to just the first few

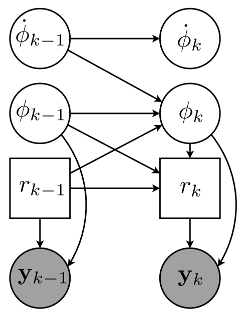


Figure 1: The bar pointer model for meter analysis. The circles and squares denote continuous and discrete variables, respectively. Grey nodes and white nodes represent observed and latent variables, respectively.

seconds of the piece and marking two consecutive downbeats. An estimate of the initial tempo can be obtained the two downbeats and used by the analysis algorithm. Finally, the various tasks were described using terminology of IAM, but they are applicable to any music with hierarchical metrical structures that can be described with beats, downbeats and rhythm patterns.

3. METER ANALYSIS MODEL

To compare different informed analysis tasks, we use and adapt a state of the art Dynamic Bayesian Network (DBN). Referred to as bar pointer model (BP-model) [23], has been successfully applied for meter analysis in different music cultures [11, 19]. We describe the model briefly while a detailed description is presented in [12]. We then explain how it can be adapted to the informed analysis tasks.

In a DBN, an observed sequence of features derived from an audio signal $\mathbf{y}_{1:K} = \{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$ is generated by a sequence of hidden (latent) variables $\mathbf{x}_{1:K} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$, where K is the length of the feature sequence (number of audio frames). The joint probability distribution of hidden and observed variables factorizes as,

$$P(\mathbf{y}_{1:K}, \mathbf{x}_{0:K}) = P(\mathbf{x}_0) \cdot \prod_{k=1}^{K} P(\mathbf{x}_k \mid \mathbf{x}_{k-1})\, P(\mathbf{y}_k \mid \mathbf{x}_k)$$

where, $P(\mathbf{x}_0)$ is the initial state distribution, $P(\mathbf{x}_k|\mathbf{x}_{k-1})$ is the transition model, and $P(\mathbf{y}_k|\mathbf{x}_k)$ is the observation model. The structure of the BP-model in Figure 1 shows the conditional dependence relations between the variables.

3.1 Hidden Variables

At each audio frame k, the hidden variable vector $\mathbf{x}_k$ describes the state of a hypothetical bar pointer $\mathbf{x}_k = [\phi_k \;\; \dot{\phi}_k \;\; r_k]$, representing the bar position, instantaneous tempo and a rhythmic pattern indicator, respectively.

Rhythmic pattern indicator: The rhythmic pattern variable $r \in \{1, \ldots, R\}$ is an indicator variable to select one of the R observation models corresponding to each bar (cycle) length rhythmic pattern of a rhythm class that are learned from training data. Each pattern r corresponds to a rhythm class (or meter type or tāḷa) and has an associated length of cycle M_r and number of beat (or mātrā) pulses B_r.

Bar position: The bar position $\phi \in [0, M_r)$ variable tracks the progression through the bar and indicates a position in the bar at any audio frame. The variable traverses the whole bar and wraps around to zero at the end of the bar to track the next bar.

Instantaneous tempo: Instantaneous tempo $\dot{\phi}$ is the rate at which the bar position variable progresses through the bar at each frame, measured in bar positions per time frame.

3.2 Transition and Observation Model

The initial state distribution $P(\mathbf{x}_0)$ can be used to incorporate prior information about the metrical structure of the music into the model. Given the conditional dependence relations in Figure 1, the transition model factorizes as,

$$P(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = P(\phi_k \mid \phi_{k-1}, \dot{\phi}_{k-1}, r_{k-1}) P(\dot{\phi}_k \mid \dot{\phi}_{k-1}) \\ P(r_k \mid r_{k-1}, \phi_k, \phi_{k-1}) \quad (1)$$

The individual terms of the equation can be expanded as,

$$P(\phi_k \mid \phi_{k-1}, \dot{\phi}_{k-1}, r_{k-1}) = \mathbb{1}_\phi \quad (2)$$

where $\mathbb{1}_\phi$ is an indicator function that takes a value of one if $\phi_k = (\phi_{k-1} + \dot{\phi}_{k-1}) \operatorname{mod}(M_{r_{k-1}})$ and zero otherwise. The tempo transition is given by,

$$P(\dot{\phi}_k \mid \dot{\phi}_{k-1}) \propto \mathcal{N}(\dot{\phi}_{k-1}, \sigma^2_{\dot{\phi}_k}) \times \mathbb{1}_{\dot{\phi}} \quad (3)$$

where $\mathbb{1}_{\dot{\phi}}$ is an indicator function that equals one if $\dot{\phi}_k \in [\dot{\phi}_{\text{min}}, \dot{\phi}_{\text{max}}]$ and zero otherwise, restricting the tempo to be between a predefined range. $\mathcal{N}(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2. The value of $\sigma_{\dot{\phi}_k}$ depends on the value of tempo to allow for larger tempo variations at higher tempi. We set $\sigma_{\dot{\phi}_k} = \sigma_n \cdot \dot{\phi}_{k-1}$, where σ_n $(= 0.02)$ is a user parameter that controls the amount of local tempo variations we allow in the music piece.

The transition probability of pattern indicator variable $P(r_k \mid r_{k-1}, \phi_k, \phi_{k-1})$ is governed by $\mathbb{A}$, a $R \times R$ time-homogeneous transition matrix where $\mathbb{A}(i, j)$ is the transition probability from r_i to r_j. However, since the rhythmic patterns are one bar (cycle) in length, pattern transitions are allowed only at the end of the bar ($\phi_k < \phi_{k-1}$).

The observation model is identical to the one used in [12], and depends only on the bar position and rhythmic pattern variables, without any influence from tempo. To model rhythm patterns, we compute spectral flux feature from audio in two frequency bands (Low: ≤ 250 Hz, High: > 250 Hz). Using beat and downbeat annotated training data, the audio features are grouped into bar length patterns on a bar discretized into 64th note cells. A k-means clustering algorithm then assigns each bar of the dataset to one of the R rhythmic patterns. All the features within the cell of each pattern are collected and maximum likelihood estimates of the parameters of a two component Gaussian Mixture Model (GMM) are obtained. The observation probability within a 64th note cell is assumed to be constant and is computed as,

$$P(\mathbf{y} \mid \mathbf{x}) = P(\mathbf{y} \mid \phi, r) = \sum_{i=1}^{2} \pi_{\phi,r,i} \mathcal{N}(\mathbf{y}; \boldsymbol{\mu}_{\phi,r,i}, \boldsymbol{\Sigma}_{\phi,r,i})$$

where, $\mathcal{N}(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes a normal distribution of the two dimensional feature $\mathbf{y}$. For the mixture component i, $\pi_{\phi,r,i}$, $\boldsymbol{\mu}_{\phi,r,i}$ and $\boldsymbol{\Sigma}_{\phi,r,i}$ are the component weight, mean (2-dim.) and the covariance matrix (2×2), respectively.

3.3 Inference in BP-model

The goal of inference with the BP-model is to estimate a hidden variable sequence $\mathbf{x}^*_{1:K}$ that maximizes the posterior probability $P(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K})$ given an observed sequence of features $\mathbf{y}_{1:K}$. The sequence $\mathbf{x}^*_{1:K}$ can then be translated into a sequence of downbeat (sama) instants ($t^*_k \mid \phi^*_k = 0$), beat instants ($t^*_k \mid \phi^*_k = i \cdot {}^{M_{r^*}}/_{B_{r^*}}, i = 1, \ldots, B_r$), local instantaneous tempo ($\dot{\phi}^*_k$) and rhythmic patterns (r^*).

In this paper, we use an approximate particle filter [6] based inference scheme called the Auxiliary Mixture Particle Filter (AMPF), which has been shown to be effective for meter analysis [12]. In a particle filter, the posterior is estimated pointwise by approximating it using a weighted set of points (known as particles) in the state space as,

$$P(\mathbf{x}_{1:K} \mid \mathbf{y}_{1:K}) \approx \sum_{i=1}^{N_p} w_K^{(i)} \delta(\mathbf{x}_{1:K} - \mathbf{x}_{1:K}^{(i)}) \qquad (4)$$

Here, $\{\mathbf{x}_{1:K}^{(i)}\}$ is a set of points (particles) with associated weights $\{w_K^{(i)}\}$, $i = 1, \ldots, N_p$, $\mathbf{x}_{1:K}$ is the set of all state trajectories until frame K, $\delta(x)$ is the Dirac delta function, and N_p is the number of particles. The AMPF algorithm includes several enhancements to make it suitable for inference with the BP-model, a detailed description of which has been presented in [12].

3.4 BP-model and AMPF for Informed Meter Analysis

The AMPF algorithm on the BP-model is generic and can be adapted to be applicable to the informed meter analysis tasks described in Section 2. For meter inference, the rhythm class (tāḷa) can be estimated by allowing rhythmic patterns of different lengths from different rhythm classes to be present in the model, as used by [12]. For meter tracking tasks, we assume that the rhythm class is known and all rhythm patterns belong to that class, i.e. $M_r = M$ and $B_r = B \, \forall \, r$.

The initial state distribution $P(\mathbf{x}_0)$ and the initialization of the particle filter system are modified to suit the informed meter tracking tasks. A uniform initialization over all allowed states is used for `Inference` and `Track` tasks, while a narrower informed initialization is done for informed meter tracking. For `TI-Track` task, we use the median ground truth tempo of the music piece being tracked and initialize the tempo variable $\dot{\phi}$ within a tight bound allowing for 10% variation in tempo around the median value. This enables the tracking algorithm to restrict the tempo variable within the tight tempo range and track the correct tempo at the right metrical level. For `SI-Track` task, the provided sama instance is used to initialize the bar position variable ϕ to zero at the related time position. For `STI-Track` task, both the tempo and bar position variables are initialized appropriately using the given information. The tracking algorithm hence gets the tempo and the beginning of the cycle in the piece, tracking the remaining beats and downbeats.

4. EXPERIMENTS

The experiments aim to compare performance across different informed meter analysis tasks and investigate the advantage of the additional prior information they utilize. While the focus of experiments is on Indian music, we also report the results on a collection of Ballroom dances to evaluate the extensibility of the informed analysis tasks. Furthermore, reproducibility will be ensured by providing free access for research purposes to all code repositories and datasets on the companion webpage, which also provides additional resources and music examples. [1]

Dataset	#Pieces	#Ann.	#Sama
CMR	118	28725	5560
HMR$_s$	92	32731	2572
HMR$_l$	59	3280	304
Total (IAM)	269	64736	8436

Table 1: The Carnatic (CMR) and Hindustani (HMR$_l$and HMR$_s$) music datasets showing the number of pieces, sama and beat/mātrā annotations.

4.1 Music Datasets

For the experiments, we use rhythm annotated datasets of Carnatic and Hindustani music (described in Table 1) that have been previously used for evaluating automatic meter analysis algorithms. The Carnatic music rhythm dataset (CMR dataset) [19] includes 118 two minute long excerpts of Carnatic music sampled from commercial releases. The recordings span four commonly used tāḷas with different number of beats in a cycle, with a total duration of 236 minutes. The dataset consists of audio, manually annotated time-aligned markers indicating the progression through the tāḷa cycle, and the associated tāḷa related metadata.

The Hindustani music rhythm dataset consists of 151 two minute long excerpts of Hindustani music sampled from the CompMusic Hindustani music research corpus [21], a curated collection of commercial audio releases and metadata. The excerpts span four popular tāls of Hindustani music that are structurally different and of different lengths. For each audio excerpt, the annotations consist of editorial metadata about the tāl, as well as time-aligned metrical annotations of all beat and sam instances.

The dataset consists of excerpts with a wide tempo range from 10 MPM (mātrās per minute) to 370 MPM. Hindustani music divides tempo into three main tempo classes (lay). Since no exact tempo ranges are defined for these classes, we determined suitable ranges in correspondence with a professional Hindustani musician as 10-60 MPM, 60-150 MPM, and >150 MPM for the slow (vilaṁbit), medium (madhya), and fast (dr̥t) tempi, respectively. The tempo class of a piece has a significant effect on meter analysis due to the wide range of possible tempi. To study any effects of the tempo class, the full Hindustani dataset is divided into two other subsets - the long cycle duration subset called the HMR$_l$ dataset consisting of vilaṁbit pieces and the short cycle duration subset HMR$_s$ dataset with madhya and the dr̥t lay pieces. The complete collection of Carnatic and Hindustani music datasets together is called IAM dataset.

[1] `http://compmusic.upf.edu/informed-meter-tracking`

In addition to Indian art music, we evaluate the tasks on a set of Ballroom dances, which includes beat and bar annotations of audio recordings of several dance styles sourced from `BallroomDancers.com` [9, 11]. The ballroom dataset contains eight different dance styles (Cha cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz, and (slow) Waltz) and has been widely used for several MIR tasks such as genre classification, tempo tracking, beat and downbeat tracking [1, 9, 12]. It consists of 698 thirty second long audio excerpts and has tempo and dance style annotations. The dataset contains two different meters (3/4 and 4/4) and all pieces have constant meter.

4.2 Evaluation Measures

We evaluate the tasks through the relevant meter components they estimate - meter type, tempo, beats and downbeats. We evaluate only the applicable components that are not assumed to be known *a priori* in an informed task (e.g. meter type is known in `Track` task and hence only tempo, beats and downbeats are evaluated).

A variety of measures are available for evaluating beat and downbeat tracking [4]. We chose the f-measure ($\mathfrak{f}$) metric that is widely used in beat tracking evaluation. Other measures were applied in addition during the experiments, but did not add further detail and hence are not reported. It is a number between 0 and 1 computed from estimated and ground truth annotation sequences as the harmonic mean of the precision and recall measures. The definition extends to tracking both the beat/mātrās ($\mathfrak{f}_b$) and the downbeats/samas ($\mathfrak{f}_s$). For `Inference` and `Track` tasks, we additionally report the results of median tempo estimation, comparing the median estimated tempo and the median annotated ground truth tempo with a 5% error margin. For `Inference` task, the algorithms also detect the rhythm class (or tāḷa) and hence the accuracy of this detection is also reported.

4.3 Experimental Setup

Experiments are done separately on each of the three IAM datasets (CMR, HMR_s, HMR_l) and the Ballroom dataset. To compute the f-measure in CMR, HMR_s and Ballroom datasets, an error tolerance window of 70 ms is used between the annotation and the estimated beat/sama. The computation of f-measure with HMR_l dataset is an exception, where a bigger margin window is allowed. Since cycles are of long duration in HMR_l dataset and current evaluation approaches were not designed with such long cycles in mind, an error tolerance window of 70 ms is very tight. To account for the length of the cycle in the error margin, a 6.25% median inter annotation interval is used as the tolerance window, as used in many other beat tracking evaluations (e.g. by [10]). This choice of a larger allowance window also corroborates well with the observation that in vilaṁbit pieces of the HMR_l dataset, there can be significant freedom in pulsation and that larger timing deviations go unnoticed since the pieces are not rhythmically dense. It can be argued that the beat pulsation in vilaṁbit pieces is beyond the duration of what is called the perceptual present [2], and can therefore not be considered to belong to metrical structure. However, it is to be noted that the allowance used in this paper is a compromise and better evaluation measures that can handle these complexities are to be developed.

Dataset	CMR	HMR_s	HMR_l	IAM	Ballroom
Accuracy	68	63	27	57	89

Table 2: Tāḷa recognition accuracy (%) in `Inference` task. Time signature recognition accuracy is reported for Ballroom dataset.

The tempo ranges for initialization of AMPF in `Inference`, `Track` and `SI-Track` tasks are learned from training data of each fold and an additional 20% margin is added to extend to unseen data. However, if the learned ranges are beyond the minimum and maximum tempo limits of each music culture, we set it to the minimum or the maximum. We use one rhythmic pattern per tāḷa (or dance style). Hence, we use $R = 1$ for meter tracking, when a known meter is being tracked, while $R = 4$ (8 in Ballroom dataset) is used for meter inference, with one pattern per tāḷa/rhythm. We use the number of bar positions, $M_r = 1600$ for the longest rhythmic pattern we encounter in the dataset and scale all other pattern lengths accordingly. For the AMPF algorithm, we use 1500 particles per rhythm pattern, with other parameters identical to those used in [12]. A hop size of 20 ms is used to compute the two dimensional spectral flux feature.

4.4 Results and Discussion

The results in Table 2 and Figures 2-3 summarize the performance across different datasets and informed analysis tasks. All results are reported as the mean performance over three runs in a 2-fold (equal size) cross validation experiment on each dataset. The results are presented for each dataset as an average over the pieces in all the tāḷas (or rhythm classes). Table 2 shows the tāḷa recognition accuracy for the Indian music datasets (and time signature estimation accuracy for Ballroom dataset) from the `Inference` task. Figure 3 shows the median tempo estimation accuracy for different datasets in the `Inference`, `Track`, and `SI-Track` tasks, where median tempo is not known *a priori*. The beat and downbeat f-measure values are reported for all the informed analysis tasks in Figure 2. We use a paired-sample t-test to assess statistically significant differences in beat and downbeat tracking performance by pooling the results of Indian music datasets.

Table 2 shows a similar performance with the CMR and HMR_s datasets, but is significantly poor for the long cycle subset of Hindustani music (HMR_l dataset). Whereas in the Carnatic and Hindustani music datasets, each tāḷa has a distinct length, the eight rhythm classes in the Ballroom data are assigned to only two time signatures reducing the task to a classification task between 3/4 and 4/4 time signatures. Ballroom dataset hence shows the best recognition performance.

Tāḷa recognition accuracy affects tempo estimation, as seen in Figure 3 with a poor tempo estimation performance within the HMR_l dataset. Median tempo estimation accuracy is similar for CMR and HMR_s datasets. Tempo es-

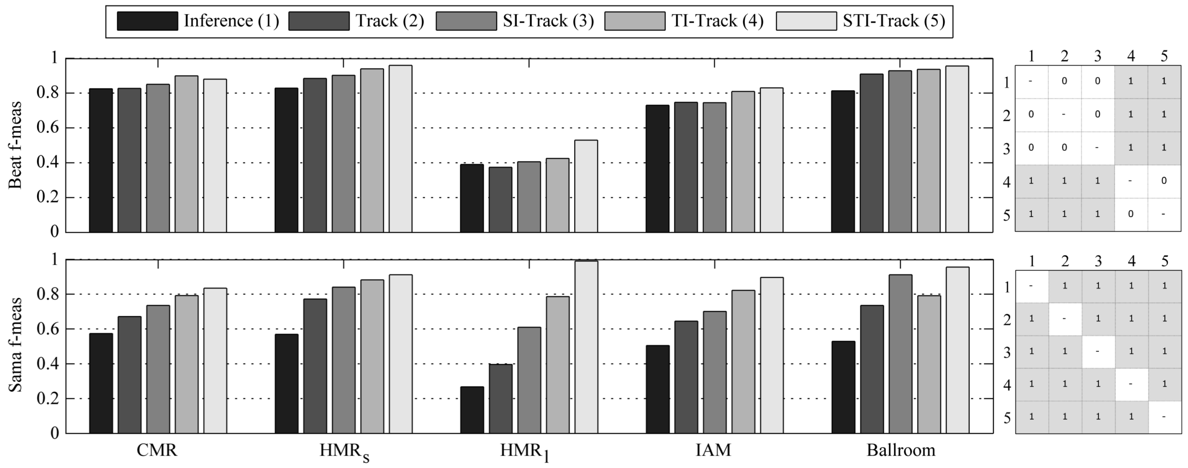


Figure 2: Beat and sama (downbeat) tracking results showing the f-measure as bar plots for different datasets and informed analysis tasks. The matrix on the right shows the results of a significance test between analysis tasks (numbers 1-5 correspond to tasks in the legend) for the IAM dataset. A box with numeral 1 indicates a statistically significant difference in a paired-sample t-test (at $p = 0.05$) while numeral 0 indicates a difference that is not statistically significant.

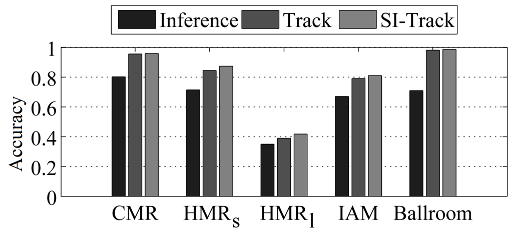


Figure 3: Median tempo estimation accuracy in the `Inference`, `Track` and `SI-Track` tasks.

timation accuracy improves for `Track` task compared to `Inference` task, showing the utility of knowing the meter type in estimating the correct tempo. However, additional downbeat information in `SI-Track` task does not add much to tempo estimation, with marginal or no further improvement. Ballroom dataset shows the best tempo estimation performance except for `Inference` task, where wrong estimations of the rhythm class leads to poorer tempo estimation.

The beat f-measure ($\mathfrak{f}_b$) results in Figure 2 across different informed analysis tasks shows a marginal improvement with informed tracking tasks, but statistically significant improvements are observed only with `TI-Track` and `STI-Track` tasks for IAM datasets, when median tempo is known *a priori*. This shows that the tempo information is more relevant than tāḷa and sama information to improve beat tracking performance for Indian art music. The biggest gains in informed meter analysis are seen in sama f-measure ($\mathfrak{f}_s$), with significant improvements achieved with more informed analysis tasks. For the pooled IAM dataset, starting with a $\mathfrak{f}_s = 0.51$ with `Inference` task, `STI-Track` task achieves $\mathfrak{f}_s = 0.82$, showing the benefit and the utility of both tempo and sama information in informed meter analysis for a more difficult task of downbeat estimation.

For Ballroom dataset, compared to the `Track` task, we observe that downbeat tracking performance for `SI-Track` improves more over `TI-Track` task. This indicates that downbeat information is more important than tempo information. It is perhaps due to the fact that Ballroom dances have a stable tempo and clear repeated rhythmic patterns. Accurate tempo estimation is achieved even without prior tempo information (Figure 3), and hence downbeat information is more useful.

A comparison of performance across datasets shows that CMR, HMR$_s$ and Ballroom datasets have similar trends of improvement in both beat and sama (downbeat) tracking with informed tracking tasks. The largest gains however are obtained with the long cycle HMR$_l$ dataset, which improves from a poor $\mathfrak{f}_s = 0.26$ (`Inference`) to $\mathfrak{f}_s = 0.99$ (`STI-Track`). While we note that a larger error margin and fewer sama examples in the long cycle dataset contribute to this high performance, the overall results considering all datasets and tasks conclude that the use of tempo and sama information enhances the capabilities of automatic meter analysis algorithms to track downbeats.

5. CONCLUSIONS

Starting with a hypothesis that automatic meter analysis performance can be improved by utilizing additional information about meter or tempo of a piece, we formulated relevant informed meter analysis tasks that can incorporate varying levels of prior information about the meter type, tempo and downbeat position. An evaluation on corpora of Indian art music and Ballroom dances showed the utility of prior information for automatic meter analysis, where tempo information is useful for beat tracking and the tempo and downbeat information was shown to be useful for downbeat tracking. We also showed that with minimal effort by a potential user of an annotation system, a high accuracy in tempo, beat and downbeat estimation can be achieved through informed meter analysis algorithms. Evaluation of informed analysis tasks in the paper was done through individual components of meter (tempo, beat, downbeat). In future work, we plan to develop unified meter analysis evaluation measures that take into account the hierarchical structure of musical meter.

6. ACKNOWLEDGMENTS

This work is partly supported by the European Research Council as part of the CompMusic project (ERC grant agreement 267583). Ajay Srinivasamurthy is currently with Idiap Research Institute, Martigny, Switzerland.

7. REFERENCES

[1] S. Böck, F. Krebs, and G. Widmer. A Multi-model Approach to Beat Tracking Considering Heterogenous Music Styles. In *Proc. of the 15th Intl. Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 602–607, Taipei, Taiwan, October 2014.

[2] E. Clarke. Rhythm and timing in music. In Diana Deutsch, editor, *The Psychology of Music*, pages 473–500. Academic Press, San Diego, II edition, 1999.

[3] M. Clayton. *Time in Indian Music : Rhythm, Metre and Form in North Indian Rag Performance*. Oxford University Press, 2000.

[4] M. E. P. Davies, N. Degara, and M. D. Plumbley. Evaluation Methods for Musical Audio Beat Tracking Algorithms. *Technical Report C4DM-TR-09-06, Queen Mary University of London*, October 2009.

[5] M. E. P. Davies and M. D. Plumbley. Context-Dependent Beat Tracking of Musical Audio. *IEEE Trans. on Audio, Speech, and Language Processing*, 15(3):1009–1020, March 2007.

[6] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 2009.

[7] S. Durand, J. P. Bello, B. David, and G. Richard. Downbeat tracking with multiple features and deep neural networks. In *Proc. of the 40th IEEE Intl. Conference on Acoustics, Speech, and Signal Processing (ICASSP 2015)*, Brisbane, Australia, May 2015.

[8] D. Ellis. Beat Tracking by Dynamic Programming. *Journal of New Music Research*, 36(1):51–60, 2007.

[9] 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*, 14(5):1832–1844, 2006.

[10] J. Hockman, M. E. P. Davies, and I. Fujinaga. One in the Jungle: Downbeat Detection in Hardcore, Jungle, and Drum and Bass. In *Proc. of the 13th Intl. Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 169–174, Porto, Portugal, October 2012.

[11] F. Krebs, S. Böck, and G. Widmer. Rhythmic Pattern Modeling for Beat- and Downbeat Tracking in Musical Audio. In *Proc. of the 14th Intl. Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 227–232, Curitiba, Brazil, November 2013.

[12] F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer. Inferring Metrical Structure in Music Using Particle Filters. *IEEE/ACM Trans. on Audio, Speech, and Language Processing*, 23(5):817–827, May 2015.

[13] M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri. Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms. *Journal of New Music Research*, 36(1):1–16, 2007.

[14] G. Peeters and H. Papadopoulos. Simultaneous Beat and Downbeat-Tracking Using a Probabilistic Framework: Theory and Large-Scale Evaluation. *IEEE Trans. on Audio, Speech and Language Processing*, 19(6):1754–1769, 2011.

[15] A. Pikrakis, I. Antonopoulos, and S. Theodoridis. Music meter and tempo tracking from raw polyphonic audio. In *Proc. of the 5th Intl. Conference on Music Information Retrieval (ISMIR 2004)*, Barcelona, Spain, October 2004.

[16] P. Sambamoorthy. *South Indian Music Vol. I-VI*. The Indian Music Publishing House, 1998.

[17] X. Serra. Creating Research Corpora for the Computational Study of Music: the case of the CompMusic Project. In *Proc. of the 53rd AES Intl. Conference on Semantic Audio*, London, January 2014.

[18] A. Srinivasamurthy. *A Data-driven Bayesian Approach to Automatic Rhythm Analysis of Indian Art Music*. Doctoral dissertation, Universitat Pompeu Fabra, Barcelona, Spain, 2016.

[19] A. Srinivasamurthy, A. Holzapfel, Ali Taylan Cemgil, and X. Serra. Particle Filters for Efficient Meter Tracking with Dynamic Bayesian Networks. In *Proc. of the 16th Intl. Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 197–203, Malaga, Spain, October 2015.

[20] A. Srinivasamurthy, A. Holzapfel, and X. Serra. In Search of Automatic Rhythm Analysis Methods for Turkish and Indian Art Music. *Journal of New Music Research*, 43(1):97–117, 2014.

[21] A. Srinivasamurthy, G. K. Koduri, S. Gulati, V. Ishwar, and X. Serra. Corpora for Music Information Research in Indian Art Music. In *Proc. of 13th Sound and Music Computing Conference*, pages 1029–1036, Athens, Greece, September 2014.

[22] A. Srinivasamurthy and X. Serra. A Supervised Approach to Hierarchical Metrical Cycle Tracking from Audio Music Recordings. In *Proc. of the 39th IEEE Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, pages 5237–5241, Florence, Italy, May 2014.

[23] N. Whiteley, A. T. Cemgil, and S. Godsill. Sequential Inference of Rhythmic Structure in Musical Audio. In *Proc. of the IEEE Intl. Conference on Acoustics, Speech and Signal Processing (ICASSP 2007)*, volume 4, pages 1321–1325, Honolulu, USA, April 2007.

INTELLIGIBILITY OF SUNG LYRICS: A PILOT STUDY

Karim M. Ibrahim[1] **David Grunberg**[1] **Kat Agres**[2]
Chitralekha Gupta[1] **Ye Wang**[1]
[1] Department of Computer Science, National University of Singapore, Singapore
[2] Institute of High Performance Computing, A*STAR, Singapore
karim.ibrahim@comp.nus.edu.sg, wangye@comp.nus.edu.sg

ABSTRACT

We propose a system to automatically assess the intelligibility of sung lyrics. We are particularly interested in being able to identify songs which are intelligible to second language learners, as such individuals often sing along the song to help them learn their second language, but this is only helpful if the song is intelligible enough for them to understand. As no automatic system for identifying 'intelligible' songs currently exists, songs for second language learners are generally selected by hand, a time-consuming and onerous process. We conducted an experiment in which test subjects, all of whom are learning English as a second language, were presented with 100 excerpts of songs drawn from five different genres. The test subjects listened to and transcribed the excerpts and the intelligibility of each excerpt was assessed based on average transcription accuracy across subjects. Excerpts that were more accurately transcribed on average were considered to be more intelligible than those less accurately transcribed on average. We then tested standard acoustic features to determine which were most strongly correlated with intelligibility. Our final system classifies the intelligibility of the excerpts and achieves 66% accuracy for 3 classes of intelligibility.

1. INTRODUCTION

While various studies have been conducted on singing voice analysis, one aspect which has not been well-studied is the *intelligibility* of a given set of lyrics. Intelligibility describes how easily a listener can comprehend the words that a performer sings; the lyrics of very intelligible songs can easily be understood, while the lyrics of less intelligible songs sound garbled or even incomprehensible to the average listener. People's impressions of many songs are strongly influenced by how intelligible the lyrics are, with one study even finding that certain songs were perceived as 'happy' when people could not understand its lyrics, but was perceived as 'sad' when the downbeat lyrics were made comprehensible [20]. It would thus be useful to enable systems to automatically determine intelligibility, as it is a key factor in people's perception of a wide variety of songs.

We are particularly interested in measuring the intelligibility of songs with respect to second language learners. Many aspects of learning a second language to the point of fluency have been shown to be difficult, including separating the phonemes of an unfamiliar language [30], memorizing a large number of vocabulary words and grammar rules [22], and maintaining motivation for the length of time required to learn the language. Consequently, many second language learners need help, and music has been shown to be a useful tool for this purpose. Singing and language development have been shown to be closely related at the neurological level [24, 32], and experimental results have demonstrated that singing along with music in the second language is an effective way of improving memorization and pronunciation [12, 19]. However, specific songs are only likely to help these students if they can understand the content of the lyrics [11]. As second language learners may have difficulty understanding certain songs in their second language due to their lack of fluency, they could be helped by a system capable of automatically determining which songs they are likely to find intelligible or unintelligible.

We therefore seek to design a system which is capable of assessing a given song and assigning it an intelligibility score, with the standard of intelligibility biased towards people who are learning the language of the lyrics but have not yet mastered it. To gather data for this system we compiled excerpts from 50 songs and had volunteering participants listen to the song in order to discover how intelligible they found the lyrics. Rather than simply having the participants rate the intelligibility of the song, we had the participants transcribe the lyrics that they heard and then calculated an intelligibility score for each excerpt based on the statistics of how accurately the students transcribed it. Excerpts that were transcribed more accurately on average were judged to be more intelligible than those transcribed less accurately on average. A variety of acoustic features were then used to build a classifier which could determine the intelligibility of a given piece of music. The classifier was then run on the same excerpts used in the listening experiment, and the results of each were compared.

The remaining outline of this paper is as follows: Section 2 lists relevant literature in the field. Section 3 describes the transcription experiment performed to gather data. Section 4 discusses the features and the classifier. Finally, Sections 5 and 6 shows the evaluation of our proposed model and our conclusions, respectively.

2. LITERATURE REVIEW

That sung lyrics could be more difficult to comprehend than spoken words has long been established in the scientific community. One study showed that even professional voice teachers and phoneticians had difficulty telling vowels apart when sung at high pitch [7]. Seminal work by Collister and Huron found listeners to make hearing errors as much as seven times more frequently when listening to sung lyrics than spoken ones [3]. Such studies also noted lyric features which could help differentiate intelligible from unintelligible songs; for instance, one study noted that songs comprised mostly of common words sounded more intelligible than songs with rarer words [9]. However, lyric features alone are not sufficient to assess intelligibility; the same lyrics can be rendered more or less intelligible depending on, for instance, the speed at which they are sung. These other factors must be taken into account to truly assess lyric intelligibility.

Studies have been conducted on assessing the overall *quality* of singing voice. One acoustic feature which multiple studies have found to be useful for this purpose is the power ratio of frequency bands containing energy from the singing voice to other frequency bands; algorithms using this feature have been shown to reliably distinguish between trained and untrained singers [2,23,34]. Calculation of pitch intervals and vibrato have also been shown to be useful for this purpose [21]. However, while the quality of singing voice may be a factor in assessing intelligibility, it is not the only such factor. Aspects of the song that have nothing to do with the skill of the singer or the quality of their performance, such as the presence of loud background instruments, can contribute, and additional features that take these factors into account are needed for a system which determines lyric intelligibility.

Another related task is that of singing transcription, in which a computer must listen to and transcribe sung lyrics [18]. It may seem that one could assess intelligibility by comparing a computer's transcription of the lyrics to a ground truth set of lyrics and determining if the transcription is accurate. But this too does not really determine intelligibility, at least as humans perceive it. A computer can use various filters and other signal processing or machine learning tools to process the audio and make it easier to understand, but a human listening to the music will not necessarily have access to such tools. Thus, even if a computer can understand or accurately transcribe the lyrics of a piece of music, this does not indicate whether those lyrics would be intelligible to a human as well.

3. BEHAVIORAL EXPERIMENT

To build a system that can automatically process a song and evaluate the intelligibility of its lyrics, it is essential to gather ground truth data that reflects this intelligibility on average across different listeners. Hence, we conducted a study where participants were tasked with listening to short excerpts of music and transcribing the lyrics, a common task for evaluating intelligibility of lyrics [4]. The accuracy of their transcription can be used to assess the intelligibility of each excerpt.

3.1 Method

3.1.1 Participants

Seventeen participants (seven females and ten males) volunteered to take part in the experiment. Participants were between 21 to 41 years (mean = 27.4 years). All participants indicated no history of hearing impairment and that they spoke some English as a second language. Participants were rewarded with a $10 voucher for their time. Participants were recruited through university channels via posters and fliers. The majority of the participants were university students.

3.1.2 Materials

For the purpose of this study, we focused solely on English-language songs. Because one of the main applications for such a system is to recommend music for students who are learning foreign languages, we focused on genres that are popular for students. To identify these genres, we asked 48 university students to choose the 3 genres that they listen to the most, out of the 12 genres introduced in [4], as these 12 genres cover a wide variety of singing styles. The twelve genres are: Avante-garde, Blues, Classical, Country, Folk, Jazz, Pop/Rock, Rhythm and Blues, Rap, Reggae, Religious, and Theater. Because the transcription task is long and tiring for participants, we limited the number of genres tested to only five, from which we would draw approximately 45 minutes worth of music for transcription. We selected the five most popular genres indicated by the 48 participants: Classical, Folk, Jazz, Pop/Rock, and Rhythm and Blues.

After selecting the genres, we collected a dataset of 10 songs per genre. Because we were interested in evaluating participants' ability to transcribe an unfamiliar song, as opposed to transcribing a known song from memory, we focused on selecting songs that are not well-known in each genre. We approached this by selecting songs that have less than 200 ratings on the website Rate Your Music (rateyourmusic.com). Rate Your Music is a database of popular music where users can rate and review different songs, albums and artists. Popular songs have thousands of ratings while less known songs have few ratings. We used this criteria to collect songs spanning the 5 genres to produce our dataset. The songs were randomly selected, with no control over the vocal range or the singer's accent, as long as they satisfied the condition of being in English and having few ratings.

Because transcribing an entire song, let alone 50 songs, would be an overwhelming process for the participants, we selected short excerpts from each song to be transcribed. Two excerpts per song were selected randomly such that each excerpt would include a complete utterance (e.g., no excerpts were terminated mid-phrase). Excerpts varied between 3 to 16 seconds in length (average = 6.5 seconds), and contained 9.5 words on average. The ground-truth lyrics for these songs were collected from online sources and reviewed by the experimenters to ensure they matched the version of the song used in the experiment. It is important to note that selecting short excerpts might affect intelligibility, because the context of the song (which may help in understanding the lyrics) is lost. However, using these short excerpts is essential in making the experiment feasible for the participants, and would still broadly reflect the intelligibility of the song. The complete dataset is composed of 100 excerpts from 50 songs, 2 excerpts per song, covering 5 genres, and 10 songs per genre. Readers who are interested in experimenting on the dataset can contact the authors.

3.1.3 Procedure

We conducted the experiment in three group listening sessions. During each session, the participants were seated in a computer lab, and recorded their transcriptions of the played excerpts on the computer in front of them. The excerpts were played in randomized order, and each excerpt was played twice consecutively. Between the two playbacks of each excerpt there was a pause of 5 seconds, and between different excerpts a pause of 10 seconds, to allow the participants sufficient time to write their transcription. The total duration of the listening session is 46:59 minutes. Two practice trials were presented before the experimental trials began, to familiarize participants with the experimental procedure.

3.2 Results and Discussion

To evaluate the accuracy of the participants' transcription, we counted the number of words correctly transcribed by the participant that match the ground truth lyrics. For each transcription by each student, the ratio between correctly transcribed words to the total number of words in the excerpt was calculated. We then calculated the average ratio for each excerpt across all 17 participants to yield an overall score for each excerpt between 0 and 1. This score was used to represent the ground-truth transcription accuracy, or *Intelligibility score*, for each excerpt. The distribution of Intelligibility scores in the dataset is shown in Figure 1. From the figure, we can observe that the intelligibility scores are biased towards higher values, i.e. there are relatively few excerpts with a low intelligibility score. This may be caused by the restricted set of popular genres indicated by students, as certain excluded genres would be expected to have low intelligibility, such as Heavy Metal.

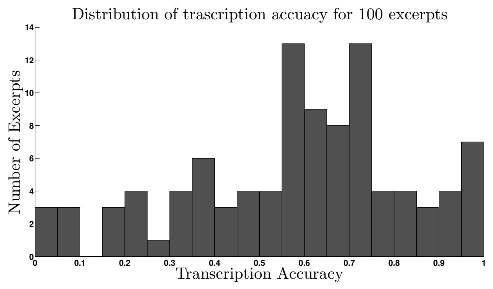


Figure 1. The distribution of the transcription accuracies (Intelligibility score).

4. COMPUTATIONAL SYSTEM

The purpose of this study is to select audio features that can be used to build a system capable of 1) predicting the intelligibility of song lyrics, and 2) evaluating the accuracy of these predictions with respect to the ground truth gathered from human participants. In the following approach, we analyze the input signal and extract expressive features that reflect the different aspects of an intelligible singing voice. Several properties may contribute to making the singing voice less intelligible than normal speech. One such aspect is the presence of background music, as accompanying music can cover or obscure the voice. Therefore, highly intelligible songs would be expected to have a dominant singing voice compared with the accompanying music [4]. Unlike speech, the singing voice has a wider and more dynamic pitch range, often featuring higher pitches in soprano vocal range. This has been shown to affect the intelligibility of the songs, especially with respect to the perception of sung vowels [1, 3]. An additional consideration is that in certain genres, such as Rap, singing is faster and has a higher rate of words per minute than speech, which can reduce intelligibility. Furthermore, as indicated in [10], the presence of common, frequently occurring words helps increase intelligibility, while uncommon words decrease the likelihood of understanding the lyrics. In our model, we aimed to include features that express these different aspects to determine the intelligibility of song lyrics across different genres. These features are then used to train the model to accurately predict the intelligibility of lyrics in the dataset, based on the ground truth collected in our behavioral experiment.

4.1 Preprocessing

To extract the proposed features from an input song, two initial steps are required: separating the singing voice from the accompaniment, and detecting the segments with vocals. To address these steps, we selected the following approaches based on current state-of-the-art methods:

4.1.1 Vocals Separation

Separating vocals from accompaniment music is a well-known problem that has received considerable attention in the research community. Our approach makes use of the popular Adaptive REPET algorithm [16]. This algorithm is based on detecting the repeating patten in the song, which is meant to represent the background music. Separating the detected pattern leaves the non-repeating part of the song, meant to capture the vocals. Adaptive REPET also has the advantage of discovering local repeating patterns in the song over the original REPET algorithm [26]. Choosing Adaptive REPET was based on two main advantages: The algorithm is computationally attractive, and it shows competitive results compared to other separation algorithms, as shown in the evaluation of [14].

4.1.2 Detecting Vocal Segments

Detecting vocal and non-vocal segments in the song is an important step in extracting additional information about the intelligibility of the lyrics. Various approaches have been proposed to perform accurate vocal segmentation, however, it remains a challenging problem. For our approach, we implemented a method based on extracting the features proposed in [15], then training a Random Forest classifier using the Jamendo corpus [1] [27]. The classifier was then used to binary classify each frame of the input file as either vocals or non-vocals.

4.2 Audio features

In this section, we investigate the set of features we used in training the model for estimating lyrics intelligibility. We use a mix of features reflecting specific aspects of intelligibility plus common standard acoustic features. The selected features are:

1. **Vocals to Accompaniment Music Ratio (VAR)**: Defined as the energy of the separated vocals divided by the energy of the accompaniment music. This ratio is computed only in segments where vocals are present. This feature reflects how strong the vocals are compared to the accompaniment. High VAR suggests that vocals are relatively loud and less likely to be obscured by the music. Hence, higher VAR counts for higher intelligibility. This feature is particularly useful in identifying songs that are unintelligible due to loud background music which obscures the vocals.

2. **Harmonics-to-residual Ratio (HRR)**: Defined as the the energy in a detected fundamental frequency (f0) according to the YIN algorithm [5] plus the energy in its 20 first harmonics (a number chosen based on empirical trials), all divided by the energy of the residual. This ratio is also applied only to segments where vocals are present. Since harmonics of the detected f0 in vocal segments are expected to be produced by the singing voice, this ratio, like VAR, helps to determine whether the vocals in a given piece of music are stronger or weaker than the background music which might obscure it.

[1] http://www.mathieuramona.com/wp/data/jamendo/

3. **High Frequency Energy (HFE)**: Defined as the sum of the spectral magnitude above 4kHz,

$$HFE_n = \sum_{k=f_{4k}}^{N_b/2} a_{n,k} \tag{1}$$

where $a_{n,k}$ is the magnitude of block n and FFT index k of the short time Fourier transform of the input signal, f_{4k} is the index corresponding to 4 kHz and N_b is the FFT size [8]. We calculate the mean across all frames of the separated and segmented vocals signal, as we are interested in the high energy component in vocals and not the accompanying instruments. We get a scalar value per input file reflecting high frequency energy. Singing in higher frequencies has been proven to be less intelligible than music in low frequencies [3], so detection of high frequency energy can be a useful clue that such vocals might be present and could reduce the intelligibility of the music, such as frequently happens with opera music.

4. **High Frequency Component (HFC)**: Defined as the sum of the amplitudes and weighted by the frequency squared,

$$HFC_n = \sum_{k=1}^{N_b/2} k^2 a_{n,k} \tag{2}$$

where $a_{n,k}$ is the magnitude of block n and FFT index k of the short time Fourier transform of the input signal and N_b is the FFT size [17]. This is another measure of high frequency content.

5. **Syllable Rate**: Singing at a fast pace while pronouncing several syllables over a short period of time can negatively affect the intelligibility [6]. In the past, Rao et al. used temporal dynamics of timbral features to separate singing voice from background music [28]. These features showed more variance over time for singing voice, while being relatively invariant to background instruments. We expect that these features will also be sensitive to the syllable rate in singing. We use the temporal standard deviation of two of their timbral features: sub-band energy (SE) in the range of ([300-900 Hz]), and sub-band spectral centroid (SSC) in the range of ([1.2-4.5 kHz]), defined as

$$SSC = \frac{\sum_{k=k_{low}}^{k_{high}} f(k)|X(k)|}{\sum_{k=k_{low}}^{k_{high}} |X(k)|} \tag{3}$$

$$SE = \sum_{k=k_{low}}^{k_{high}} |X(k)|^2 \tag{4}$$

where $f(k)$ and $|X(k)|$ are frequency and magnitude spectral value of the k^{th} frequency bin, and k_{low} and k_{high} are the nearest frequency bins to the lower and upper frequency limits on the sub-band respectively.

According to [28], SE enhances the fluctuations between voiced and unvoiced utterances, while SSC enhances the variations in the 2^{nd}, 3^{rd} and 4^{th} formants across phone transitions in the singing voice. Hence, it is reasonable to expect high temporal variance of these features for songs with high syllable rate, and vice versa. Thus, this feature is able to differentiate songs with high and low syllable rates. We would expect that very high and very low syllable rates should lead to low intelligibility score, while rates in a similar range to that of speech should result in high intelligibility score.

6. **Word-Frequency Score**: Songs which use common words have been shown to be more intelligible than those which use unusual or obscure words [10]. Hence, we calculate a word-frequency score for the lyrics of the songs as an additional feature. This feature is a non-acoustic feature that is useful in cases where the lyrics of the song are available. We calculate the word-frequency score using the `wordfreq` open-source toolbox [31] which provides an estimates of the frequencies of words in many languages.

7. **Tempo and Event Density**: These two rhythmic features reflect how fast the beat and rhythm of the song are. Event density is defined as the average frequency of events, i.e., the number of note onsets per second. Songs with very fast beats and high event density are likely to be less intelligible than slower songs, since the listener has less time to process each event before the next one begins. We used the `MIRToolbox` [13] to extract these rhythmic features.

8. **Mel-frequency cepstral coefficients (MFCCs)**: MFCCs approximates the human auditory system's response more closely than the linearly-spaced frequency bands [25]. MFCCs have been proven to be effective features in problems related to singing voice analysis [29], and so were considered as a potential feature here as well. For our system, we selected the 17 first coefficients (excluding the 0th) as well as the deltas of those features, which proved empirically to be the best number of coefficients. The MFCCs are extracted from the original signal without separation, as it reflects how the whole song is perceived.

By extracting this set of features for an input file, we end up with a vector of 43 features to be used in estimating the intelligibility of the lyrics in this song.

4.3 Model training

We used the dataset and ground-truth collected in our behavioral experiment to train a Support Vector Machine model to estimate the intelligibility of the lyrics. To categorize the intelligibility to different levels that would match a language student's fluency level, we divided our dataset to three classes:

`High Intelligibility`: excerpts with transcription accuracy of greater than 0.66.

`Moderate Intelligibility`: excerpts with transcription accuracy between 0.33 and 0.66 inclusive.

`Low Intelligibility`: excerpts with transcription accuracy of less than 0.33.

Out of the 100 samples in our dataset, 43 are in the High Intelligibility class, 42 are in the Moderate Intelligibility class, and the remaining 15 are in the Low Intelligibility class. For this pilot study, we tried a number of common classifiers, including Support Vector Machine (SVM), random forest and k-nearest neighbors. Our trials for finding a suitable model led to using SVM with a linear kernel, as it is an efficient, fast and simple model which is suitable for this problem. Finally, as a preprocessing step, we normalize all the input feature vectors before passing them to the model to be trained.

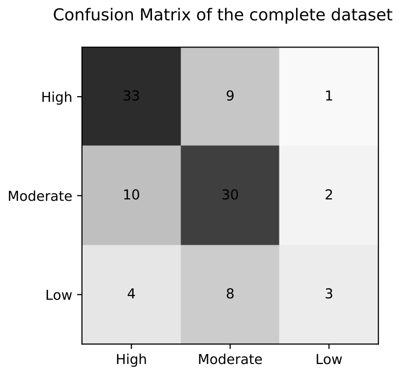


Figure 2. Confusion Matrix of the SVM output.

5. MODEL EVALUATION

Because this problem has not been addressed before in the literature, and it is not possible to perform evaluation using other methods, we based our evaluation on classification accuracy from the dataset. Given the relatively small number of samples in the dataset, we used leave-one-out cross-validation for evaluation. To evaluate the performance of our model, we compute overall accuracy, as well as the Area Under the ROC Curve (AUC). We scored AUC of 0.71 and accuracy of 66% with the aforementioned set of features and model. The confusion matrix of validating our model using leave-one-out cross-validation on our collected dataset is shown in Figure 2. The figure shows that the classifier has relatively more accuracy in predicting high and moderate than low intelligibility, which is often confused with the moderate class. Given that our findings are based on a relatively small segment of excerpts with low intelligibility, the classifier was found to be trained to work better on the high and moderate excerpts.

Following model evaluation on the complete dataset, we were interested in investigating how the model performs on different genres, specifically how it performs when tested

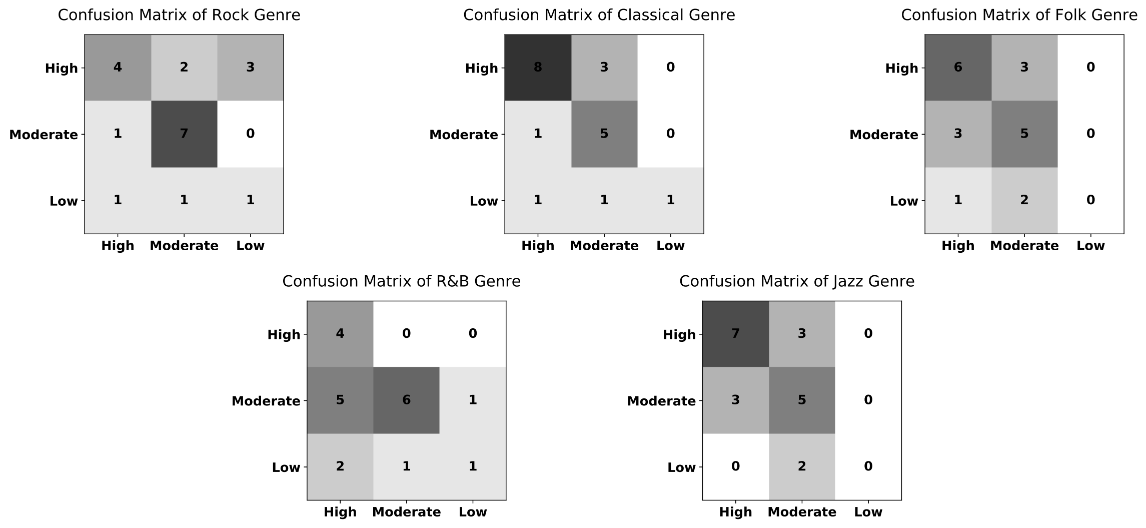


Figure 3. Confusion matrix of the different genres

Genre	Classification Accuracy
Pop/Rock	60%
R&B	55%
Classical	70%
Folk	55%
Jazz	60%

Table 1. Classification accuracy for different genres

with a genre that was not included in the training dataset. This would imply how the model generalizes when running on different genres that was not present during training, as well as showing how changing genres affect classification accuracy. We performed an evaluation where we trained our model using 4 out of the 5 genres in our dataset, and tested it on the 5th genre. The classification accuracy across different genres is shown in Table 1. The results show variance in classifying different genres. For example, Classical music receives higher accuracy, while genres as Rhythm and Blues and Folk shows less accuracy. By analyzing the confusion matrices of each genre shown in Figure 3, we found that the confusion is mainly between high and moderate classes.

By reviewing the impact of the different features on the classifier performance, we looked into what features have the biggest impact using the attribute ranking feature in Weka [35]. We found that several MFCCs contribute most in differentiating between the three classes, which we interpret to be due to analyzing the signal in different frequency sub-bands incorporates perceptual information of both the singing voice and the background music. This was followed by the features reflecting the syllable rate in the song, because singing rate can radically affect the intelligibility. Vocals-to-Accompaniment Ratio and High Frequency Energy followed in their impact on differentiating between the three classes. The features that had the least impact were the tempo and event density, which does not necessarily reflect the rate of singing.

For further studies on the suitability of the features in classifying songs with very low intelligibility, the genres pool can be extended to include other genres with lower intelligibility, rather than being limited to the popular genres between students. Further studies can also include the feature selection and evaluation process: similar to the work in [33], deep learning methods may be explored to select the features which perform best, rather than hand-picking features, to find the most suitable set of features for this problem. It is possible to extend the categorical approach of intelligibility levels to a regression problem, in which the system evaluates the song's intelligibility with a percentage. Similarly, certain ranges of the intelligibility score can be used to recommend songs to students based on their fluency level.

6. CONCLUSION

In this study, we investigated the problem of evaluating the intelligibility of song lyrics to provide an aid for language learners who listen to music as part of language immersion. We conducted a behavioral experiment to review how the intelligibility of lyrics in different genres of songs are perceived by human participants. We then developed a computational system to automatically estimate the intelligibility of lyrics in a given song. In our system, we proposed features to reflect different factors that affect the intelligibility of lyrics according to previous empirical studies. We used the proposed features along with standard audio features to train a model capable of estimating the intelligibility of lyrics (as low, moderate, or high intelligibility) with an AUC of 0.71. The study provides evidence that the proposed system has promising initial results, and draws attention to the problem of lyrics intelligibility, which has received little attention in terms of computational audio analysis and automatic evaluation.

7. REFERENCES

[1] Martha S Benolken and Charles E Swanson. The effect of pitch-related changes on the perception of sung vowels. *The Journal of the Acoustical Society of America*, 87(4):1781–1785, 1990.

[2] Ugo Cesari, Maurizio Iengo, and Pasqualina Apisa. Qualitative and quantitative measurement of the singing voice. *Folia Phoniatrica et Logopaedica*, 64(6):304–309, 2013.

[3] Lauren Collister and David Huron. Comparison of word intelligibility in spoken and sung phrases. *Empirical Musicology Review*, 3(3):109–125, 2–8.

[4] Nathaniel Condit-Schultz and David Huron. Catching the lyrics. *Music Perception: An Interdisciplinary Journal*, 32(5):470–483, 2015.

[5] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[6] Aihong Du, Chundan Lin, and Jingjing Wang. Effect of speech rate for sentences on speech intelligibility. In *Communication Problem-Solving (ICCP), 2014 IEEE International Conference on*, pages 233–236. IEEE, 2014.

[7] Harry Hollien, Ana Mendes-Schwartz, and Kenneth Nielsen. Perceptual confusions of high-pitched sung vowels. *Journal of Voice*, 14(2):287–298, 2000.

[8] Kristoffer Jensen and Tue Haste Andersen. Real-time beat estimationusing feature extraction. In *International Symposium on Computer Music Modeling and Retrieval*, pages 13–22. Springer, 2003.

[9] Randolph Johnson, David Huron, and Lauren Collister. Music and lyrics interaction and their influence on recognition of sung words: an investigation of word frequency, rhyme, metric stress, vocal timbre, melisma, and repetition priming. *Empirical Musicology Review*, 9(1):2–20, 2014.

[10] Randolph B Johnson, David Huron, and Lauren Collister. Music and lyrics interactions and their influence on recognition of sung words: an investigation of word frequency, rhyme, metric stress, vocal timbre, melisma, and repetition priming. *Empirical Musicology Review*, 9(1):2–20, 2013.

[11] Tung-an Kao and Rebecca Oxford. Learning language through music: A strategy for building inspiration and motivation. *System*, 43:114–120, 2014.

[12] Anne Kultti. Singing as language learning activity in multilingual toddler groups in preschool. *Early Child Development and Care*, 183(12):1955–1969, 2013.

[13] Olivier Lartillot and Petri Toiviainen. A matlab toolbox for musical feature extraction from audio. `https://www.jyu.fi/hytk/fi/laitokset/mutku/en/research/materials/mirtoolbox`, 2007.

[14] Bernhard Lehner and Gerhard Widmer. Monaural blind source separation in the context of vocal detection. In *16th International Society for Music Information Retrieval Conference (ISMIR), At Malaga, Spain*, 2015.

[15] Bernhard Lehner, Gerhard Widmer, and Reinhard Sonnleitner. On the reduction of false positives in singing voice detection. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7480–7484. IEEE, 2014.

[16] Antoine Liutkus, Zafar Rafii, Roland Badeau, Bryan Pardo, and Gaël Richard. Adaptive filtering for music/voice separation exploiting the repeating musical structure. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 53–56. IEEE, 2012.

[17] Paul Masri and Andrew Bateman. Improved modelling of attack transients in music analysis-resynthesis. In *ICMC*, 1996.

[18] Annamaria Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010.

[19] Carmen Mora. Foreign language acquisition and melody singing. *ELT journal*, 54(2):146–152, 2000.

[20] Kazuma Mori and Makoto Iwanaga. Pleasure generated by sadness: Effect of sad lyrics on the emotions induced by happy music. *Psychology of Music*, 42(5), 2014.

[21] Tomoyasu Nakano. An automatic singing skill evaluation method for unknown melodies using pitch interval accuracy and vibrato features. In *Proceedings of INTERSPEECH2006*, 2006.

[22] Joan Netten and Claude Germain. A new paradigm for the learning of a second or foreign language: the neurolinguistic approach. *Neuroeducation*, 1(1), 2012.

[23] Koichi Omori, Ashutosh Kacker, Linda Carroll, William Riley, and Stanley Blaugrund. Singing power ratio: quantitative evaluation of singing voice quality. *Journal of Voice*, 10(3):228–235, 1996.

[24] Aniruddh Patel. Language, music, syntax and the brain. *Nature Neuroscience*, 6(7):674–681, 2003.

[25] Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. PTR Prentice Hall, 1993.

[26] Zafar Rafii and Bryan Pardo. Repeating pattern extraction technique (repet): A simple method for music/voice separation. *IEEE transactions on audio, speech, and language processing*, 21(1):73–84, 2013.

[27] Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal detection in music with support vector machines. In *Proc. ICASSP '08*, pages 1885–1888, March 31 - April 4 2008.

[28] Vishweshwara Rao, Chitralekha Gupta, and Preeti Rao. Context-aware features for singing voice detection in polyphonic music. In *International Workshop on Adaptive Multimedia Retrieval*, pages 43–57. Springer, 2011.

[29] Martın Rocamora and Perfecto Herrera. Comparing audio descriptors for singing voice detection in music audio files. In *Brazilian symposium on computer music, 11th. san pablo, brazil*, volume 26, page 27, 2007.

[30] Daniele Schon, Sylvain Moreno, Mireille Besson, Isabelle Peretz, and Regine Kolinsky. Songs as an aid for language acquisition. *Cognition*, 106(2):975–983, 2008.

[31] Robert Speer, Joshua Chin, Andrew Lin, Lance Nathan, and Sara Jewett. wordfreq: v1.5.1. `https://doi.org/10.5281/zenodo.61937`, September 2016.

[32] Valerie Trollinger. The brain in singing and language. *General Music Today*, 23(2), 2010.

[33] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.

[34] Christopher Watts, Kathryn Barnes-Burroughs, Julie Estis, and Debra Blanton. The singing power ratio as an objective measure of singing voice quality in untrained talented and nontalented singers. *Journal of Voice*, 20(1):82–88, 2006.

[35] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

LYRICS-BASED MUSIC GENRE CLASSIFICATION USING A HIERARCHICAL ATTENTION NETWORK

Alexandros Tsaptsinos
ICME, Stanford University, USA
alextsap@stanford.edu

ABSTRACT

Music genre classification, especially using lyrics alone, remains a challenging topic in Music Information Retrieval. In this study we apply recurrent neural network models to classify a large dataset of intact song lyrics. As lyrics exhibit a hierarchical layer structure—in which words combine to form lines, lines form segments, and segments form a complete song—we adapt a hierarchical attention network (HAN) to exploit these layers and in addition learn the importance of the words, lines, and segments. We test the model over a 117-genre dataset and a reduced 20-genre dataset. Experimental results show that the HAN outperforms both non-neural models and simpler neural models, whilst also classifying over a higher number of genres than previous research. Through the learning process we can also visualise which words or lines in a song the model believes are important to classifying the genre. As a result the HAN provides insights, from a computational perspective, into lyrical structure and language features that differentiate musical genres.

1. INTRODUCTION

Automatic classification of music is an important and well-researched task in Music Information Retrieval (MIR) [25]. Previous work on this topic has focused primarily on classifying mood [13], genre [21], annotations [27], and artist [9]. Typically one or a combination of audio, lyrical, symbolic, and cultural data is used in machine learning algorithms for these tasks [23].

Genre classification using lyrics presents itself as a natural language processing (NLP) problem. In NLP the aim is to assign meaning and labels to text; here this equates to a genre classification of the lyrical text. Traditional approaches in text classification have utilised n-gram models and algorithms such as Support Vector Machines (SVM), k-Nearest Neighbour (k-NN), and Naïve Bayes (NB).

In recent years the use of deep learning methods such as recurrent neural networks (RNNs) or convolutional neural networks (CNNs) has produced superior results and represent an exciting breakthrough in NLP [16, 17]. Whilst linear and kernel models rely on good hand-selected features, these deep learning architectures circumvent this by letting models learn important features themselves.

Deep learning has in recent years been utilised in several MIR research topics including live score following [7], music instrument recognition [20], and automatic tagging [3]. In many cases, these approaches have led to significant improvements in performance. For example, Kum et al. [18] utilise multi-column deep neural networks to extract melody on vocal segments while Southall et al. [34] approach automatic drum transcription using bidirectional recurrent neural networks.

Neural methods have further been utilised for the genre classification task on audio and symbolic data. Sigtia and Dixon [31] use the hidden states of a neural network as features for song on which a Random Forest classifier was built, reporting an accuracy of 83% among 10 genres. Costa et al. [6] compare the performance of CNNs in genre classification through spectrograms with respect to results obtained through hand-selected features and SVMs. Jeong and Lee [14] learn temporal features in audio using a deep neural network and apply this to genre classification. However, not much research has looked into the performance of these deep learning methods with respect to the genre classification task on lyrics. Here, we attempt to remedy this situation by extending deep learning approaches to text classification to the particular case of lyrics.

Hierarchical methods attempt to use some sort of structure of the data to improve the models and have previously been utilised in vision classification tasks [30]. Yang et al. [37] propose a hierarchical attention network (HAN) for the task of document classification. Since documents often contain structure whereby words form to create sentences, sentences to paragraphs, etc. they introduce this knowledge to the model, resulting in superior classification results. It is evident that songs and, in particular, lyrics similarly contain a hierarchical composition: Words combine to form lines, lines combine to form segments, and segments combine to form the whole song. A segment of a song is a verse, chorus, bridge, etc. of a song and typically comprises several lines. The hierarchical nature of songs has been previously exploited in genre classification tasks with Du et al. [8] utilising hierarchical analysis of spectrograms to help classify genre.

Here, we propose application of an HAN for genre classification of intact lyrics. We train such a network, allowing it to apply attention to words, lines, and segments. Results show the network produces higher accuracies in the lyrical classification task than previous research and from the attention learned by the network we can observe which words are indicative of different genres.

The remainder of the paper is structured as follows. In Section 2 we describe our methods, including the dataset and a description of the HAN. In Section 3 we provide results and visualisations from our experiments. We conclude with a discussion in Section 4.

2. METHODS

2.1 Dataset

Research involving song lyrics has historically suffered from copyright issues. Consequently most previous literature has utilised count-based bag-of-words lyrics. In this format, structure and word order are lost, and it has been shown that utilising intact lyrics reveals superior results in classification tasks [11, 32].

Seeking an intact lyrics corpus for the present study, we obtained a collection of lyrics through a signed research agreement with LyricFind [1]. This corpus has been used in the past to study novelty [10] and influence [1] in lyrics. The complete set contained 1,039,151 song lyrics in JSON format, as well as basic metadata including artist(s) and track name. As the corpus provided no genre information, we aggregated it ourselves using the iTunes Search API [2], extracting the value for the `primaryGenreName` key as baseline truth. Several different sources were not used for consistency reasons with iTunes found to be the largest, easily accessible source with reasonable genre tags. This unfortunately still greatly reduced the size of the dataset due to the sparse iTunes database. We then further removed any songs that were linked with a genre tag of 'Music Video', leaving a dataset comprising 244 genres. As this dataset had a very long tail of sparse genres, we further filter the dataset via two methods. Firstly we remove any genres with less than 50 instances, giving a dataset of size 495,188 lyrics and 117 genres. Secondly we retain only the top 20 genres, giving a dataset of 449,458 lyrics. We note also that the dataset originally contained various versions of the same lyrics, due to the prevalence of cover songs; we retain only one of these versions chosen at random. The song lyrics are split into lines and segments which we tokenised using the `nltk` package [3] in Python. We split the dataset into a rough split of 80% for training, 10% for validation, and 10% for testing. All preprocessing was done via Python with the neural networks built using Tensorflow [4].

2.2 Hierarchical Attention Networks

The structure of the model follows that of Yang et al. [37]. Each layer is run through a bidirectional gated recurrent

[1] `http://lyricfind.com/`
[2] `http://apple.co/1qHOryr`
[3] `http://www.nltk.org/`
[4] `https://www.tensorflow.org/`

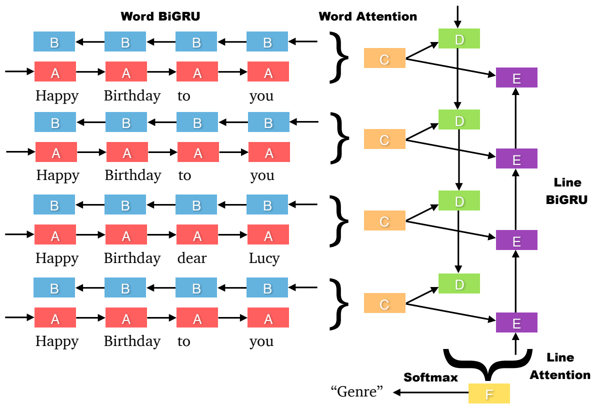


Figure 1: Representation of the HAN architecture; boxes represent vectors. A and B vectors represent the hidden states for the forward and backward pass of the GRU at the word level, respectively. The line vectors C are then obtained from these hidden states via the attention mechanism. The D and E vectors represent the forward and backward pass of the GRU at the line level, respectively. The song vector F is then obtained from these hidden states via the attention mechanism. Finally classification is performed via the softmax activation function.

unit (GRU) with attention applied to the output. The attention weights are used to create a vector via a weighted sum which is then passed as the input to the next layer. A representation of the architecture for the example song of 'Happy Birthday' can be seen in Figure 1, where the layers are applied at the word, line, and song level. We briefly step through the various components of the model.

2.2.1 *Word Embeddings*

An important idea in NLP is the use of dense vectors to represent words. A successful methodology proposes that similar words have similar context and thus vectors can be learned through their context, such as in the word2vec model [26]. Pennington et al. [29] propose the GloVe method which combines global matrix factorisation and local context window methods to produce word vectors that outperform previous word2vec and SVM based models.

Here we take as our vocabulary the top 30,000 most frequent words from the whole LyricFind corpus, including those from songs we did not match with a genre. We train 100-dimensional GloVe embeddings for these words using methods obtained from the GloVe website [5]. Previous research has shown that retraining these word vectors over the extrinsic task at hand can improve results if the dataset is large enough [5]. In a preliminary genre classification task we found that retraining these word embeddings did improve accuracy, and so we let our model learn superior embeddings to those provided by GloVe [29].

2.2.2 *Gated Recurrent Units*

Introduced by Chung et al. [4], GRUs are a form of gating mechanism in RNNs designed to help overcome the

[5] `http://nlp.stanford.edu/projects/glove/`

struggle to capture long-term dependencies in RNNs. This is achieved by the introduction of intermediate states between the hidden states in the RNN. An update gate z_t is introduced to help determine how important the previous hidden state is to the next hidden state. A reset gate r_t is introduced to help determine how important the previous hidden state is in the creation of the next memory. The hidden state is h_t, whilst new memory is computed and stored in $\tilde{h}_t$. Mathematically we describe the process as

$$z_t = \text{sigmoid}\left(W_z x_t + U_z h_{t-1} + b_z\right) \tag{1}$$

$$r_t = \text{sigmoid}\left(W_r x_t + U_r h_{t-1} + b_r\right) \tag{2}$$

$$\tilde{h}_t = \tanh\left(W_h x_t + r_t \circ U_h h_{t-1} + b_h\right) \tag{3}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t, \tag{4}$$

where x_t is the word vector input at time-step t, $\circ$ is the Hadamard product, and sigmoid is the sigmoid activation function. W_z, U_z, W_r, U_r, W_h, and U_h are weight matrices randomly initialised and to be learned by the model along with the b_z, b_r, and b_h bias terms. Bias terms were not included in the original model by Chung et al. [4], however have been included here as in Jozefowicz et al. [15].

2.2.3 *Hierarchical Attention*

Attention was first proposed by Bahdanau et al. [2] with respect to neural machine translation to allow the model to learn which words were more important in the translation objective. Along the lines of that study, we would like our model to learn which words are important in classifying genre and then apply more weight to these words. Similarly, we can apply attention again on lines or segments to let the model learn which lines or segments are more important in classification.

Given input vectors h_i for $i = 1, \ldots, n$ the attention mechanism can be formulated as

$$u_i = \tanh\left(W_a h_i + b_a\right) \tag{5}$$

$$\alpha_i = \frac{\exp(u_i^T u_a)}{\sum_{k=1}^{n} \exp(u_k^T u_a)} \tag{6}$$

$$s = \sum_{i=1}^{n} \alpha_i h_i, \tag{7}$$

where s is the output vector passed to the next layer consisting of the weighted sum of the current layers vectors. Parameters W_a, b_a, and u_a are learned by the model after random initialisation.

One layer of the network takes in vectors $x_1, \ldots, x_n$, applies a bidirectional GRU to find a forward hidden state $\overrightarrow{h}_j$ and a backward hidden state $\overleftarrow{h}_j$, and then uses the attention mechanism to form a weighted sum of these hidden states to output as the representation. Letting GRU indicate the output of a GRU and ATT represent the output from an attention mechanism, one layer is formulated as

$$\overrightarrow{h}_j = \overrightarrow{GRU}(x_j), \tag{8}$$

$$\overleftarrow{h}_j = \overleftarrow{GRU}(x_j), \tag{9}$$

$$h_j = [\overrightarrow{h}_j; \overleftarrow{h}_j], \tag{10}$$

$$s = ATT(h_1, \ldots, h_L). \tag{11}$$

Our HAN consists of two layers, one at the word level, and one at the line/segment level. Consider a song of L lines or segments s_j, each consisting of n_j words w_{ij}. Let E be the pre-trained word embedding matrix. Letting LAY represent the dimension reduction operation of a layer in the network as in Eqns 8–11 the whole HAN can be formulated for $i = 1, \ldots, n_j$ and $j = 1, \ldots, L$ as

$$x_{ij} = E w_{ij} \tag{12}$$

$$s_j = LAY(x_{1j}, \ldots, x_{n_j j}), \tag{13}$$

$$s = LAY(s_1, \ldots, s_L). \tag{14}$$

Each layer has its own set of GRU weight matrix and bias terms to learn, as well as its own attention weight matrix, bias terms, and relevance vector to learn.

2.2.4 *Classification*

With the song vector s now obtained, classification is performed by using a final softmax layer

$$p = \text{softmax}\left(W_p s + b_p\right), \tag{15}$$

where intuitively we take the entry of highest magnitude as the prediction for that song. To train the model we minimise over cross-entropy loss.

3. EXPERIMENTS

3.1 Baseline Models

We compare the performance of the HAN against various baseline models.

1. Majority classifier (MC): ‘Rock’ is the most common genre in our dataset. The MC simply predicts ‘Rock’.
2. Logistic regression (LR): A LR run on the average song word vector produced from the GloVe embeddings.
3. Long Short-Term Memory (LSTM): An LSTM, treating the whole song as a single sequence of words and use max-pooling of the hidden states for classification. Fifty hidden units were used in the LSTM and each song had a maximum of 600 words. For full discussion of the LSTM framework see Hochreiter and Schmidhuber [12].
4. Hierarchical network (HN-L): The HN structure in the absence of attention run at the line level. At each layer all of the representations are simply averaged to produce the next layer input.

For LR, LSTM, and HN-L we let the model retrain the word embeddings as it trained.

3.2 Model Configuration

The lyrics are padded/truncated to have uniform length. In the line model, each line has a maximum of 10 words and a maximum of 60 lines. In the segment model each segment has a maximum of 60 words and a maximum of 10 segments. Fifty hidden units are utilised in the bidirectional GRUs, whilst one hundred states are output from the

Model	117 Genres	20 Genres
MC	24.71	27.17
LR	35.21	38.13
LSTM	43.66	49.77
HN-L	45.85	49.09
HAN-L	46.42	49.50
HAN-S	45.05	47.60

Table 1: Genre classification test accuracies for the two datasets (%) using majority classifier (MC), logistic regression (LR), Long Short-Term Model (LSTM), hierarchical network (HN-L), and line- and segment-level HAN (HAN-L, HAN-S).

attention mechanisms. Before testing the model, hyperparameters were tuned on the validation set. Dropout [35] and gradient clipping [28] were both found to benefit the model. We dropout at each layer with probability $p = 0.5$ and gradients are clipped at a maximum norm of 1 in the backpropogation. We utilise a mini-batch size of 64 and optimise using RMSprop [36] with a learning rate of 0.01. The models were all run until their validation loss did not decrease for 3 successive epochs. In all the HAN models, this occurred between the 5th and 8th epoch.

The code to train the model and perform the experiments described are made publicly available [6] .

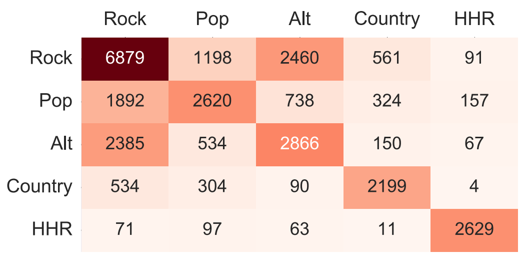
3.3 Results

For both dataset sizes we run the baseline models and the HAN at the line and segment level. Let HAN-L represent running over lines and HAN-S represent running over segments. The test accuracies are seen in Table 1.

From the results we see a trend between model complexity and classification accuracy. The very simple majority classifier performs weakest and is improved upon by the simple logistic regression on average bag-of-words. The neural-based models perform better than both of the simple models. The LSTM model, which takes into account word order and tries to implement a memory of these words, gives performances of 43.66% and 49.77%, outperforming the HAN on the 20-genre dataset. Over the 117-genre dataset the best performing models were the HANs, with a highest accuracy of 46.42% when run over lines. It is observed that for the simpler 20-genre case, the more complex HAN is not required since the simpler LSTM beats it, although the LSTM took almost twice as long to train as the HAN. However for the more challenging 117-genre case, the HAN-L outperforms the LSTM, perhaps picking up on more of the intricacies of rarer genres.

In both cases the HAN run at the line level produced superior results than that run over the segment level, giving a bump of roughly 1.4% and 1.9% in the 117-genre and 20-genre datasets, respectively. The HN-L, which is run at the line level, additionally outperforms the HAN at segment level. This indicates that the model performs better when looking at songs line by line rather than segment by

[6] `https://github.com/alexTsaptsinos/lyricsHAN`



	Rock	Pop	Alt	Country	HHR
Rock	6879	1198	2460	561	91
Pop	1892	2620	738	324	157
Alt	2385	534	2866	150	67
Country	534	304	90	2199	4
HHR	71	97	63	11	2629

Figure 2: HAN-L confusion matrix for Rock, Pop, Alternative (Alt), Country, and Hip-Hop/Rap (HHR) genres over larger (117-genre) dataset. Rows represent true genre, whilst columns are predicted.

segment. In the HAN-L the model can pick up on many repeated lines or lines of a similar ilk, rather than the few similar segments it attains in the HAN-S, and this may be attributive to the better performance. The network does benefit from the inclusion of attention, with HAN-L classifying with higher accuracies than HN-L. This increase is marginal and requires an increased cost, however allows for the extraction of attention in the visualisations of the following section.

As expected, classifying over the 20-genre dataset has given boosts of roughly 3% and 2.5% in the HAN-L and HAN-S, respectively. It is interesting to note that discarding roughly 10% of the data by only keeping roughly a sixth of the genres has not strengthened the model by much. Given the similarity of recognition performance between the two datasets, even with the simplest of models, it is likely that the extra genres are predominantly noise added to the 20-genre dataset. With the HAN-L outperforming the LSTM over the 117-genre dataset this then indicates that the model is more robust to noise.

The confusion matrix for HAN-L run over the larger dataset for the top 5 genres can be seen in Figure 2. We can see from the matrix that Rock, Pop, and Alternative (Alt) are all commonly confused; the model predicts Rock for Alternative almost as many times as it does Alternative. As the most common genre in the dataset by about 30,000 it is unsurprising to see the model try and predict Rock more often, and it is unclear whether a person would be able to distinguish between the lyrics of these genres. However, we see that both Country and Hip-Hop/Rap (HHR) are more separated. With their distinct lyrical qualities, especially in the case of Hip-Hop/Rap, this is an encouraging result indicating that the model has learned some of the qualities of both these genres.

3.3.1 Attention Visualisation

To help illustrate the attention mechanism, we feed song lyrics into the HAN-L and observe the weights it applies to words and lines. For each song we extract the 5 most heavily weighted lines and a visualisation of their weights and the individual word weights for a few different correctly predicted song lyrics can be seen in Figure 3.

From these visualisations we notice that the model has placed greater weights on words we may associate with a certain genre. For example ‘baby’ and ‘ai’ are weighted

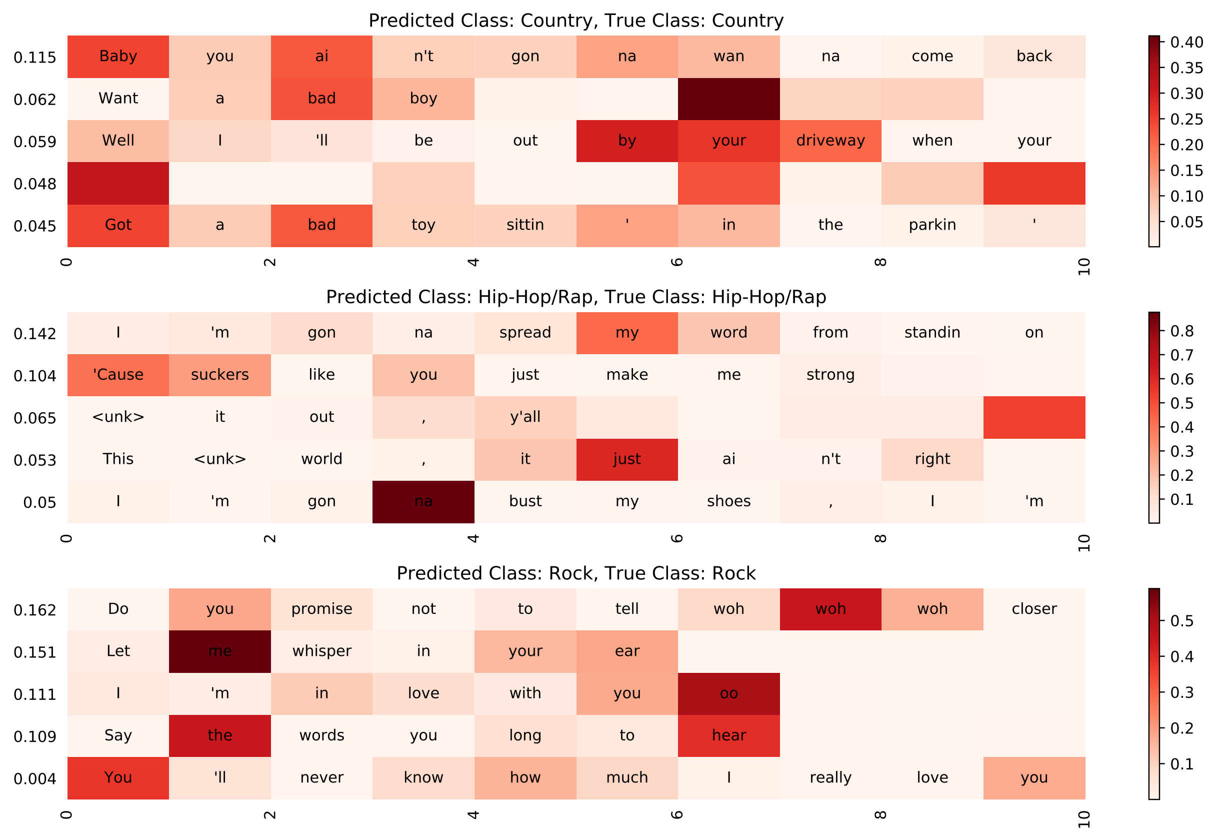


Figure 3: Weights applied by the HAN-L for song lyrics that were correctly classified. Line weights appear to the left of each line and word weights are coloured according to the respective colorbars on the right.

heavily in the Country song, and the most heavily weighted line in that song is characteristically Country. The model has placed great weight on a blank line, indicating the break between segments; it is unclear whether the model is learning to place importance on how songs are segmented and the number of segments occurring. In the Hip-Hop/Rap song the model places attention on colloquially spelled words ‘cause’ and ‘gonna’. Although not included here, it was observed that for many rap songs swear words and racial terms were heavily weighted. The model picks up the ‘woh’ and ‘oo’ in the Rock song and also heavily weights occurrences of second-person determiner ‘your’ and pronoun ‘you’. It was found that for many Rock songs this was the case.

In addition some visualisations of lyrics that were incorrectly classified by the HAN-L can be seen in Figure 4. We observe the model predicting Country for a Pop song, applying weights to ‘sin’ and ‘strong’ which could be characteristic of Country songs. The dataset contains songs with foreign language lyrics. Here we observe a song with Spanish lyrics classed as Pop Latino by the model whilst iTunes deems it Pop. This seems like a fair mistake for the model to have made since it has evidently recognised the Spanish language. The model also incorrectly classifies the Hip-Hop/Rap song as Pop. In the 5 most heavily weighted lines we do not spot any instances of language that indicate a Hip-Hop/Rap song and we hypothesise that the genericness of the lyrics has led the model to predict Pop.

4. DISCUSSION

Genre is an inherently ambiguous construct, but one that plays a major role in categorising musical works [24, 33]. From one standpoint, genre classification by lyrics will always be inherently flawed by vague genre boundaries and many genres borrowing lyrics and styles from one another. Previous research has shown that lyrical data performs weakest in genre classification compared to other forms of data [23]. As a consequence, this problem is not as well researched and preference has been given to other methods.

SVMs, k-NN, and NB have been heavily used in previous lyrical classification research. In addition very rarely has research looked into classifying more than between 10 genres despite the prevalence of clearly many more genres. Fell and Sporleder classify among 8 genres using n-grams along with other hand-selected features to help represent vocabulary, style, structure, and semantics [11]. Ying et al. make use of POS tags and classify among 10 genres using SVMs, k-NN, NB with a highest accuracy of 39.94% [38]. McKay et al. utilise hand-selected features to produce classification accuracies of 69% among 5 genres and 43% among 10 genres [23].

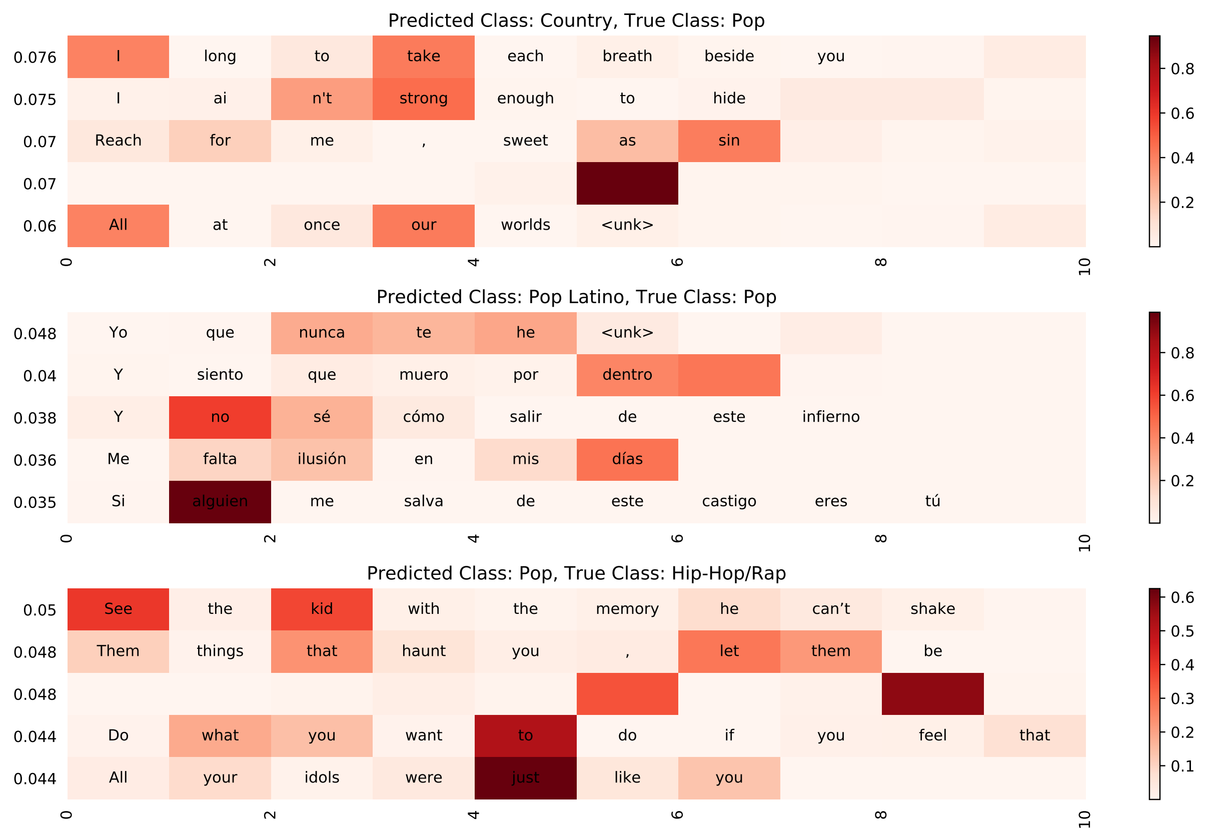


Figure 4: Weights applied by the HAN-L for song lyrics that were incorrectly classified. Line weights appear to the left of each line and word weights are coloured according to the respective colorbars on the right.

In this paper we have shown that an HAN and other neural-based methods can improve on the genre classification accuracy. In large part this model has beaten all previously reported lyrical-only genre classification model accuracies, except for the classification among 5 genres. Whilst having been trained on different datasets the jump in classification accuracies achieved by the HAN and LSTM across the 20-genre datasets compared to previous research indicate that neural structures are clearly beneficial. However, with very similar results between the neural structures it is still unclear what the optimal neural structure may be and there is certainly room for further experimentation. We have shown that the HAN works better with layers at the word, line, and song level rather than word, segment, and song level. One known issue of the present dataset is that iTunes attributes genres by artist, not by track; this is a problem for artists whose work may cover multiple genres and is something that should be addressed in the future. A larger issue concerns the accuracy of the iTunes genre labels more generally, especially for the larger 117-genre dataset which naturally includes more subjective and vague genre definitions.

Visualisations of the weights the HAN applies to words and lines were produced to help see what the model was learning. In a good amount of cases, words and lines were heavily weighted that were cohesive with the song genre; however, this was not always the case. We note that in general the model tended to let one word dominate a single line with the greatest weight. However this was not as apparent across lines, with weights among lines more evenly spread. With a large amount of foreign-language lyrics also present in the dataset, an idea for further research is to build a classifier that identifies language, and from there classifies by genre. Any such research would be inhibited, however, by the lack of such a rich dataset to train on.

To produce a state-of-the-art classifier it is evident that the classifier must take into account more than just the lyrical content of the song. Mayer et al. combine audio and lyrical data to produce a highest accuracy of 63.50% within 10 genres via SVMs [21]. Mayer and Rauber then use a cartesian ensemble of lyric and audio features to gain a highest accuracy of 74.08% within 10 genres [22]. Further research could look into employing this hierarchical attention model to the audio and symbolic data, and combining with the lyrics to build a stronger classifier. Employment of the HAN in the task of mood classification via sentiment analysis is another possible area of research. In addition the HAN could be extended to include both a layer at the line and segment level, or even at the character level, to explore performance.

5. ACKNOWLEDGEMENTS

Many thanks to Will Mills and Mohamed Moutadayne from LyricFind for providing access to the data, and the ISMIR reviewers for their helpful comments.

6. REFERENCES

[1] Jack Atherton and Blair Kaneshiro. I said it first: Topological analysis of lyrical influence networks. In *ISMIR*, pages 654–660, 2016.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Keunwoo Choi, George Fazekas, and Mark Sandler. Automatic tagging using deep convolutional neural networks. *arXiv preprint arXiv:1606.00298*, 2016.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[6] Y. MG Costa, L. S. Oliveira, and C. N. Silla. An evaluation of convolutional neural networks for music classification using spectrograms. *Applied Soft Computing*, 52:28–38, 2017.

[7] Matthias Dorfer, Andreas Arzt, Sebastian Böck, Amaury Durand, and Gerhard Widmer. Live score following on sheet music images. *arXiv preprint arXiv:1612.05076*, 2016.

[8] W. Du, H. Lin, J. Sun, B. Yu, and H. Yang. A new hierarchical method for music genre classification. In *CISP-BMEI*, pages 1033–1037. IEEE, 2016.

[9] Hamid Eghbal-Zadeh, Markus Schedl, and Gerhard Widmer. Timbral modeling for music artist recognition using i-vectors. In *EUSIPCO*, pages 1286–1290. IEEE, 2015.

[10] R. J. Ellis, Fang J. Xing, Z., and Y. Wang. Quantifying lexical novelty in song lyrics. In *ISMIR*, pages 694–700, 2015.

[11] M. Fell and C. Sporleder. Lyrics-based analysis and classification of music. In *COLING*, volume 2014, pages 620–631, 2014.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] X. Hu and J. S. Downie. Improving mood classification in music digital libraries by combining lyrics and audio. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 159–168. ACM, 2010.

[14] Il-Young Jeong and Kyogu Lee. Learning temporal features using a deep neural network and its application to music genre classification. In *ISMIR*, pages 434–440, 2016.

[15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *ICML*, pages 2342–2350, 2015.

[16] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[17] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[18] Sangeun Kum, Changheun Oh, and Juhan Nam. Melody extraction on vocal segments using multi-column deep neural networks. In *ISMIR*, pages 819–825, 2016.

[19] T. LH Li, A. B. Chan, and A. Chun. Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*, 2010.

[20] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for musical instrument recognition. *arXiv preprint arXiv:1605.06644*, 2016.

[21] R. Mayer, R. Neumayer, and A. Rauber. Combination of audio and lyrics features for genre classification in digital audio collections. In *Proceedings of the 16th ACM international conference on Multimedia*, pages 159–168. ACM, 2008.

[22] R. Mayer and A. Rauber. Musical genre classification by ensembles of audio and lyrics features. In *ISMIR*, pages 675–680, 2011.

[23] C. McKay, J. A. Burgoyne, J. Hockman, J. BL Smith, G. Vigliensoni, and I. Fujinaga. Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features. In *ISMIR*, pages 213–218, 2010.

[24] C. McKay and I. Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR*, pages 101–106, 2006.

[25] M. McKinney and J. Breebaart. Feature for audio and music classification. In *ISMIR*, pages 151–158, 2003.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[27] J. Nam, J. Herrera, M. Slaney, and J. O. Smith. Learning sparse feature representations for music annotation and retrieval. In *ISMIR*, pages 565–570, 2012.

[28] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML*, 28:1310–1318, 2013.

[29] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[30] P. H. Seo, Z. Lin, S. Cohen, X. Shen, and B. Han. Progressive attention networks for visual attribute prediction. *arXiv preprint arXiv:1606.02393*, 2016.

[31] S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In *ICASSP*, pages 6959–6963. IEEE, 2014.

[32] A. Smith, C. Zee, and A. Uitdenbogerd. In your eyes: Identifying clichés in song lyrics. In *Australasian Language Technology Workshop*, pages 88–96, 2012.

[33] M. Sordo, O. Celma, M. Blech, and E. Guaus. The quest for musical genres: Do the experts and the wisdom of crowds agree? In *ISMIR*, pages 255–260, 2008.

[34] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *ISMIR*, pages 591–597, 2016.

[35] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[36] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

[37] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489, 2016.

[38] T. C. Ying, S. Doraisamy, and L. N. Abdullah. Genre and mood classification using lyric features. In *International Conference on Information Retrieval & Knowledge Management*, pages 260–263. IEEE, 2012.

METRICAL-ACCENT AWARE VOCAL ONSET DETECTION IN POLYPHONIC AUDIO

Georgi Dzhambazov[1] **Andre Holzapfel**[2]
Ajay Srinivasamurthy[1] **Xavier Serra**[1]
[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Media Technology and Interaction Design, KTH Royal Institute of Technology, Stockholm, Sweden
{georgi.dzhambazov,ajays.murthy,xavier.serra}@upf.edu, holzap@kth.se

ABSTRACT

The goal of this study is the automatic detection of onsets of the singing voice in polyphonic audio recordings. Starting with a hypothesis that the knowledge of the current position in a metrical cycle (i.e. metrical accent) can improve the accuracy of vocal note onset detection, we propose a novel probabilistic model to jointly track beats and vocal note onsets. The proposed model extends a state of the art model for beat and meter tracking, in which a-priori probability of a note at a specific metrical accent interacts with the probability of observing a vocal note onset. We carry out an evaluation on a varied collection of multi-instrument datasets from two music traditions (English popular music and Turkish makam) with different types of metrical cycles and singing styles. Results confirm that the proposed model reasonably improves vocal note onset detection accuracy compared to a baseline model that does not take metrical position into account.

1. INTRODUCTION

Singing voice analysis is one of the most important topics in the field of music information retrieval because singing voice often forms the melody line and creates the impression of a musical piece. The automatic transcription of singing voice can be considered to be a key technology in computational studies of singing voice. It can be utilized for end-user applications such as enriched music listening and singing education. It can as well enable other computational tasks including singing voice separation, karaoke-like singing voice suppression or lyrics-to-audio alignment [5].

The process of converting an audio recording into some form of musical notation is commonly known as automatic music transcription. Current transcription methods use general purpose models, which are unable to capture the rich diversity found in music signals [2]. In particular, singing voice poses a challenge to transcription algorithms because of its soft onsets, and phenomena such as portamento and vibrato. One of the core subtasks of singing voice transcription (SVT) is detecting note events with a discrete pitch value, an onset time and an offset time from the estimated time-pitch representation. Detecting the time locations of vocal note onsets can benefit from automatically detected events from musical facets, such as musical meter. In fact, the accents in the metrical cycle determine to a large extent the temporal backbone of singing melody lines. Studies on sheet music showed that the locations of vocal note onsets are influenced by the their position in a metrical cycle [7, 10]. Despite that, there have been few studies on meter aware analysis of onsets in music audio [4].

In this work we propose a novel probabilistic model that tracks simultaneously note onsets of singing voice and instrumental energy accents in a metrical cycle. We extend a state of the art model for beat and meter tracking, based on dynamic Bayesian networks (DBN). A model variable is added that models the temporal segments of a note and their interaction with metrical position. The proposed model is applied for the automatic detection of vocal note onsets in multi-instrumental recordings with predominant singing voice. Evaluation is carried out on datasets from music traditions, for which there is a clear correlation between metrical accents and the onset times in the vocal line.

2. RELATED WORK

2.1 Singing Voice Transcription

A probabilistic note hidden Markov model (HMM) is presented in [18], where a note has 3 states: attack (onset), stable pitch state and silent state. The transition probabilities are learned from data. Recently [14] suggested to compact musical knowledge into rules as a way to describe the observation and transition likelihoods, instead of learning them from data. The authors suggest covering a range with distinct pitch from lowest MIDI C2 up to B7. Each MIDI pitch is further divided into 3 sub-pitches, resulting in $n = 207$ notes with different pitch, each having the 3 note states. Although being conceptually capable of tracking onsets in singing voice audio with accompaniment, these approaches were tested only on a cappella singing.

In multi-instrumental recordings, an essential first step is to extract reliably the predominant vocal melody. There have been few works dealing with SVT in multi-instrumental recordings in general [13, 15], and with onset detection, in particular [3]. Some of them [13, 15] rely on the algorithm for predominant melody extraction of [19].

2.2 Beat Detection

Recently a Bayesian approach, referred to as the *bar-pointer* model, has been presented [20]. It describes events in music as being driven by their current position in a metrical cycle (i.e. musical bar). The model represents as hidden variables in a Dynamic Bayesian network (DBN) the current position in a bar, the tempo, and the type of musical meter, which can be referred to as bar-tempo state space.

The work of [9] applied this model to recordings from non-Western music, in order to handle jointly beat and downbeat tracking. The authors showed that the original model can be adapted to different rhythmic styles and time signatures, and an evaluation is presented on Indian, Cretan and Turkish music datasets.

Later [12] suggested a modification of the bar-tempo state space, in order to reduce the computational burden from its huge size.

3. DATASETS

3.1 Turkish Makam

The Turkish dataset has two meter types, referred to as usuls in Turkish makam: the 9/8-usul aksak and the 8/8-usul düyek. It is a subset of the dataset presented in [9], including only the recordings with singing voice present. The beats and downbeats were annotated by [9]. The vocal note onsets are annotated by the first author, whereby only pitched onsets are considered (2100 onsets). To this end, if a syllable starts with an unvoiced consonant, the onset is placed at the beginning of the succeeding voiced phoneme [1].

For this study we divided the dataset into training and test subsets. The test dataset comprises 5 1-minute excerpts from recordings with solo singing voice only for each of the two usuls (on total 780 onsets). The training dataset spans around 7 minutes of audio from each of the two usuls. Due to the scarcity of material with solo singing voice, several excerpts with choir sections were included in the training data.

3.2 English Pop

The datasets, on which singing voice transcription in multi-instrumental music is evaluated, are very few [2]: Often a subset of the RWC dataset is employed, which does not contain diverse genres and singers [6]. To overcome this bias, we compiled the *lakh-vocal-segments* dataset: We selected 14 30-second audio clips of English pop songs, which have been aligned to their corresponding MIDIs in a recent study [17]. Criteria for selecting the clips are the predominance of the vocal line; 4/4 meter; correlation between the beats and the onset times. We derived the locations of the vocal onsets (850 on total) from the aligned vocal MIDI channel, whereby some imprecise locations were manually corrected. To encourage further studies on singing voice transcription we make available the derived annotations [2].

4. APPROACH

The proposed approach extends the beat and meter tracking model, presented in [12]. We adopt from it the variables for the position in the metircal cycle (bar position) ϕ and the instantaneous tempo $\dot{\phi}$. We also adopt the observation model, which describes how the metrical accents (beats) are related to an observed onset feature vector y_f. All variables and their conditional dependencies are represented as the hidden variables in a DBN (see Figure 1). We consider that the *a priori* probability of a note at a specific metrical accent interacts with the probability of observing a vocal note onset. To represent that interaction we add a hidden state for the temporal segment of a vocal note *n*, which depends on the current position in the metrical cycle. The probability of observing a vocal onset is derived from the emitted pitch y_p of the vocal melody.

In the proposed DBN, an observed sequence of features derived from an audio signal $y_{1:K} = \{y, .., y_K\}$ is generated by a sequence of hidden (unknown) variables $x_{1:K} = \{x_1, ..., x_K\}$, where K is the length of the sequence (number of audio frames in an audio excerpt). The joint probability distribution of hidden and observed variables factorizes as:

$$P(x_{1:K}, y_{1:K}) = P(x_0)\Pi_{k=1}^{K} P(x_k|x_{k-1})P(y_k|x_k) \quad (1)$$

where $P(x_0)$ is the initial state distribution; $P(x_k|x_{k-1})$ is the transition model and $P(y_k|x_k)$ is the observation model.

4.1 Hidden Variables

At each audio frame k, the hidden variables describe the state of a hypothetical bar pointer $x_k = [\dot{\phi}_k, \phi_k, n_k]$, representing the instantaneous tempo, the bar position and the vocal note respectively.

4.1.1 Tempo State $\dot{\phi}$ and Bar Position State ϕ

The bar position ϕ points to the current position in the metrical cycle (bar). The instantaneous tempo $\dot{\phi}$ encodes how many bar positions the pointer advances from the current to the next time instant. To assure feasible computational time we relied on the combined bar-tempo efficient state space, presented in [12]. To keep the size of the bar-tempo

[1] The dataset is described at `http://compmusic.upf.edu/node/345`

[2] `https://github.com/georgid/lakh_vocal_segments_dataset`

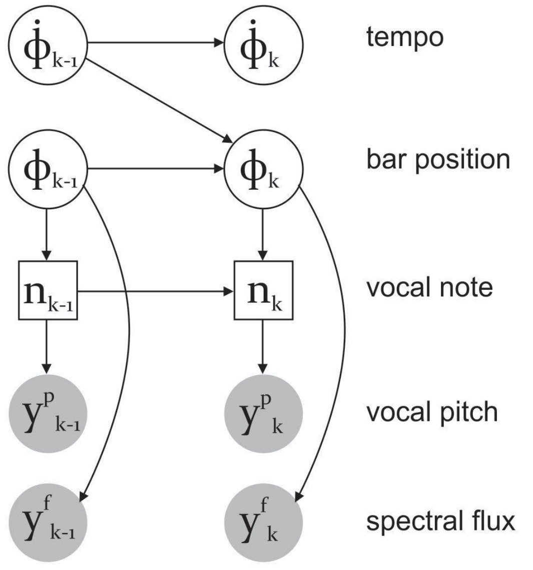


Figure 1: A dynamic Bayesian network for the proposed beat and vocal onset detection model. Circles and squares denote continuous and discrete variables, respectively. Gray nodes and white nodes represent observed and hidden variables, respectively.

state space small, we input the ground truth tempo for each recording, allowing a range for $\dot{\phi}$ within ± 10 bpm from it, in order to accommodate gradual tempo changes. This was the minimal margin at which beat tracking accuracy did not degrade substantially. For a study with data with higher stylistic diversity, it would make sense to increase it to at least 20% as it is done in [8, Section 5.2]. This yields around 100-1000 states for the bar positions within a single beat (in the order of 5000 for 4 beats, and 10000 for 8-9 beats for the usuls).

4.1.2 *Vocal Note State n*

The vocal note states represent the temporal segments of a sung note. They are a modified version of these suggested in the note transcription model of [14]. We adopted the first two segments: attack region (A), stable pitch region (S). We replaced the silent segment with non-vocal state (N). Because full-fledged note transcription is outside the scope of this work, instead of 3 steps per semitone, we used for simplicity only a single one, which deteriorated just slightly the note onset detection accuracy. Also, to reflect the pitch range in the datasets, on which we evaluate, we set as minimal MIDI note E3 covering almost 3 octaves up to B5 (35 semitones). This totals to 105 note states.

To be able to represent the DBN as an HMM, the bar-tempo efficient state space is combined with the note state space into a joint state space x. The joint state space is a cartesian product of the two state spaces, resulting in up to $10000 \times 105 \approx 1$ M states.

4.2 Transition Model

Due to the conditional dependence relations in Figure 1 the transitional model factorizes as

$$\begin{aligned} P(x_k|x_{k-1}) = \quad & P(\dot{\phi}_k|\dot{\phi}_{k-1}) \times \\ P(\phi_k|\phi_{k-1}, \dot{\phi}_{k-1}) \times \quad & P(n_k|n_{k-1}, \phi_k) \end{aligned} \tag{2}$$

The tempo transition probability $p(\dot{\phi}_k|\dot{\phi}_{k-1})$ and bar position probability $p(\phi_k|\phi_{k-1}, \dot{\phi}_{k-1})$ are the same as in [12]. Transition from one tempo to another is allowed only at bar positions, at which the beat changes. This is a reasonable assumption for the local tempo deviations in the analyzed datasets, which can be considered to occur relatively beat-wise.

4.2.1 *Note Transition Probability*

The probability of advancing to a next note state is based on the transitions of the note-HMM, introduced in [14]. Let us briefly review it: From a given note segment the only possibility is to progress to its following note segment. To ensure continuity each of the self-transition probabilities is rather high, given by constants c_A, c_S and c_N for A, S and N segments respectively (c_A=0.9; c_S=0.99; $c_N = 0.9999$). Let $P_{N_iA_j}$ be the probability of transition from non-vocal state N_i after note i to attack state A_j of its following note j. The authors assume that it depends on the difference between the pitch values of notes i and j and it can be approximated by a normal distribution centered at change of zero ([14], Figure 1.b). This implies that small pitch changes are more likely than larger ones. Now we can formalize their note transition as:

$$p(n_k|n_{k-1}) = \begin{cases} P_{N_iA_j}, & n_{k-1} = N_i \quad n_k = A_j \\ c_N, & n_{k-1} = n_k = N_i \\ 1 - c_A, & n_{k-1=}A_i \quad n_k = S_j \\ c_A, & n_{k-1} = n_k = A_i \\ 1 - c_S & n_{k=1} = S_i \quad n_k = N_j \\ c_S, & n_{k-1} = n_k = S_i \\ 0 & else \end{cases} \tag{3}$$

Note that the outbound transitions from all non-vocal states N_i should sum to 1, meaning that

$$c_N = 1 - \sum_i P_{N_iA_j} \tag{4}$$

In this study, we modify $P_{N_iA_j}$ to allow variation in time, depending on the current bar position ϕ_k.

$$p(n_k|n_{k-1,}\phi_k) = \begin{cases} P_{N_iA_j}\Theta(\phi_k), & n_{k-1} = N_i, n_k = A_j \\ c_N, & n_{k-1} = n_k = N_i \\ \ldots \end{cases} \tag{5}$$

where

$\Theta(\phi_k)$: function weighting the contribution of a beat adjacent to current bar position ϕ_k

and

$$c_N = 1 - \Theta(\phi_k) \sum_i P_{N_i A_j} \tag{6}$$

The transition probabilities in all the rest of the cases remain the same. We explore two variants of the weighting function $\Theta(\phi_k)$:

1. Time-window redistribution weighting: Singers often advance or delay slightly note onsets off the location of a beat. The work [15] presented an idea on how to model vocal onsets, time-shifted from a beat, by stochastic distribution. Similarly, we introduce a normal distribution $\mathcal{N}_{0,\sigma}$, centered around 0 to re-distribute the importance of a metrical accent (beat) over a time window around it. Let b_k be the beat, closest in time to a current bar position ϕ_k. Now:

$$\Theta(\phi_k) = [\mathcal{N}_{0,\sigma}(d(\phi_k, b_k))]^w e(b_k) \tag{7}$$

where

$e(b)$: probability of a note onset co-occurring with the b^{th} beat (b $\in B$); B is the number of beats in a metrical cycle

w : sensitivity of vocal onset probability to beats

$d(\phi_k, b_k)$: the distance from current bar position ϕ_k to the position of the closest beat b_k

Equation 5 means essentially that the original $P_{N_i A_j}$ is scaled according to how close in time to a beat it is.

2. Simple weighting: We also aim at testing a more conservative hypothesis that it is sufficient to approximate the influence of metrical accents only at the locations of beats. To reflect that, we modify the $P_{N_i A_j}$ only at bar positions corresponding to beat positions, for which the weighting function is set to the peak of $N_{0,\sigma}$, and to 1 elsewhere.

$$\Theta(\phi_k) = \begin{cases} [N_{0,\sigma}(0)]^w e(b_k), & d(\phi_k, b_k) = 0 \\ 1 & else \end{cases} \tag{8}$$

4.3 Observation Models

The observation probability $P(y_k|x_k)$ describes the relation between the hidden states and the (observed) audio signal. In this work we make the assumption that the observed vocal pitch and the observed metrical accent are conditionally independent from each other. This assumption may not hold in cases when energy accents of singing voice, which contribute to the total energy of the signal, are correlated to changes in pitch. However, for music with percussive instruments the importance of singing voice accents is diminished to a significant extent by percussive accents. Now we can rewrite Eq. 1 as

$$\begin{gathered} P(x_{1:K}, y^f_{1:K}, y^p_{1:K}) = \\ P(x_0)\Pi_{k=1}^{K} P(x_k|x_{k-1}) P(y^f_k|x_k) P(y^p_k|x_k) \end{gathered} \tag{9}$$

This means essentially that the observation probability can be represented as the product of the observation probability of a metrical accent $P(y^f_k|x_k)$ and the observation probability of vocal pitch $P(y^p_k|x_k)$.

4.3.1 Accent Observation Model

In this paper for $P(y^f_k|x_k)$ we train GMMs on the spectral flux-like feature y^f, extracted from the audio signal using the same parameters as in [12] and [9]. The feature vector y^f summarizes the energy changes (accents) that are likely to be related to the onsets of all instruments together. This forms a rhythmic pattern of the accents, characteristic for a given metrical type. The probability of observing an accent thus depends on the position in the rhythmic pattern, $P(y^f_k|x_k) = P(y^f_k|\phi_k)$.

4.3.2 Pitch Observation Model

The pitch probability $P(y^p_k|x_k)$ reduces to $P(y^p_k|n_k)$, because it depends only on the current vocal note state. We adopt the idea proposed in [14] that a note state emits pitch y^p according to a normal distribution, centered around its average pitch. The standard deviation of stable states and the one of the onset states are kept the same as in the original model, respectively 0.9 and 5 semitones. The melody contour of singing is extracted in a preprocessing step. We utilized for English pop a method for predominant melody extraction [19]. For Turkish makam, we instead utilized an algorithm, extended from [19] and tailored to Turkish makam [1]. In both algorithms, each audio frame k gets assigned a pitch value and probability of being voiced v_k. Based on frames with zero probabilities, one can infer which segments are vocal and which not. Since correct vocal segments is crucial for the sake of this study and the voicing estimation of these melody extraction algorithms are not state of the art, we manually annotated segments with singing voice, and thus assigned $v_k = 0$ for all frames, annotated as non-vocal.

For each state the observation probability $P(y^p_k|n_k)$ of vocal states is normalized to sum to v_k (unlike the original model which sums to a global constant v). This leaves the probability for each non-vocal state be $1 - v_k / n$.

4.4 Learning Model Parameters

4.4.1 Accent Observation Model

We trained the metrical accent probability $P(y^f_k|\phi_k)$ separately for each meter type: The Turkish meters are trained on the training subset of the makam dataset (see section 3.1). For each usul (8/8 and 9/8) we trained a rhythmic pattern by fitting a 2-mixture GMM on the extracted feature vector y^f. Analogously to [12], we pooled the bar positions down to 16 patterns per beat. For English pop we used the 4/4 rhythmic pattern, trained by [11] on ballroom dances. The feature vector is normalized to zero mean, unit variance and taking moving average. Normalization is done per song.

4.4.2 *Probability of Note Onset*

The probability of a vocal note onset co-occurring at a given bar position $e(b)$ is obtained from studies on sheet music. Many notes are aligned with a beat in the music score, meaning a higher probability of a note at beats compared to inter-beat bar positions. A separate distribution $e(b)$ is applied for each different metrical cycle. For the Turkish usuls $e(b)$ has been inferred from a recent study [7, Figure 5. a-c]. The authors used a corpus of music scores, on data from the same corpus, from which we derived the Turkish dataset. The patterns reveal that notes are expected to be located with much higher likelihoods on those beats with percussive strokes than on the rest.

In comparison to a classical tradition like makam, in modern pop music the most likely positions of vocal accents in a bar are arguably much more heterogeneous, due to the big diversity of time-deviations from one singing style to another [10]. Due to lack of a distribution pattern $e(b)$, characteristic for English pop, we set it manually with probabilities $(0.8, 0.6, 0.8, 0.6)$ for the 4 beats.

4.5 Inference

We obtain the most optimal state sequence $x_{1:K}$ by decoding with the Viterbi algorithm. A note onset is detected when the state path enters an attack note state after being in non-vocal state.

4.5.1 *With Manually Annotated Beats*

We explored the option that beats are given as input from a preprocessing step (i.e. when they are manually annotated). In this case, the detection of vocal onsets can be carried out by a reduced model with a single hidden variable: the note state. The observation model is then reduced to the pitch observation probability. The transition model is reduced to a bar-position aware transition probability $a_{ij}(k) = p(n_k = j|n_{k-1} = i, \phi_k)$ (see Eq. 5). To represent the time-dependent self-transition probabilities we utilize time-varying transition matrix. The standard transition probabilities in the Viterbi maximization step are substituted for the bar-position aware transitions $a_{ij}(k)$

$$\delta_k(j) = \max_{i \in (j, j-1)} \delta_{k-1}(i)\, a_{ij}(k)\, b_j(O_k) \qquad (10)$$

Here $b_j(O_k)$ is the observation probability for state i for feature vector O_k and $\delta_k(j)$ is the probability for the path with highest probability ending in state j at time k (complying with the notation of [16, III. B]

4.5.2 *Full Model*

In addition to onsets, a beat is detected when the bar position variable hits one of B positions of beats within the metrical cycle.

Note that the size of the state space x poses a memory requirement. A recording of 1 minute has around 10000 frames at a hopsize of 5.8 ms. To use Viterbi thus requires to store in memory pointers to up to 4 G states, which amounts to 40 G RAM (with uint32 python data type).

5. EXPERIMENTS

The hopsize of computing the spectral flux feature, which resulted in most optimal beat detection accuracy in [12] is $h_f = 20$ ms. In comparison, the hopsize of predominant vocal melody detection is usually of smaller order i.e. $h_p = 5.8$ ms (corresponding to 256 frames at sampling rate of 44100). Preliminary experiments showed that extracting pitch with values of h_p bigger than this values reasonably deteriorates the vocal onset accuracy. Therefore in this work we use hopsize of 5.8 ms for the extraction of both features. The time difference parameter for the spectral flux computation remains unaffected by this change in hopsize, because it can be set separately.

As a baseline we run the algorithm of [14] with the 105 note states, we introduced in Section 4.1.2 [3]. The note transition probability is the original as presented in Eq. 3, i.e. not aware of beats. Note that in [14] the authors introduce a post-processing step, in which onsets of consecutive sung notes with same pitch are detected considering their intensity difference. We excluded this step in all system variants presented, because it could not be integrated in the proposed observation model in a trivial way. This means that, essentially, in this paper cases of consecutive same-pitch notes are missed, which decreases inevitably recall, compared to the original algorithm.

5.1 Evaluation Metrics

5.1.1 *Beat Detection*

Since improvement of the beat detector is outside the scope of this study, we report accuracy of detected beats only in terms of their f-measure [4]. This serves solely the sake of comparison to existing work [5]. The f-measure can take a maximum value of 1, while beats tapped on the off-beat relative to annotations will be assigned an f-measure of 0. We used the default tolerance window of 70 ms, also applied in [9].

5.1.2 *Vocal Onset Detection*

We measured vocal onset accuracy in terms of precision and recall [6]. Unlike a cappella singing, the exact onset times of singing voice accompanied by instruments, might be much more ambiguous. To accommodate this fact, we adopted the tolerance of $t = 50$ ms, used for vocal onsets in accompanied flamenco singing by [13], which is much bigger than the $t = 5$ ms used by [14] for a cappella. Note that measuring transcription accuracy remains outside the scope of this study.

[3] We ported the original VAMP plugin implementation to python, which is available at `https://github.com/georgid/pypYIN`

[4] The evaluation script used is at `https://github.com/CPJKU/madmom/blob/master/madmom/evaluation/beats.py`

[5] Note that the f-measure is agnostic to the phase of the detected beats, which is clearly not optimal

[6] We used the evaluation script available at `https://github.com/craffel/mir_eval/blob/master/mir_eval/onset.py#L56`

meter		beat Fmeas	P	R	Fmeas	meter		beat Fmeas	P	R	Fmeas
aksak	Mauch	-	33.1	31.6	31.6	4/4	Mauch	-	29.6	38.3	33.2
	Ex-1	-	37.5	38.4	37.2		Ex-1	-	30.3	42.5	35.1
	Ex-2	86.4	37.8	36.1	36.1		Ex-2	94.2	31.6	39.4	34.4
düyek	Mauch	-	42.1	36.9	37.9	total	Mauch	-	34.8	35.6	35.2
	Ex-1	-	44.3	41.0	41.4		Ex-1	-	38.3	40.6	39.5
	Ex-2	72.9	45.0	39.0	40.3		Ex-2	84.3	38.1	38.2	38.1

Table 1: Evaluation results for Experiment 1 (shown as Ex-1) and Experiment 2 (shown as Ex-2). Mauch stands for the baseline, following the approach of [14]. P, R and Fmeas denote the precision, recall and f-measure of detected vocal onsets. Results are averaged per meter type.

5.2 Experiment 1: With Manually Annotated Beats

As a precursor to evaluating the full-fledged model, we conducted an experiment with manually annotated beats. This is done to test the general feasibility of the proposed note transition model (presented in 4.2.1), unbiased from errors in the beat detection.

We did apply both the simple and the time-redistribution weighting schemes, presented respectively in Eq. 8 and in Eq. 7. In preliminary experiments we saw that with annotated beats the simple weighting yields much worse onset accuracy than the time-redistributed one. Therefore the results reported are conducted with the latter weighting.

We have tested different pairs of values for w and σ from Eq. 5. For Turkish makam the onset detection accuracy peaks at $w = 1.2$ and $\sigma = 30\,\text{ms}$, whereas for the English pop optimal are $w = 1.1$ and $\sigma = 45\,\text{ms}$. Table 1 presents metrics compared to the baseline[7]. Inspection of detections revealed that the metrical-accent aware model could successfully detect certain onsets close to beats, which are omitted by the baseline.

5.3 Experiment 2: Full Model

To assure computational efficient decoding, we did an efficient implementation of the joint state space of [12][8]. To compare to that work, we measured the beat detection with both their original implementation and our proposed one. Expectedly, the average f-measure of the detected beats were the same for each of the three metrical cycle types in the datasets, which can be seen in Table 1. For aksak and düyek usuls, the accuracy is somewhat worse than the results of 91 and 85.2 respectively, reported in [9, Table 1.a-c, R=1]. We believe the reason is in the smaller size of our training data. Table 1 evidences also a reasonable improvement of the vocal onset detection accuracy for both music traditions. The results reported are only with the simple weighting scheme for the vocal note onset transition model (the time-redistribution weighting was not implemented in this experiment).

Adding the automatic beat tracking improved the baseline, whereas this was not the case with manual beats for simple weighting. This suggests that the concurrent tracking of beats and vocal onsets is a flexible strategy and can accommodate some vocal onsets, slightly time-shifted from a beat. We observe also that the vocal onset accuracy is on average a bit inferior to that with manual beat annotations (done with the time-redistribution weighting).

For the 4/4 meter, despite the highest beat detection accuracy, the improvement of onset accuracy over the baseline is the least. One reason for that may be that the note probability pattern $e(b)$, used for 4/4 is not well representative for the singing style differences.

A paired t-test between the baseline and each of Ex-1 and Ex-2 resulted in p-values of respectively 0.28 and 0.31 on total for all meter types. We expect that statistical significance can be evaluated more accurately with a bigger number of recordings.

6. CONCLUSIONS

In this paper we presented a Bayesian approach for tracking vocal onsets of singing voice in polyphonic music recordings. The main contribution is that we integrate in one coherent model two existing probabilistic approaches for different tasks: beat tracking and note transcription. Results confirm that the knowledge of the current position in the metrical cycle can improve the accuracy of vocal note onset detection over different metrical cycle types. The model has a comprehensive set of parameters, whose appropriate tuning allows application to material with different singing style and meter.

In the future the manual adjustment of these parameters could be replaced by learning their values from sufficiently big training data, which was not present for this study. In particular, the *lakh-vocal-segments* dataset could be easily extended substantially, which we plan to do in the future. Moreover, one could decrease the expected parameter values range, based on learnt values, and thus decrease the size of the state space, which is a current computational limitation. We believe that the proposed model could be applied as well to full-fledged transcription of singing voice.

[7] Per-recording results for the makam dataset are available at `https://tinyurl.com/y8r73zfh` and for the *lakh-vocal-segments* dataset at `https://tinyurl.com/y9a67p8u`

[8] We extended the python toolbox for beat tracking https://github.com/CPJKU/madmom/, which we make available at `https://github.com/georgid/madmom`

Acknowledgements We thank Sebastian Böck for the implementation hints. Ajay Srinivasamurthy is currently with the Idiap Research Institute, Martigny, Switzerland.

This work is partly supported by the European Research Council under the European Union's Seventh Framework Program, as part of the CompMusic project (ERC grant agreement 267583) and partly by the Spanish Ministry of Economy and Competitiveness, through the "María de Maeztu" Programme for Centres/Units of Excellence in R&D" (MDM-2015-0502).

7. REFERENCES

[1] Hasan Sercan Atlı, Burak Uyar, Sertan Şentürk, Barış Bozkurt, and Xavier Serra. Audio feature extraction for exploring Turkish makam music. In *Proceedings of 3rd International Conference on Audio Technologies for Music and Media (ATMM 2014)*, pages 142–153, Ankara, Turkey, 2014.

[2] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[3] Sungkyun Chang and Kyogu Lee. A pairwise approach to simultaneous onset/offset detection for singing voice using correntropy. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 629–633. IEEE, 2014.

[4] Norberto Degara, Antonio Pena, Matthew EP Davies, and Mark D Plumbley. Note onset detection using rhythmic structure. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5526–5529. IEEE, 2010.

[5] Masataka Goto. Singing information processing. In *12th International Conference on Signal Processing (ICSP)*, pages 2431–2438. IEEE, 2014.

[6] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 287–288, 2002.

[7] Andre Holzapfel. Relation between surface rhythm and rhythmic modes in turkish makam music. *Journal of New Music Research*, 44(1):25–38, 2015.

[8] Andre Holzapfel and Thomas Grill. Bayesian meter tracking on learned signal representations. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 262–268, 2016.

[9] Andre Holzapfel, Florian Krebs, and Ajay Srinivasamurthy. Tracking the "odd": Meter inference in a culturally diverse music corpus. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 425–430, Taipei, Taiwan, 2014.

[10] David Brian Huron. *Sweet anticipation: Music and the psychology of expectation*. MIT press, 2006.

[11] Florian Krebs, Sebastian Böck, and Gerhard Widmer. Rhythmic pattern modeling for beat and downbeat tracking in musical audio. In *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, Curitiba, Brazil, 2013.

[12] Florian Krebs, Sebastian Böck, and Gerhard Widmer. An Efficient State-Space Model for Joint Tempo and Meter Tracking. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 72–78, Malaga, Spain, October 2015.

[13] Nadine Kroher and Emilia Gómez. Automatic transcription of flamenco singing from polyphonic music recordings. *IEEE Transactions on Audio, Speech and Language Processing*, 24(5):901–913, 2016.

[14] Matthias Mauch, Chris Cannam, Rachel Bittner, George Fazekas, Justin Salamon, Jiajie Dai, Juan Bello, and Simon Dixon. Computer-aided melody note transcription using the tony software: Accuracy and efficiency. In *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR 2015)*, pages 23–30, 2015.

[15] Ryo Nishikimi, Eita Nakamura, Katsutoshi Itoyama, and Kazuyoshi Yoshii. Musical note estimation for F0 trajectories of singing voices based on a bayesian semi-beat-synchronous HMM. In *Proceedings of the 17th International Society for Music Information Retrieval Conference, (ISMIR 2016)*, pages 461–467, 2016.

[16] Lawrence Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[17] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

[18] Matti Ryynänen. Probabilistic modelling of note events in the transcription of monophonic melodies. Master's thesis, 2004.

[19] Justin Salamon and Emilia Gómez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1759–1770, 2012.

[20] Nick Whiteley, Ali Taylan Cemgil, and Simon Godsill. Bayesian modelling of temporal structure in musical audio. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, pages 29–34, Victoria, Canada, October 2006.

MINING LABELED DATA FROM WEB-SCALE COLLECTIONS FOR VOCAL ACTIVITY DETECTION IN MUSIC

Eric J. Humphrey[1] **Nicola Montecchio**[1] **Rachel Bittner**[1,2]
Andreas Jansson[1,3] **Tristan Jehan**[1]
[1] Spotify, New York, USA
[2] Music, Audio & Research Lab (MARL), New York University, USA
[3] City University, London, UK
{ejhumphrey, venice, rachelbittner, andreasj, tjehan}@spotify.com

ABSTRACT

This work demonstrates an approach to generating strongly labeled data for vocal activity detection by pairing instrumental versions of songs with their original mixes. Though such pairs are rare, we find ample instances in a massive music collection for training deep convolutional networks at this task, achieving state of the art performance with a fraction of the human effort required previously. Our error analysis reveals two notable insights: imperfect systems may exhibit better temporal precision than human annotators, and should be used to accelerate annotation; and, machine learning from mined data can reveal subtle biases in the data source, leading to a better understanding of the problem itself. We also discuss future directions for the design and evolution of benchmarking datasets to rigorously evaluate AI systems.

1. INTRODUCTION

Over the last few years, the ubiquity of cheap computational power and high quality open-source machine learning software toolkits has grown considerably. This trend underscores the fact that attaining state-of-the-art solutions via machine learning increasingly depends more on the availability of large quantities of data than the sophistication of the approach itself. Thus, when tackling less traditional or altogether novel problems, machine learning practitioners often choose between two paths to acquiring data: manually create (or curate) a dataset, or attempt to leverage existing resources.

Both approaches present unique challenges. Curation is necessary when precise information is required or insufficient data are available, but can incur large costs in both time and money. Alternatively, "mining" data – recovering useful information that occurs serendipitously in different contexts – can result in large datasets with far less effort, *e.g.,* recovering labels from the text around an image. While this information is typically generated as a by-product of some other pre-existing human activity and prone to both noise and bias, recent machine learning research has managed to use this approach to great effect [5].

With the continued growth of digital music services, vocal activity detection (VAD) is a task of increasing importance. Robust VAD is a key foundational technology that could power or simplify a number of end-user applications, such as vocalist similarity, music recommendation, artist identification, source separation, or lyrics transcription. Despite previous research, the state of the art continues to advance with diminishing returns, rendering VAD an unsolved problem with considerable potential.

Given the dominance of data-driven methods in machine learning, it stands to reason that data scarcity may be contributing to the apparent ceiling in the performance of VAD algorithms. Modest progress has been made toward increasing the size of labeled datasets, limiting the efficacy of modern approaches, *e.g.,* deep learning. Efforts to leverage strongly labeled datasets have converged to hundreds of observations [1,13,15,16], with which complex methods have been explored [9, 10, 18]. Recent research succeeded in curating a private dataset of $10k$, 30 second weakly labeled clips, *e.g.,* "completely instrumental" or "contains singing voice", using this dataset to train convolutional neural networks [17].

In short, VAD research remains largely dependent on sustained human involvement in sourcing labeled data, but this approach struggles to scale. Here, we propose leveraging a huge, untapped resource in modern music to circumvent this challenge: the "instrumental version", *i.e.,* a song in which the vocals have been omitted. The goal of this work is thus the exploration of this opportunity, achieved in four steps: mine original-instrumental pairs from a massive catalogue of music content; estimate time-varying vocal density given corresponding tracks; exploit this signal to train deep neural networks to detect singing voice; and understand the effects of this data source on the resulting models.

2. DATA GENERATION

In Western popular music, a song's arrangement often revolves around a lead vocalist, accompanied by instruments such as guitar, drums, bass, *etc.* It is not uncommon for an artist to *also* release an "instrumental" version of the same song (to be used for *e.g.,* remixes or karaoke), in which the primary difference between it and the corresponding "original" recording is the absence of vocals. [1] In principle, the difference between these two sound recordings should be highly correlated with vocal activity, which would provide a fine-grained signal for training machine learning models. However, to exploit this property at scale, it is necessary to identify and align pairs of original recordings and matching instrumental versions *automatically*.

We outline a three-step approach toward mining strongly labeled instances of singing voice from a music catalogue: identify original-instrumental pairs from track metadata; estimate a vocal density signal for the original track, given its instrumental; draw positive observations from an original track as a function of estimated vocal density.

2.1 Selection of Matching Recordings

We search the full Spotify catalogue, a set of tens of millions of commercially recorded tracks, for paired versions using a heuristic based on track metadata. A pair of tracks (A, B) are marked as (original, instrumental) if:

- A and B are recorded by the same artist.
- "instrumental" does not appear in the title of A.
- "instrumental" does appear in the title of B.
- The titles of A and B are *fuzzy matches*.
- The track durations differ by less than 10 seconds.

Fuzzy matching is performed on track titles by first latinizing non-ASCII characters, removing parenthesized text, and finally converting to lower-case; this yields about 164k instrumental tracks. Note that this is a many-to-many mapping, as an original version can point to several different instrumentals, and vice versa.

A tiny subset of this content is manually reviewed to check for quality, and we find roughly 1 in 10 tracks to be a mismatched pair: the majority of errors are due to instrumental tracks that appear on multiple albums, such as compilations or movie soundtracks, but are only tagged as such in some contexts. An open-source audio fingerprinting algorithm is used to remove suspect pairs from the candidate set [6]. Sequences of codes for tracks are extracted, and track pairs are discarded as a function of Jaccard similarity if code sequences do not overlap sufficiently (an erroneous fuzzy metadata match) or overlap too much (the tracks in the pair being both instrumental or vocal). Finally, redundant associations from this mapping are removed, so that each original track is linked to only one instrumental track. Overall this process yields roughly $24k$ tracks, or $12k$ original-instrumental pairs, totaling some 1500 hours of audio.

2.2 Estimating Vocal Density

Let T^O and T^I denote two recordings, corresponding to an "original" and "instrumental" version, respectively. A Time-Frequency Representation (TFR) is computed for both tracks, respectively X^O and X^I. Subsequently, the TFRs are aligned to estimate time-varying vocal density.

In this work a *Constant-Q Transform* (CQT) [3] is chosen for its complementary relationship between convolutional neural networks and music audio; the CQT uses a logarithmic frequency scale that linearizes pitch, allowing networks to learn pitch-invariant features as a result [8]. The frequency range of the transform is constrained to the human vocal range, *i.e.,* E2 - E7 (5 octaves, spanning 82.4-2637 Hz), and a moderately high resolution is employed, with 36 bins per octave and 32 frames per second. Logarithmic compression is applied pointwise to the TFR.

The pair of TFRs (X^O, X^I) undergoes a feature dimensionality reduction via Principal Component Analysis [2], producing (Z^O, Z^I); based on empirical findings, $k = 20$ components were found to yield good results. This step not only provides an increase in computational efficiency in subsequent processing steps, but also affords a useful degree of invariance because of the lower dimensionality.

The transformed sequences are then aligned using Dynamic Time Warping (DTW), yielding two sequences, n^O, n^I, of indices over the original and instrumental song, respectively [14]. This allows us to recover points in time from both a full and instrumental mix where the background musical content is roughly identical.

Using these indices, the CQT spectra (X^O, X^I) are resampled to equivalent shapes, and the half-wave rectified difference between log-magnitude spectra yields the following residual:

$$\delta_{j,k} = \max\left(0, \log|X^O_{n^O_{j,k}} + 1| - \log|X^I_{n^I_{j,k}} + 1|\right) \quad (1)$$

In the ideal case, where any difference is due entirely to vocals, this residual represents the vocal CQT spectra, and will behave like a smooth contour through successive time-frequency bins. Practically, however, there will likely be other sources of residual energy, due to suboptimal alignment or true signal differences. To best characterize contour-like residuals, we normalize the spectral energy in each time frame and apply the Viterbi algorithm to decode the most likely path through the residual spectra; this step is inspired by previous work on tracking fundamental frequency in a time-frequency activation map [12]. Empirically we find this process far more robust to residual noise than simpler aggregation schemes, such as summing energy over frequency.

[1] Though other differences in signal characteristics may occur due to production effects, *e.g.,* mastering, compression, equalization, these are not considered here.

[2] We are not interested in learning a general transform; the Principal Components of each pair of tracks are computed independently of the overall dataset.

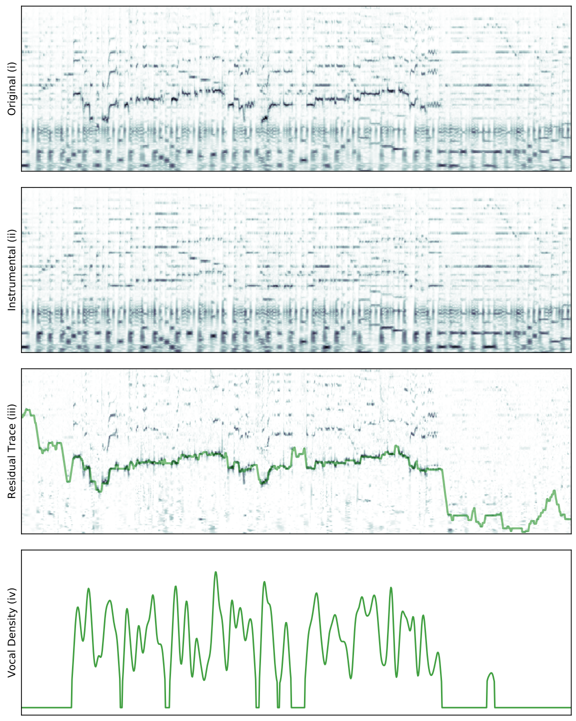


Figure 1. The intermediate stages in estimating vocal density from an original-instrumental pair of recordings, showing (i) the original and (ii) instrumental CQT spectra, (iii) the residual with a trace of its fundamental, and (iv) the estimated vocal density over time.

The amplitude of this time-frequency path, ρ, provides an estimate of vocal *density*, ϕ, the likelihood that vocals are present in the *original* recording, T^O, as a function of time. Finally, we forwards-backwards filter ϕ with a Hanning window ($L = 15$), to both smooth and dilate the density signal to encompass vocal onsets and offsets. The stages of this process are pictured in Figure 1.

2.3 Sampling of Positive and Negative Observations

Having estimated *where* vocals likely occur in a piece of audio, we turn our attention to *how* this information is utilized for supervised training. We highlight that track-level metadata presents a multiple-instance learning problem, where each recording can be understood as a *bag* of samples with a single binary label: "vocal" if it contains at least one positive sample, or "non-vocal" if all samples are negative. In this setting, positively labeled bags are inherently noisy, with some unknown percentage of negative samples effectively mislabeled as a result. To address this issue, the vocal density estimate is used to reweight the contributions of samples drawn from positive bags.

An estimator is trained by drawing positive ($Y = 1$) and negative ($Y = 0$) samples from original and instrumental tracks, with equal frequency. Negative samples are drawn uniformly from instrumental tracks, while positive samples are drawn as a function of the vocal density ϕ. To smoothly interpolate between a uniform distribution and the vocal density estimate over positive samples, two parameters are introduced, a threshold, τ, and a compression factor, ϵ:

$$Pr(X_n^O|Y = 1) \propto \begin{cases} \phi_n^\epsilon & \phi_n \geq \tau \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Here, we are interested in exponentials in the range of $0 < \epsilon < 1$, which flatten the density function. Note that $\epsilon = 0, \tau = 0$ corresponds to uniform sampling over time, and is equivalent to the weakly labeled setting, *i.e.,* a label applies equally to all samples.

As a final consideration, we highlight that the original and instrumental recordings are aligned in the course of computing the vocal density estimate. Therefore, it is possible to draw correlated positive-negative pairs from both the original and instrumental tracks corresponding to the same point in time, a sampling condition we refer to as *entanglement*, $\zeta \in \{True, False\}$. One would expect that these paired samples live near the decision boundary, being near-neighbors in the input space but belonging to different classes, and we are interested in exploring how training with entangled pairs may affect model behavior.

3. SYSTEM DESIGN

3.1 Previous Approaches

The majority of VAD research follows a similar architecture: short-time observations are fed to a classifier, each observation is assigned to either a vocal or a non-vocal class, and optionally post-processing is applied to eliminate spurious predictions. Early work uses "the acoustic classifier of a speech recognizer as a detector for speech-like sounds" to feed an Artificial Neural Network trained on a speech dataset (NIST Broadcast News) [1], while [16] attempts to explicitly exploit *vibrato* and *tremolo*, two characteristics that are specific of vocal signals. Alternativley, Support Vector Machines (SVMs) are used for frame classification and Hidden Markov Models act as smoothing step [15]; a similar solution is proposed by [13], which exploits a wider set of features, including ones derived from a predominant melody extraction step.

More recently, increasingly complex classifiers are preferred to feature engineering, given the widespread success of deep learning methods and modest increases in available training data. Much prior research explores the application of deep learning to music tagging, which typically encompasses one or more classes for singing voice in the taxonomy considered [7]. Elsewhere, deep networks have been used for pinpointing singing voice in source separation systems [19]. Regarding the particular task at hand, [9] proposes a sophisticated architecture based on Recurrent Neural Networks that does not have a separate smoothing step, while [17] uses a conventional convolutional network topology, further advancing the state of the art.

3.2 Proposed System

The log-magnitude CQT representation described in 2.2 is processed in 1 second windows, with a dimensionality of (32×180) bins in time and frequency, respectively. We adopt a five-layer neural network, with three (3D) convolutional layers, each followed by max-pooling, and two fully-connected layers, with the following parameter shapes: $w_0 = (1, 64, 5, 13), p_0 = (2, 3)$, $w_1 = (64, 32, 3, 9), p_1 = (2, 2), w_2 = (32, 24, 5, 1), p_2 = (2, 1)$, $w_3 = (1540, 768), w_4 = (768, 2)$. All layer activations are hard rectified linear units (ReLUs), with the exception of the last (classifer) layer, which uses a softmax.

The network is trained using a negative log-likelihood loss function and parameters are optimized with minibatch stochastic gradient descent. We implement our model in *Theano*[3], leveraging the *Pescador*[4] package for drawing samples from our dataset, and accelerate training with a NVIDIA Titan X GPU. Networks are trained for 500k iterations ($\approx$ 20 hours) with a learning rate of 0.05 and a batch size of 50. Dropout is used in all but the last layer, with a parameter of 0.125. In addition to the weakly labeled case, $\{\epsilon = 0.0, \tau = 0.0, \zeta = F\}$, we explore model behavior over two sampling parameter settings, with and without entanglement: $\{\epsilon = 0.3, \tau = 0.05\}$ and $\{\epsilon = 1.0, \tau = 0.2\}$. These values are informed by first computing a histogram of vocal activation signals over the collection, revealing that a large number of values occur near zero ($\leq$ 0.05), while the upper bound rolls off smoothly at $\approx$ 2.5.

4. EXPERIMENTAL RESULTS

We evaluate our models on two standard datasets in VAD research: the *Jamendo* collection, containing 93 manually annotated songs [15]; and the *RWC-Pop* collection, containing 100 manually annotated songs [13]. To compare with previously reported results, we consider the area under the curve (AUC) score and max-accuracy [17]. The AUC score provides insight into the rank ordering of class likelihoods, and max-accuracy indicates the performance ceiling (or error floor) given an optimal threshold.

4.1 Quantitative Evaluation

Table 1 shows the summary statistics over the two datasets considered as a function of sampling parameters, alongside previously reported results for comparison [17]. The first three systems (α, β, γ) are successive boosted versions of each other, *i.e.,* α is trained with weak labels, and its predictions on the training set are used to train β, and so on; the *fine* model is trained directly with strongly labeled data, and we refer to each by suffix, *e.g.,* α. Additionally, the authors train these models with a witheld dataset unavailable to our work here.

These results provide a few notable insights. First, we confirm that our automated approach of mining training data is sufficient to train models that can match state of the art performance. Configuration `I`, corresponding to the weak labeling condition, performs roughly on par with a comparably trained system, α, validating previous results; configuration `V` achieves the best scores of our models, and

				RWC		JAMENDO	
				AUC	ACC$^+$	AUC	ACC$^+$
SCHLÜTER-α				0.879	0.856	0.913	0.865
SCHLÜTER-β				0.890	0.861	0.923	0.875
SCHLÜTER-γ				0.939	0.887	**0.960**	**0.901**
SCHLÜTER-FINE				**0.947**	0.882	0.951	0.880
	τ	ϵ	γ				
I	0.0	0.0	F	0.891	0.856	0.911	0.856
II	0.05	0.3	F	0.918	0.879	0.925	0.869
III	0.05	0.3	T	0.918	0.879	0.934	0.874
IV	0.2	1.0	F	0.937	0.887	0.935	0.872
V	0.2	1.0	T	0.939	**0.890**	0.939	0.878

Table 1. AUC-scores and maximum accuracies across models on the *RWC* and *Jamendo* datasets.

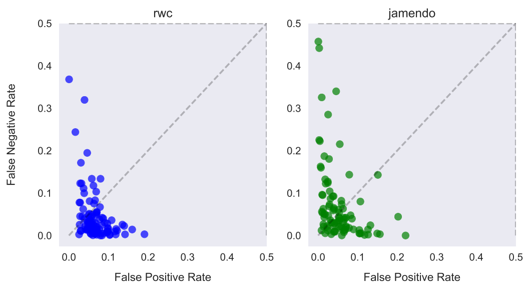


Figure 2. Trackwise error rates, plotting false positives versus false negatives for IV; one outlier ($fn \approx$ 0.66) in the *Jamendo* set is not shown to maintain aspect ratio.

is consistent with gains in prior work. That said, the difference between models is in the range of 0.02-0.05 across metrics, which is of limited reliability with datasets of this size. In terms of sampling parameters, we observe a direct correlation between signal-to-noise ratio in our training data, *i.e.,* the more non-vocal observations are discarded, the better the models behave on these measures. Training with entangled pairs ($\zeta = T$) also seems to have a small positive effect. Finally, we note a possible corpus effect between these systems and previously reported results, where models (V and γ) perform better on different data. Though minor, this potential corpus effect serves as a dimension to explore in subsequent analysis.

4.2 Error Analysis

As the systems reported here are high performing, a potentially more informative path to understanding model behavior is through analyzing the errors made. Here, false positives occur when a different sound source is mistaken for voice; false negatives occur when the energy of a vocal source has fallen below the model's sensitivity. Observations drawn from the same music recording will be highly correlated, due to the repetitive nature of music, and so we explore the track-wise frequency of errors to identify behaviors that may reveal broader trends.

A slight corpus effect is seen in Figure 2 between the *RWC* and *Jamendo* collections. In the former, the majority

[3] `https://github.com/Theano/Theano`
[4] `https://github.com/pescadores/pescador`

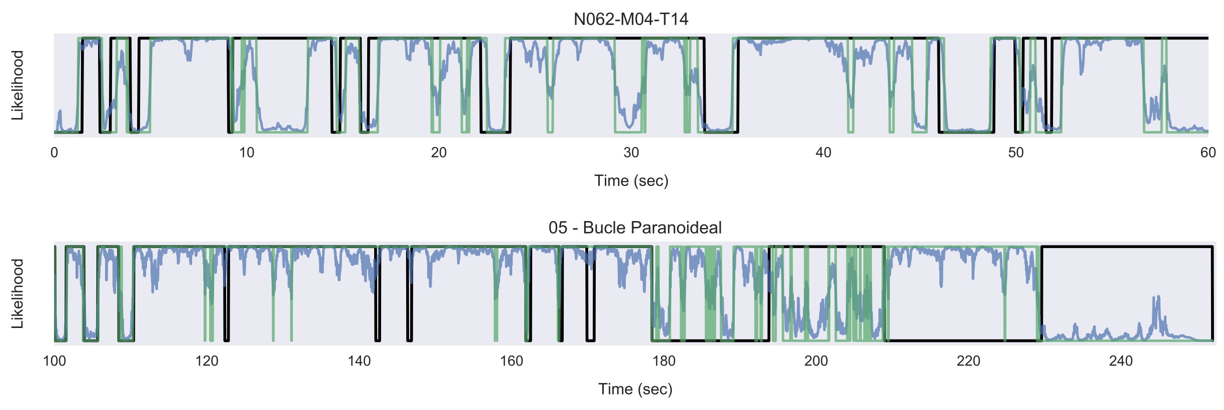


Figure 3. Examples from the evaluation dataset, showing the ground truth (black), estimated likelihoods (blue) and thresholded prediction (green) over time: (top) a track from the *RWC* corpus demonstrates how a model can operate with greater temporal precision than a human annotator, a common source of false negatives; (bottom) a track from the *Jamendo* collection illustrates different challenges, including imposter sources (a guitar solo), sensitivity to vocals, and annotator error.

of error is due to false positives, but at a much lower rate of occurrence ($fp < 0.1$) than false negatives. Additionally, when errors do occur in a track, they tend to be primarily of one type, and seldom both. This is less the case for the *Jamendo* set, comprised of both "worse" tracks and a (slightly) larger co-occurrence of error types in a given track.

Using this visualization of trackwise errors, an investigation into the various outliers yields a few observations. There are two primary sources of false negatives: one, shown in Figure 3 (top), trained models exhibit a level of temporal precision beyond the annotators' in either dataset, pinpointing breaths and pauses in otherwise continuous vocalizations; and two, nuances of the data used for training seem to induce a production bias, whereby the model under-predicts singing voice in lower quality mixes. In hindsight, it is unsurprising that models trained on professionally produced music might develop a sensitivity to mixing quality, and we note this as a topic for future exploration. A similar bias also appears to account for the majority of all false positives, often corresponding with monophonic instrumental melodies, *e.g.,* guitar riffs or solos, but less so for *polyphonic* melodies, *i.e.,* two or more notes played simultaneously by the same source, consistent with previous findings [11].

Figure 3 (bottom) illustrates an interesting example of this behavior. In the first 80 seconds shown here, the model agrees with the human annotation. The model fails at the 180 second mark, misclassifying a guitar line, and continues through 194-210 seconds, where the model struggles to detect rap vocals at a softer volume. However, from that point onwards, the human annotation itself is wrong, while the model is correct; vocals are indeed present between 210-230 seconds and 230-252 contains no voice, which accounts for 16% of the track. Coincidentally, this further underscores the need for large, diverse evaluation datasets to produce reliable metrics.

4.3 Multitrack Analysis

The above results confirm the intuition that it can be challenging to manually annotate singing voice activity with machine precision. Ideally, though, human annotation approximates a smoothed, thresholded version of the vocal signal energy in isolation, and as such, we are interested understanding the degree to which model estimations correspond with the pure vocal signal. Another way of measuring our models' capacity to estimate singing voice from a "down-mixed" recording is through the use of multitrack audio, which provides direct access to the signal of interest, *i.e.,* vocals, in isolation.

We now turn our attention to MedleyDB, a dataset of 122 songs containing recordings of individual stems and corresponding mixes [2]. For each of the 47 songs that have vocals in isolation, we create a single vocal track for analysis, and compute the log-magnitude CQT for the full mix (the "original" version) X^M, and the isolated vocals, X^V. Whereas previously Viterbi was used to track vocal density, here the reference vocal density contains no noise and can be computed by summing the spectral energy over frequency, *i.e.,* $\phi_n^V = \sum_k X_{n,k}^V$. The trained models are applied to the full mix, X^M, for inference, producing a time-varying likelihood, L^M.

The reference vocal density is not a class label but a continuous value, and the comparison metrics must be adjusted accordingly. Maximum accuracy is generalized to the case where independent thresholds are considered for ϕ^V, L^M over the dataset, providing insight into the best-case agreement between the two signals. We also consider the Spearman rank-order correlation between the two sets, a measure of the relative rank order between distributions [20].

An exploration of model performance on this dataset validates earlier observations, summarized in Table 2. On manual inspection of the temporal precision of the model on the Medley dataset, we confirm that deviations between estimated likelihoods and the reference vocal density are

	$\{\tau, \epsilon, \zeta\}$	SPEARMAN-R	ACC$^+$
I	0.0, 0.0, F	0.681	0.812
II	0.05, 0.3, F	0.779	0.849
III	0.05, 0.3, T	0.768	0.854
IV	0.2, 1.0, F	0.784	0.852
V	0.2, 1.0, T	**0.796**	**0.862**

Table 2. Spearman rank-order correlation and maximum accuracy scores across models on the MedleyDB vocal subset.

representative of true model errors, setting a baseline for future work. As seen previously, false negatives again correspond to vocal loudness relative to the mix, and false positives are caused by loud melodic contours. Note also that the Spearman rank-order correlation is consistent with previously observed trends across models, while providing more nuance; the greatest difference between models is > 0.11, versus ≈ 0.05 for maximum accuracy. Finally, we note that the flexibility of multitrack datasets presents a great opportunity for rigorously testing future work, whereby the pitch and loudness of a vocal track can be used to synthesize "imposters" with different timbres, *e.g.,* a sine wave or flute, mixed with instrumental tracks, and used to measure false positives.

5. DISCUSSION

This work presents an approach to mining strongly labeled data from web-scale music collections for detecting vocal activity. Original recordings, containing vocals, are automatically paired with their instrumental counterparts, and differential information is used to estimate vocal activity over time. This signal can be used to train convolutional neural networks; the strongly labeled training data produces superior results to the weakly labeled setting, achieving state of the art performance. While analyzing errors made by our models, three distinct lessons stood out.

First, in addition to curation and mining, it is valuable to recall a third path to acquiring sufficiently large datasets: active learning. Imperfect models can be leveraged to make the annotation process more efficient by performing aspects of annotation that humans may find difficult or time-consuming, as well as prioritizing data as a function of model uncertainty. Here, for example, we observe that regions annotated as vocal tend to include brief pauses, no doubt resulting from the time and effort it would require to annotate at that level of detail. Alternatively, a performant model, like those described here, could segment audio into short, labeled excerpts for a human to verify or correct, eliminating a huge time cost. This would allow reliable data to be obtained at a faster rate.

Second, the application of machine learning to mined datasets can help identify particular challenges of a given task, unlocking new research directions. Here, our model identifies an interesting bias in the dataset that we had not previously considered, being the tight coupling between singing voice (timbre), melody (pitch), and production effects (loudness). Often in Western popular music, lead vocals carry the melody and tend to be one of the more prominent sources in the mix. Thus, in the dataset mined from a commercial music catalogue, instrumental versions not only lack vocal timbres, but prominent melodic contours are missing as well. This complex relationship is less obvious at a distance, but our experiments illustrate the challenges faced data-driven approaches to singing voice detection. By the same token, this also identifies an opportunity to build systems invariant to these dimensions.

Finally, these insights serve as a reminder that it is good practice to both design and evolve benchmarking datasets to encompass challenging test cases and known failure modes as they are identified. In our analysis, we find that the available benchmarking datasets consist mostly of musical content in which the melody is also voice, and therefore more "difficult" signals would help reliably discriminate between models. This content could be identified automatically via incremental evaluation methods, in which disagreement between machine estimations effectively prioritizes data to maximize discrimination between models [4].

5.1 Future Work

Perhaps the most logical next step for this work is to better augment training data, such that pitch and melodic information are well represented in negative examples. One possible approach, for example, would be to use the frequency of the vocal density estimate recovered in Section **??** to synthesize the melody with different timbres to be mixed into the instrumental recording. Whereas before entangled pairs contrast the presence of vocals, this approach would yield pairs that differ only in the timbre of the voice. Alternatively, additional sources could be leveraged for building models invariant to less relevant characteristics, such as instrumental content without a corresponding "original" version, or multitrack audio.

Additionally, more effort is required to advance evaluation methodology for automatic vocal activity detection. Multitrack datasets like MedleyDB, are a particularly promising route for rigorous benchmarking. The isolated vocal signal provides an optimal reference signal, while the other, non-vocal stems can be recombined as needed to deeply explore system behavior. We also recognize that larger, more diverse evaluation datasets are a prerequisite to advancing the state of the art in this domain. Thus, as a first step toward these ends, we provide machine estimations from our model over the datasets used here, as well as a large publicly available dataset (with audio) to facilitate the manual annotation process.[5] Though human effort is necessary to verify or correct these machine estimations, we share this data in the hope that it can serve as a starting point to accelerate the growth of labeled data for this task and facilitate efforts toward incremental evaluation.

[5] `https://github.com/ejhumphrey/vox-detect-jams`

6. REFERENCES

[1] Adam L Berenzweig and Daniel PW Ellis. Locating singing voice segments within music signals. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 119–122. IEEE, 2001.

[2] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, volume 14, pages 155–160, 2014.

[3] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[4] Ben Carterette and James Allan. Incremental test collections. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 680–687. ACM, 2005.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] Daniel PW Ellis, Brian Whitman, and Alastair Porter. Echoprint: An open music identification service. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR, 2011.

[7] Philippe Hamel, Simon Lemieux, Yoshua Bengio, and Douglas 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)*, pages 729–734, 2011.

[8] Eric J Humphrey and Juan Pablo Bello. Rethinking automatic chord recognition with convolutional neural networks. In *International Conference on Machine Learning and Applications (ICMLA)*, volume 2, pages 357–362. IEEE, 2012.

[9] Simon Leglaive, Romain Hennequin, and Roland Badeau. Singing voice detection with deep recurrent neural networks. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–125. IEEE, 2015.

[10] Bernhard Lehner, Gerhard Widmer, and Sebastian Bock. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. In *Proceedings of the 23rd European Signal Processing Conference (EUSIPCO)*, pages 21–25. IEEE, 2015.

[11] Bernhard Lehner, Gerhard Widmer, and Reinhard Sonnleitner. On the reduction of false positives in singing voice detection. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7480–7484. IEEE, 2014.

[12] Matthias Mauch and Simon Dixon. pYIN: A fundamental frequency estimator using probabilistic threshold distributions. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 659–663. IEEE, 2014.

[13] Matthias Mauch, Hiromasa Fujihara, Kazuyoshi Yoshii, and Masataka Goto. Timbre and melody features for the recognition of vocal activity and instrumental solos in polyphonic music. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, pages 233–238, 2011.

[14] Colin Raffel and Daniel PW Ellis. Large-scale content-based matching of MIDI and audio files. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*. ISMIR, 2015.

[15] Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal detection in music with support vector machines. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1885–1888. IEEE, 2008.

[16] Lise Regnier and Geoffroy Peeters. Singing voice detection in music tracks using direct voice vibrato detection. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1685–1688. IEEE, 2009.

[17] Jan Schlüter. Learning to pinpoint singing voice from weakly labeled examples. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[18] Jan Schlüter and Thomas 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)*, pages 121–126, 2015.

[19] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. Deep Karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *Latent Variable Analysis and Signal Separation, International Conference on*, pages 429–436. Springer, 2015.

[20] D. Zwillinger and S. Kokoska, editors. *Probability and Statistics Tables and Formulae.* Chapman & Hall, New York, NY, 2000.

MULTI-PART PATTERN ANALYSIS: COMBINING STRUCTURE ANALYSIS AND SOURCE SEPARATION TO DISCOVER INTRA-PART REPEATED SEQUENCES

Jordan B. L. Smith **Masataka Goto**
National Institute of Advanced Industrial Science and Technology (AIST), Japan
`jordan.smith@aist.go.jp, m.goto@aist.go.jp`

ABSTRACT

Structure is usually estimated as a single-level phenomenon with full-texture repeats and homogeneous sections. However, structure is actually multi-dimensional: in a typical piece of music, individual instrument parts can repeat themselves in independent ways, and sections can be homogeneous with respect to several parts or only one part. We propose a novel MIR task, multi-part pattern analysis, that requires the discovery of repeated patterns within instrument parts. To discover repeated patterns in individual voices, we propose an algorithm that applies source separation and then tailors the structure analysis to each estimated source, using a novel technique to resolve transitivity errors. Creating ground truth for this task by hand would be infeasible for a large corpus, so we generate a synthetic corpus from MIDI files. We synthesize audio and produce measure-by-measure descriptions of which instruments are active and which repeat themselves exactly. Lastly, we present a set of appropriate evaluation metrics, and use them to compare our approach to a set of baselines.

1. INTRODUCTION

Music structure is important to listeners and researchers, but annotating music is hard because typical songs include multiple independent instrument parts. For example, if two sections share the same basic melody, but one features an extra horn part, should one section be labeled as a repetition of the other? To decide, the annotator must consider all the ways in which the two sections are similar or different, but the outcome of their decision is encoded in a single bit: whether the label is the same or not. The annotation discards many of the decisions made by the listener, especially when these are made at the timescale of entire sections. For example, the second verse of Oasis' "Wonderwall" has the same chords and melody as the first, but different lyrics, and it includes two new instruments, cello and drums—the latter of which enters a measure late. A single large-scale section label cannot encode this interesting situation.

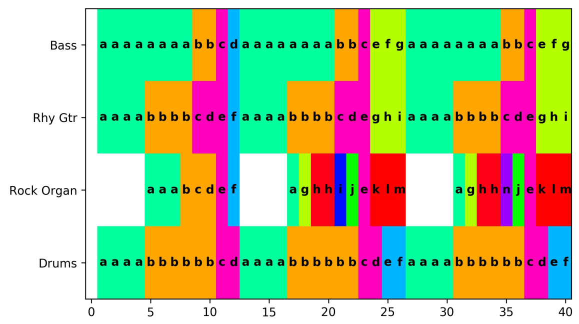


Figure 1. Example multi-part pattern description for the first 40 measures of"Come Together". Measures that repeat are given the same letter label. In this and later figures, the colors highlight repeated sequences instead of individual labels: if label i is always followed by j, and j always follows i, their color assignments are merged.

The multi-dimensional nature of structure has been commented on [22], and recent corpora of annotations have addressed it in different ways: the SALAMI dataset provides descriptions at two timescales, and of functions and leading instrument [29], and the INRIA dataset describes how segments and their component patterns are hierarchically related [2]. For music cognition research, [26] suggested that music be annotated multiple times on a per-feature basis: e.g., once while focusing only on harmony, again while focusing on timbre, and so forth. However, the challenge of hierarchy is different from the challenge of multiple independent parts. We argue that estimating the structure of these independent parts—i.e., creating a multi-part pattern analysis—should be a new MIR goal.

An example of a multi-part pattern description is shown in Fig. 1. It is derived from a MIDI transcription of "Come Together" by The Beatles from the Lakh dataset [23]. It indicates whenever an instrument in the mixture repeats itself, at the timescale of measures. This representation makes clear that the organ part (here substituting the lead vocals) is varied in the second verse, while the other instruments repeat themselves exactly. Compared to a one-dimensional structural analysis, the richer detail of a multi-part description would be more suitable for applications like automatically editing videos or choreographies to match an audio file.

We make four main contributions in this work. First, we define a new goal for MIR research; second, we propose an algorithm for accomplishing it, which uses existing technology and some new techniques; third, we propose an evaluation framework for the task, including metrics, baselines, and how to obtain ground truth; and finally, we conduct an evaluation.

In the next section, we discuss how our proposed task relates to existing MIR tasks. We present our algorithm in Section 3, present the evaluation framework in Section 4, and discuss the results in Section 5.

2. RELATED WORK

Identifying repeating motives has long been of interest to musicologists in MIR. Although most research in this area has focused on symbolic data analysis (see, e.g., [5]), when "Discovery of Repeated Themes" was added to MIREX in 2013, it included both symbolic and audio tracks (e.g., [21])—but the focus of that task is different: in it, the challenge is precisely to ignore the differences between instruments (if the piece being analyzed contains multiple parts) as well as, potentially, to ignore differences in key or modality. Our task, multi-part pattern analysis, involves a separate challenge: discovering repetitions expressed by a single voice within the mixture.

Since it involves describing the independent patterns in a mixture of tracks, the task is clearly related to source separation. Recently, approaches to source separation have become more structural, taking better advantage of the redundancies offered by repetition in music. One common technique, non-negative matrix factorization (NMF), separates sources by modeling steady states in the spectrum; an extended version, NMF decomposition (NMFD), models short sequences that are time-varying but exactly-repeating [28], and NMF was recently used to detect long loops [15]. Median filtering, which was used to efficiently perform harmonic-percussive source separation (HPSS) [6], was used in the REPET algorithm to separate a repeating background from a mixture [14]; REPET was later adapted to looping backgrounds that change over time and heterogenous backgrounds [25]. Although estimating a multi-part pattern analysis will *require* source separation, the desired output is an abstract description, not a set of separated tracks. Thus, whereas a source separation is evaluated with signal reconstruction error, a pattern analysis will be evaluated more like a structural analysis.

As for structural analysis, it has evolved toward modeling hierarchy. Early segmentation-only approaches [7] were followed by approaches that also estimate labels [8], and by approaches that model similarity differently at different timescales [11]. Since the creation of the multi-scale SALAMI and INRIA annotations, approaches to hierarchical description have been refined [17], as has the methodology for evaluating them [18]. Hierarchy is partly a consequence of multiple sources behaving independently: three repetitions of the chorus could be considered the same at a coarse timescale (the context of the song), but differences in range or instrumentation could differentiate them at a finer timescale (the context of the three choruses). Models of hierarchy will always be ambiguous, since its perception is ambiguous [12]. In contrast, the multi-layered composition of a song can be described more concretely. Thus, multi-part pattern analysis is worth treating separately from hierarchical structure, and a good multi-part analysis may be very useful for describing hierarchy.

Finally, two works have directly bridged source separation and structural analysis: First, [10] found that structure analysis could be performed more accurately with multi-track audio as input. Second, [27] discovered that spikes in the reconstruction error of a source separation algorithm can indicate structural boundaries. In defining the task of multi-part pattern description, we hope to bring these fields closer together.

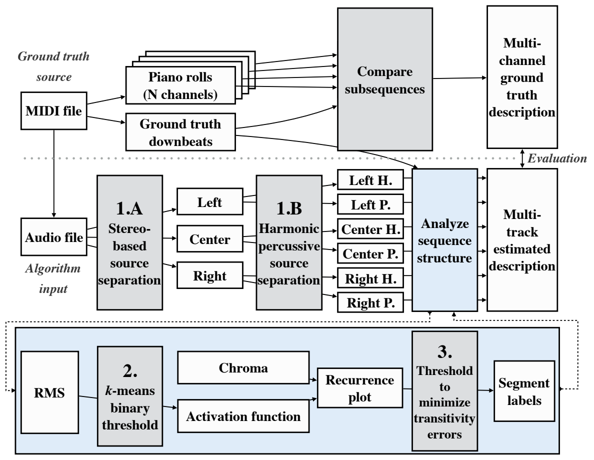


Figure 2. Algorithm and ground truth generation pipeline.

3. PROPOSED APPROACH

Our proposed algorithm is outlined in Fig. 2, and data at certain intermediate steps are illustrated in Fig. 3. The three key stages of the algorithm are:

1. Source separation. We apply source separation to the audio to convert the stereo recording to an estimated multi-track recording. We do this with two median spectral filters [13]: first, we take the median of the left and right channels to estimate the center channel, and subtract this from the original signals, resulting in three tracks. Second, we apply HPSS to each track [6]. Even if a track contains multiple pitched instruments, HPSS can separate instruments with different attacks, such as piano vs. strings, or rhythm guitar vs. organ. We end up with 6 audio tracks (see Fig. 3b).

2. Activation function estimation. The separated tracks may be sparse: e.g., if the left channel contains only strings, the left-percussive component may be nearly empty. We compute RMS to estimate when the channels are active. At this stage, we also use the ground truth downbeat labels to define our segment windows. In future

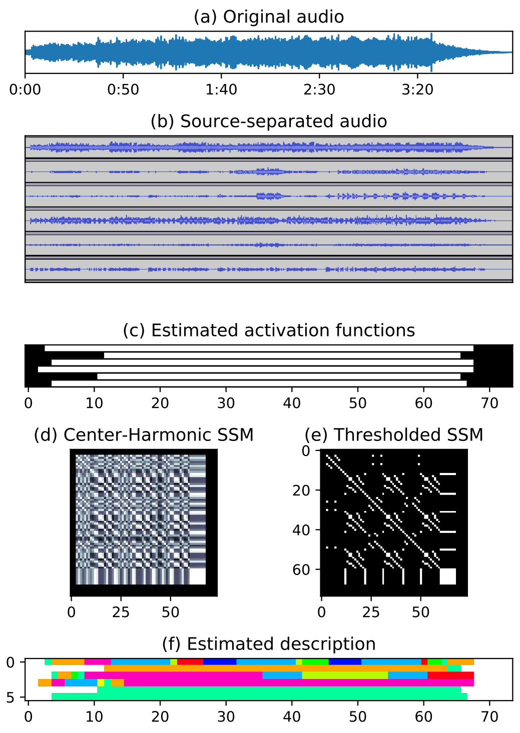


Figure 3. Data in intermediate stages of algorithm pipeline. The SSMs in plots (d) and (e) correspond to the center-harmonic track, which is the first track in plots (b) and (c). Sound example is "Across The Universe."

work, beats and downbeats could be estimated instead.

We take the mean over each window (i.e., each measure), and apply a k-means clustering to the RMS values, with $k = 2$, to classify windows as either silent or active. Even if the classes are very uneven, the difference between the two with respect to RMS tends to be extreme enough that this method is effective. At the end of this stage we have a set of estimated activation functions (see Fig. 3c).

3. Sequence analysis. We use self-similarity matrices (SSMs) to discover repetitions in each track. We compute chroma with the madmom package [4] and compute a measure-indexed SSM: element i, j gives the cosine similarity between the sequences of beat-synchronized chroma features of the i^{th} and j^{th} measures. We also use the previously-estimated activation functions to zero out the SSM when the track was judged inactive, as shown at the beginning and end of the track in Fig. 3d.

To estimate segment labels from the real-valued SSM, we choose a threshold t to binarize the matrix; then, to emphasize diagonal lines, we apply a single erosion-dilation operation (in time-lag space) with a kernal size k. We choose t and k in a novel way: my finding the values that minimize the number of transitivity errors. These errors are resolved with a novel lexical-sort approach. Transitivity errors are cases where a segment i is judged to be similar to both j and k, but segments j and k are not similar to each other; resolving these inconsistencies is a difficult part of interpreting structure from SSMs (e.g., see [20]).

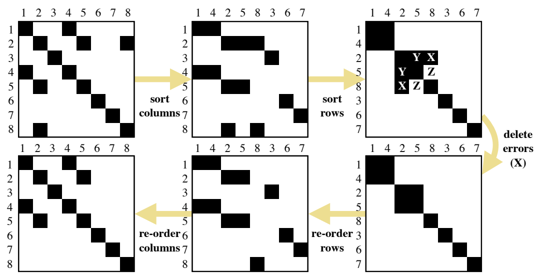


Figure 4. Eliminating transitivity errors with lexical sorting. Errors appear as inconsistent blocks in the sorted SSM. We can fix the error by eliminating pixels X, or pixels Y, or adding pixels at Z.

Given a binary SSM, we can collect repeating pixels into groups by sorting the rows lexically (i.e., alphabetically). The process is illustrated in Fig. 4: after sorting the SSM's rows and columns, groups of repeating elements become blocks on the main diagonal, and all other pixels represent transitivity errors. The third SSM in Fig. 4 can be fixed in three ways: zeroing the pixels at X, or at Y, or adding pixels at Z. We greedily eliminate the errors by walking along the main diagonal from the upper left and discarding off-diagonal elements that do not fit the current block, which corresponds to zeroing X. When the cleaned SSM is re-ordered, the result is guaranteed to be transitive.

We call the number of pixels deleted from an SSM the "strain", and the number of off-diagonal pixels that remain the "coverage". (For the example in Fig. 4, strain is 2 and coverage is 4.) Our goal is to choose t and k to maximize coverage and minimize strain, while avoiding redundant cases such as an empty SSM or an SSM that is all ones.

We sweep values of t between 0.99 and 0.8, and k between 4 and 8 measures. A set of real-world examples are shown in Fig. 5. The left column contains 5 binary SSMs derived from chroma computed on an audio track. The second and third columns give the lexically-ordered SSMs (and their strains) and their cleaned versions (and coverages). The fourth column gives the cleaned SSMs and the difference between coverage and strain, which is maximized by choosing $k = 7$. The result is a binary SSM that is sparse but not empty, and free of transitivity errors, as in Fig. 3e. It is then trivial to label the segments. The six estimated part descriptions are collected in Fig. 3f.

In structure analysis, we typically search for long repeating subsequences and long homogeneous stretches, and apply strong smoothing to the SSM to gloss over variations. In contrast, the above pipeline was designed to focus on tracking shorter patterns and to find when they repeat exactly, with the expectation of obtaining a much sparser SSM with few transitivity errors.

4. EVALUATION FRAMEWORK

4.1 Data and Ground Truth

To test the quality of a multi-part pattern analysis algorithm, we need audio files with multiple layers, with each

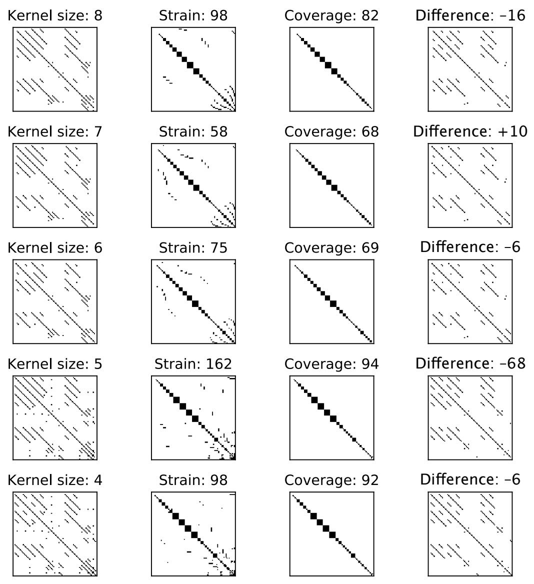


Figure 5. Illustration of strain-coverage optimization approach on a track estimated from a recording of "All My Loving." The four columns, from left, give: (1) binary SSMs filtered with different kernel sizes; (2) lexically sorted SSMs; (3) SSMs with errors removed; (4) cleaned SSMS restored to original column and row order. Kernel size 7 maximizes coverage while minimizing strain.

layer annotated to indicate repeating patterns. Creating ground truth for this task by hand would be infeasible for a large corpus. There are many public datasets of multitrack audio, but only rarely are the tracks annotated in detail. The existing dataset that most closely meets our needs is MedleyDB [3], which contains multitrack audio and melody f0 annotations for a subset of stems, but not annotations of repetitions in each track.

However, we can generate an appropriate dataset from MIDI. We used a portion of Lakh MIDI dataset [23] called the "Clean MIDI subset", which contains most of the Beatles catalogue, and used FluidSynth [1] to convert these to audio files. When there were duplicate MIDI files to choose from, we selected the version where the average panning setting of the tracks had the highest standard deviation. (Many MIDI transcriptions have no panning information at all, which would work against our algorithm.)

We processed the MIDI files (using Pretty MIDI [24]) to create, for each MIDI channel, a ground truth description of the measure-level patterns. The procedure for this is similar to our analysis algorithm (see Fig. 3). From a downbeat-segmented piano roll (Fig. 6a), we obtain an activation pattern, i.e., a timeline of 1s and 0s indicating whether an instrument has any MIDI note events during each measure-long window (Fig. 6b). Next, we estimate the similarity of every pair of measures within a track with an SSM (Fig. 6c). To compare two piano roll windows, we take the percentage of active note spans that overlap. To focus on exact repetitions, we should use a threshold of

[1] `http://www.fluidsynth.org/`

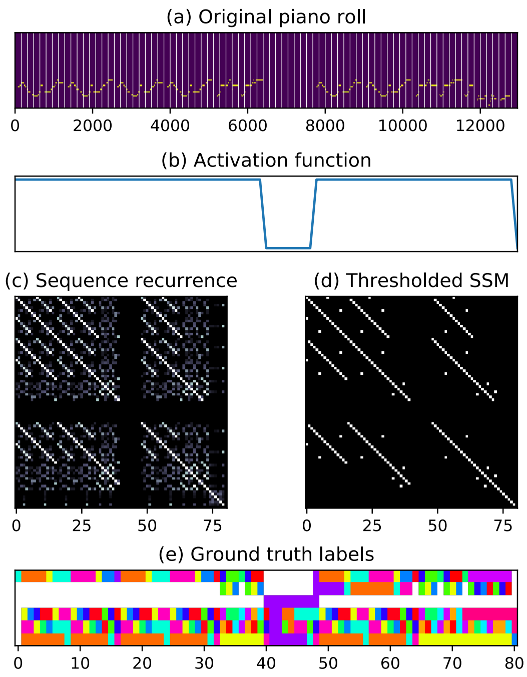


Figure 6. Data in intermediate stages of ground truth generation for the vocal channel of "All My Loving." The song's multi-part description is shown in (e).

1.0, but in practice, due to small timing differences and expressive gestures in the MIDI transcription, a threshold of 1.0 leads to extremely sparse recurrence plots—but on the other hand, lowering the threshold can lead to transitivity errors, as before. However, we found that a threshold of 0.9 was generally suitable to obtain non-empty recurrence plots without producing a large number of transitivity errors (Fig. 6d). Doing this for every track gives a multi-part description (Fig. 6e).

4.2 Evaluation Metrics

After processing the MIDI data, we obtain a "ground truth" matrix of instrument patterns A where the element $A_{i,j}$ indicates the pattern label for the i^{th} instrument during the j^{th} measure. (Such information is displayed in Fig. 1 and Fig. 6e.) We set $A_{i,j} = 0$ when the i^{th} instrument is not active. Similarly, we can obtain an estimated description E with elements $E_{i,j}$, such as in Fig. 3f.

To compare two single-track descriptions (two rows of A and E), we can use any metrics from the field of structure analysis, such as the pairwise f-measure metric [19]. (For a comparison of structure evaluation metrics, see [16].) However, the rows of A and E are not necessarily aligned in the correct order. Moreover, the number of estimated tracks in E may be smaller or greater than the number of MIDI channels in A. We present two sets of evaluation metrics: one that requires matching the layers, and one that does not. We also devise a set of baselines.

Evaluation of descriptions. Suppose we have an N-layer estimated description and an M-layer ground truth

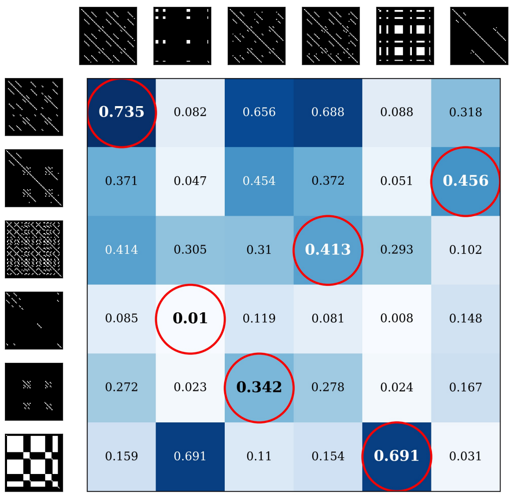


Figure 7. Optimal pairing of estimated tracks (columns) with ground truth channels (rows), according to pairwise f-measure between descriptions (which are illustrated as recurrence plots) The mean pw_f is 0.44.

description, and let $L = \min(M, N)$. We can compute the pairwise f-measure between all pairs of layers, giving a table of values like the one in Fig. 7. The Hungarian Algorithm[2] gives us the optimal one-to-one matching between L layers to maximize the average f-measure. Using the optimal pairings, we can compute the average precision and recall. Together, these serve as our set of 3 "generous" metrics, since it does not punish cases when $N \neq M$. If there are unmatched layers, whether in A or E, these should count against the estimate. We can compute a stricter mean f-measure by taking $pw_f * L/\max(M, N)$.

Evaluation of activations. The activation matrix that we estimate (e.g., Fig. 3c) is an important intermediate step. It is worth evaluating on its own, and we can do so without matching tracks to channels. We treat each column of the activation matrix, an N-dimensional binary vector, as a 'timbre label,' such that each unique column gets a unique label. (This calls to mind the timbre-mixture estimation of [1].) We perform the same process on the ground truth activation matrix. Then we can use pairwise f-measure to compare the two sequences of timbre labels.

This metric ignores the difference between an instrument being added to or subtracted from the mixture. To evaluate the retrieval of entrances and exits, we use a version of the boundary retrieval f-measure [19, p. 220], counting each entrance (or exit) in the ground truth as being correctly estimated only if some instrument in the estimated description also enters (or exits, respectively) in the same measure.

4.3 Baseline Methods

We compare our algorithm against a set of naive baselines to gauge the success of our algorithm, but also to learn how the proposed evaluation metrics behave. The labeling baselines are:

- $B_{constant}$: all measures repeat the same pattern;
- B_{null}: all measures are unique;
- $B_{periodic}$: there are three concurrent tracks playing sequence loops of length 2, 4 and 8 measures: i.e., three sequences $[ab]*$ (i.e., ab repeated), $[abcd]*$, and $[abcdefgh]*$;
- B_{block}: there are three concurrent tracks that alternate static textures with periods 2, 4 and 8 measures: i.e., $[ab]*$, $[aabb]*$, and $[aaaabbbb]*$.

The activation matrix baselines are:

- $B_{uniform}$: the song has a single texture;
- $B_{buildup}$: new instruments enter in measures 3, 5 and 7.

In addition to these naive baselines, we tested two simplified versions of our proposed approach. The first skips the source separation step: instead of estimating patterns from 6 separated tracks, we can estimate patterns from the full-audio chroma features, and then duplicate the result 6 times to match the number of estimated sources as the proposed approach ("Chr. w/o SS"). Second, since the activation matrix is evaluated as if it were a timbre label, we also estimate timbre labels by computing full-audio MFCCs, and using NMF to label the measures ("MFCCs"). All section transitions are treated as predictions of entrances and exits.

5. RESULTS AND DISCUSSION

We applied all the approaches described above to the dataset of 200 Beatles songs. The results for the multi-part pattern description task are shown in Table 1 ("Standard approach"). We find that the proposed approach outperforms the naive baselines, but that the simpler approach that skips the source separation step performs even better, even though it has lower recall. The pw_f values are almost all dominated by the lower precision values; like in structure analysis, it seems harder to achieve high precision than high recall. By tweaking the evaluation metric, we can understand why. In the bottom half of the table, we compute pw_f counting the elements on the main diagonal. The B_{null} baseline, which guesses that every measure is different, now becomes very competitive.

The explanation is that unlike in the usual structure analysis task, the ground truth for this task is very sparse. Recall that pairwise f-measure tells us how well the similarity relationships of one description are captured by the similarity relationships in another. In other words, given two binary SSMs that encode similarity descriptions, pw_f assesses how well the positive parts of these SSMs coincide. Since it is trivial to guess that each segment is similar to itself, we should ignore the contributions of the main diagonal. This does not usually affect the outcome of structure evaluation, since the repeating blocks ensure that the

[2] https://en.wikipedia.org/wiki/Hungarian_algorithm

Standard approach				
	Generous			*Strict*
	pw_f	pw_p	pw_r	pw_f
Proposed	.245	.211	.71	.184
Chr. w/o SS	**.297**	**.312**	.529	**.222**
$B_{constant}$	.144	.092	**.95**	.106
B_{null}	0	0	0	0
$B_{periodic}$	.149	.136	.318	.112
B_{block}	.06	.103	.074	.044

Counting self-labeled measures				
	pw_f	pw_p	pw_r	pw_f
Proposed	.365	.309	.819	.274
Chr. w/o SS	.466	.442	.695	.346
$B_{constant}$	.183	.115	**.962**	.135
B_{null}	**.515**	**.749**	.477	**.384**
$B_{periodic}$	.255	.218	.461	.192
B_{block}	.411	.44	.465	.309

Table 1. Above: results for estimated multi-track description quality using the proposed metric. Below: the results if self-labeled measures are counted as correct. The high retrieval for B_{null} illustrates the sparseness of the ground truth.

ground truth SSM has very many off-diagonal pixels to estimate. However, in our application, the ground truth matrices are extremely sparse: in cases where a part never repeats itself exactly, there are no off-diagonal elements.

On the other hand, this task is unlike structure analysis because in our case, elements on the main diagonal *can* equal 0, if the corresponding source is not active. This means that the B_{null} baseline does not actually achieve perfect precision: from the bottom part of Table 1 we can see that on average, sources are active for 75% of the song.

Results for the activation detection task are shown in Table 2. According to the pw_f measure, the best approach to characterize the changing timbre of the piece was our proposed one. However, the uniform baseline performs almost as well according to this metric. Although some songs have over a dozen tracks, with many entrances and exits, it seems that the majority of songs have an instrumentation that changes little. As a result, the uniform and buildup baselines achieve near-perfect recall while precision does not fall below 30%. That said, these naive baselines fail to detect nearly all the entrances and exits of instruments from the mixture, so the proposed approach beats them handily on entrance/exit f-measure.

In contrast, the MFCC approach tends to find a majority of the entrances and exits, and narrowly beats the proposed approach in terms of entrance/exit f-measure. The cost of this apparent over-segmentation is lower pairwise retrieval, and the lowest overall pw_f, for labeling the timbres.

In designing the evaluation, we made an effort to reuse metrics that are used for structure analysis. We did not expect the sparseness of the ground truth to have such an impact on the metrics, but the impact is plain to see in the success of the baselines. Perhaps we should have

	Timbre labeling			*Entrance/exit*		
	pw_f	pw_p	pw_r	f	p	r
Proposed	**.450**	.456	.546	.248	.271	.296
MFCCs	.3	**.549**	.319	**.273**	.195	**.566**
$B_{uniform}$	.433	.306	**.962**	0	0	0
$B_{buildup}$	.446	.328	.909	.071	**.351**	.045

Table 2. Results for estimated activation matrix quality.

anticipated this: data sparseness is often a problem when translating a one-dimensional function (here, the overall structure) into a higher-dimensional space (a per-channel representation). One potential way to resolve this issue is to automatically process both the ground truth and the estimated descriptions using a fixed sequences-to-blocks conversion step, such as that proposed by [9]. This would allow us to compare nearly-equivalent representations that are much less sparse.

Needless to say, the multi-track analysis approach we have proposed could be improved in many ways. We have used two source separation kernels, in a fixed way, but it is possible to apply more kernels, and to do so in an optimization framework to increase the independence of the estimated tracks [13]. Future work should also test a greater variety of source separation methods, especially NMF-based approaches. However, this first effort has helped us to understand the special challenge of this task, which is the sparseness of the ground truth.

6. CONCLUSION AND FUTURE WORK

We have described a new MIR task, multi-part pattern analysis, in which the goal is to describe each independent layer of a piece of music. The task complements recent work on estimating hierarchical structure. We have also proposed a method for estimating multi-part pattern analyses using a combination of existing source-separation tools, SSM-based structure estimation methods, and a novel approach to thresholding SSMs in order to minimize transitivity errors.

To support future work on this problem, we have proposed a method of creating ground truth annotations from MIDI files, and a set of evaluation metrics that can estimate the similarity between two multi-part descriptions or two multi-part activation functions.

In our evaluation, we found the sparseness of the data to be an issue, but it is a direct consequence of how we chose to create the ground truth. As we refine the methodology for this task in future work, we will study the impact of different ways of converting multi-channel files into ground truth recurrence plots.

7. ACKNOWLEDGEMENTS

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

8. REFERENCES

[1] J.-J. Aucouturier and Mark Sandler. Segmentation of musical signals using hidden Markov models. In *Proc. of the Audio Engineering Society Convention*, Amsterdam, The Netherlands, 2001.

[2] Frédéric Bimbot, Gabriel Sargent, Emmanuel Deruty, Corentin Guichaoua, and Emmanuel Vincent. Semiotic description of music structure: An introduction to the Quaero/Metiss structural annotations. In *Proc. of the AES Conference on Semantic Audio*, 2014.

[3] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive mir research. In *Proc. of ISMIR*, pages 155–160, Taipei, Taiwan, 2014.

[4] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. madmom: a new Python Audio and Music Signal Processing Library. In *Proc. of the ACM International Conference on Multimedia*, pages 1174–1178, Amsterdam, The Netherlands, 2016.

[5] Tom Collins, Andreas Arzt, Sebastian Flossman, and Gerhard Widmer. SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations. In *Proc. of ISMIR*, pages 549–554, Curitiba, Brazil, 2013.

[6] Derry FitzGerald. Harmonic/percussive separation using median filtering and amplitude discrimination. In *Proc. of the International Conference on Digital Audio Effects*, Graz, Austria, September 2010.

[7] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Proc. of the IEEE International Conference on Multimedia & Expo*, pages 452–455, 2000.

[8] Jonathan Foote and Matthew Cooper. Media segmentation using self-similarity decomposition. In Minerva Yeung, Rainer Lienhart, and Chung-Sheng Li, editors, *Proc. of the SPIE: Storage and Retrieval for Media Databases*, volume 5021, pages 167–175, Santa Clara, CA, USA, 2003.

[9] Harald Groghanz, Michael Clausen, Nanzhu Jiang, and Meinard Müller. Converting path structures into block structures using eigenvalue decompositions of self-similarity matrices. In *Proc. of ISMIR*, 2013.

[10] Steven Hargreaves, Anssi Klapuri, and Mark Sandler. Structural segmentation of multitrack audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(10):2637–2647, 2012.

[11] Tristan Jehan. Hierarchical multi-class self similarities. In *Proc. of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 311–314, New Paltz, NY, United States, 2005.

[12] Fred Lerdahl and Ray S. Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, 1983.

[13] Antoine Liutkus, Derry Fitzgerald, Zafar Rafii, Bryan Pardo, and Laurent Daudet. Kernel additive models for source separation. *IEEE Transactions on Signal Processing*, 62(16):4298–4310, 2014.

[14] Antoine Liutkus, Zafar Rafii, Roland Badeau, Bryan Pardo, and Gaël Richard. Adaptive filtering for music/voice separation exploiting the repeating musical structure. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 53–56, Kyoto, Japan, 2012. IEEE.

[15] Patricio López-Serrano, Christian Dittmar, Jonathan Driedger, and Meinard Müller. Towards modeling and decomposing loop-based electronic music. In *Proc. of ISMIR*, pages 502–508, New York, NY, USA, 2016.

[16] Hanna Lukashevich. Towards quantitative measures of evaluating song segmentation. In *Proc. of ISMIR*, pages 375–380, Philadelphia, PA, USA, 2008.

[17] Brian McFee and Daniel Ellis. Analyzing song structure with spectral clustering. In *Proc. of ISMIR*, pages 405–410, Taipei, Taiwan, 2014.

[18] Brian McFee, Oriol Nieto, and Juan Pablo Bello. Hierarchical evaluation of segment boundary detection. In *Proc. of ISMIR*, Málaga, Spain, 2015.

[19] Meinard Müller. Music structure analysis. In *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*, pages 167–236. Springer International Publishing, 2015.

[20] Meinard Müller, Nanzhu Jiang, and Peter Grosche. A robust fitness measure for capturing repetitions in music recordings with applications to audio thumbnailing. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(3):531–543, 2013.

[21] Oriol Nieto and Morwaread M. Farbood. Identifying polyphonic patterns from audio recordings using music segmentation techniques. In *Proc. of ISMIR*, pages 411–416, Taipei, Taiwan, 2014.

[22] Geoffroy Peeters and Emmanuel Deruty. Is music structure annotation multi-dimensional? A proposal for robust local music annotation. In *Proc. of the International Workshop on Learning the Semantics of Audio Signals*, pages 75–90, Graz, Austria, 2009.

[23] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016.

[24] Colin Raffel and Daniel P. W. Ellis. Intuitive analysis, creation and manipulation of midi data with pretty_midi. In *ISMIR Late Breaking and Demo Papers*, pages 84–93, 2014.

[25] Zafar Rafii, Antoine Liutkus, and Bryan Pardo. REPET for background/foreground separation in audio. In G. R. Naik and W. Wang, editors, *Blind Source Separation*, Signals and Communication Technology, pages 395–411. Springer-Verlag, 2014.

[26] Chris Sanden, Chad R. Befus, and John Z. Zhang. A perceptual study on music segmentation and genre classification. *Journal of New Music Research*, 41(3):277–293, 2012.

[27] Prem Seetharaman and Bryan Pardo. Simultaneous separation and segmentation in layered music. In *Proc. of ISMIR*, pages 495–501, New York, NY, USA, 2016.

[28] Paris Smaragdis. Non-negative matrix factor deconvolution: Extraction of multiple sound sources from monophonic inputs. In *Independent Component Analysis and Blind Signal Separation*, volume 3195 of *Lecture Notes in Computer Science*, pages 494–499. Springer-Verlag, Berlin, Heidelberg, 2004.

[29] Jordan B. L. Smith, J. Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J. Stephen Downie. Design and creation of a large-scale database of structural annotations. In *Proc. of ISMIR*, pages 555–560, Miami, FL, United States, 2011.

ONE-STEP DETECTION OF BACKGROUND, STAFF LINES, AND SYMBOLS IN MEDIEVAL MUSIC MANUSCRIPTS WITH CONVOLUTIONAL NEURAL NETWORKS

Jorge Calvo-Zaragoza, Gabriel Vigliensoni, and Ichiro Fujinaga
Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
McGill University, Montreal, QC, Canada

ABSTRACT

One of the most complex stages of optical music recognition workflows is the detection and isolation of musical symbols. Traditionally, this goal is achieved by performing preprocesses of binarization and staff-line removal. However, these are commonly performed using heuristics that do not generalize widely when applied to different types of documents such as medieval scores. In this paper we propose an effective and generalizable approach to address this problem in one step. Our proposal classifies each pixel of the image among background, staff lines, and symbols using supervised learning techniques, namely convolutional neural networks. Experiments on a set of medieval music pages proved that the proposed approach is very accurate, achieving a performance upwards of 90% and outperforming common ensembles of binarization and staff-line removal algorithms.

1. INTRODUCTION

Optical music recognition (OMR) is the field of computer science devoted to providing computers with the ability to extract the musical content of a score from the optical scanning of its source image [1]. This problem represents a complex challenge for which there are no completely satisfactory solutions yet [20]. The task can be further divided into two different stages [6]: document image processing, in which the objective is to detect and recognize each meaningful symbol appearing in the image; and reconstruction of musical notation, in which musical meaning is assigned to each of these symbols in order to encode the content in a structured symbolic music format such as MEI (Music Encoding Initiative) or MusicXML.

Due to the arrangement of the elements on the staff, the image-processing stage is usually approached following a strategy of segmentation and classification. That is, elements within the score are first detected independently and then a classification algorithm assigns a category to each of them. Our approach focuses on the segmentation stage.

The final objective of segmentation is to detect the regions of the image that correspond to music symbols. To achieve this, traditional segmentation workflows incorporate the steps of binarization of the document and detection and removal of staff lines. Staff-line detection and removal algorithms usually use a binarized image as input, which facilitates certain procedures such as morphological operations or histogram analysis—core processes of many of these algorithms. In addition, the segmentation workflow also allows for the detection of staff line positions. If symbol isolation were done from a color image, it would not know which parts belong to the background of the document and which to staff lines. Note that the position of staff lines is crucial for determining the pitch of the symbols.

The traditional segmentation workflow, however, has a number of drawbacks. First, the staff-line detection and removal becomes heavily dependent on the accuracy of binarization, as errors are propagated between the two stages. In addition, the traditional methods follow heuristic techniques that assume specific conditions in the images to be treated. While this may be useful if the context of their use is limited to a particular style of documents, it is difficult to generalize these methods so that they can be used in various cases. This is especially true when dealing with medieval manuscripts, which present a greater heterogeneity in this regard.

For all of the above cases, we propose a framework with the goals of isolating the symbols depicted in the image of a music score and keeping the staff-line information. In our approach we perform a document analysis procedure that allows for categorical discrimination of each pixel, according to the class it belongs to (e.g., *background*, *staff lines*, or *symbols*) in a single step. In order to make this approach generalizable we address the task using the supervised learning paradigm. That is, we assume that a reference set is available that can be used to train a model to perform such task. In particular, we make use of convolutional networks for this purpose. These networks are powerful models that are capable of learning a suitable representation for a given task, thus avoiding the necessity of developing a feature extraction strategy specifically designed for each type of document to be processed. Our experiments on two sets of medieval documents report excellent results, outperforming different combinations of binarization algorithms and staff-line removal algorithms.

The rest of the paper is structured as follows: related work and the context of our proposal is presented in Section 2; the proposed method to solve the problem is detailed in Section 3; the experimental setup to validate our approach is described in Section 4; comparative and qualitative results are reported in Section 5; and conclusions and promising avenues for future work are summarized in Section 6.

2. RELATED WORK

OMR has to deal with many aspects of musical notation, one of which is the presence of the staff. Since most symbols in the score are connected through these lines, it has been traditionally necessary to remove them in order to detect musical symbols.

The staff-line removal stage is usually performed after the binarization of the document in the OMR workflow [20] because this step helps to reduce the complexity of the problem and is required to apply certain techniques such as morphological operators, histogram analysis, or connected components. In addition, starting from the color image, the processes of binarization and staff-line removal, one after the other, allow the separation of background, staff lines, and musical symbols regions.

A comprehensive review and comparison of the early attempts for the staff-line removal can be found in the work of Dalitz et al. [7]. Given the interest in this challenging task, many other methods have been proposed more recently. Cardoso et al. [9] proposed a method that considers the staff lines as connecting paths between the two margins of the score. The score was modeled as a graph so that staff detection was solved as a maximization problem. Dutta et al. [10] developed a method for printed scores that considered the staff-line segment as a horizontal connection of vertical black runs with uniform height. Piatkowska et al. [18] designed a method that used a Swarm Intelligence algorithm. Their approach can apparently deal with any type of image, but only results on binary images were reported. Su et al. [23] fitted an approximate staff considering properties such as height and space. Geraud [11] developed a method that entails a series of morphological operators: first, a permissive hit-or-miss with a horizontal line pattern, followed by a horizontal median filter and a dilation operation. A binary mask is then obtained with a morphological closing. Finally, a vertical median filter is applied to the largest components of the mask. The procedure is directly applied to the image, which eventually removes staff lines. Montagner et al. [15] proposed to learn image operators, whose combination remove staff lines.

The problem with these methods is that they focus on particular aspects of the style of the specific scores toward which they are oriented and it is, therefore, very difficult to adapt them to other types of documents (for example, from different eras or with different notations or styles). In addition, most of these methods assume already binarized images as input. The binary nature of modern musical scores (black ink on white paper) has, to some extent, justified this assumption. Of course, document binarization is not a trivial problem—especially when dealing with ancient documents [16]. Furthermore, it turns out that traditional document binarization methods, which were designed mainly for text documents, are often not suitable for musical scores [4].

Here we introduce a more generalized framework to solve the whole segmentation problem directly. The framework is based on machine learning so that it can be applied to a wide variety of musical notation styles and musical documents, as long as training data is available. Our strategy is inspired by the work of Calvo-Zaragoza et al. [5], in which a Support Vector Machine classifier was trained to discriminate if a *foreground* pixel of a binary image belongs to a *symbol* or to a *staff line*. Our approach is similar in formulation, but we do not assume that the documents are binarized or that they contain only symbols or staff lines. Furthermore, we also extend the procedure by using a more advanced classification scheme based on Convolutional Neural Networks (CNN).

3. METHOD

Although rarely formulated in this way, the problems related to image processing for musical documents are concerned with pixel-level classification processes. That is, for each pixel of the image we want to know whether it belongs to a musical symbol or not. In the latter case, we want to know whether the pixel belongs to a staff line or not, as this information is valuable for determining the vertical position of the notes (pitches), among others.

Therefore, the process can be formulated as a classification problem in which a model is trained to distinguish the category a given pixel belongs to. Formally, our approach considers a model that categorizes a given pixel into three possible classes: *background*, *staff*, and *symbol*. The requirement to carry out this idea consists of a reference set that allows providing examples of each category to the supervised learning algorithm.

In our framework, this classification process is carried out by means of Deep Learning. Recently, Deep Neural Networks have shown a remarkable leap of performance in pattern recognition. Specifically, CNN have been applied with great success for the detection, segmentation, and recognition of objects and regions in images, approaching human performance on some of these tasks [13].

CNN are composed of a series of filters (i.e., convolutions) that obtain several representations of the input image. These filters are applied in a hierarchy of layers, each of which represent different levels of abstraction; while filters of the first layers may enhance details of the image, filters of the last layers may detect high-level entities [12]. The key to this approach is that, instead of being fixed, these filters are modified through a gradient descent optimization algorithm called back-propagation [14].

One of the main advantages of CNN is their ability to learn a suitable representation of the training data without any human intervention, affording greater generalization to documents of different style. In other words, these networks learn a suitable representation of the data from raw data, without the need of feature extraction. Since collections of music documents are a rich source of highly complex information—often more heterogeneous than other types of documents—a framework based on CNN is promising.

3.1 Input Feature Set

As mentioned above, our intention is to train a CNN to differentiate pixels belonging to the different categories. Analyzing the organization of musical documents, we hypothesize that a pixel can be correctly categorized by using its local, neighboring information. In other words, we assume that the surrounding region of a pixel contains enough discriminative information to classify it into its correct category. As a result, the input set to the classifier in our framework is a portion of the input image, centered at the pixel of interest. Figure 1 illustrates some examples of input feature set for each of the considered categories, in which the pixel to be classified is located in the center of the patch.

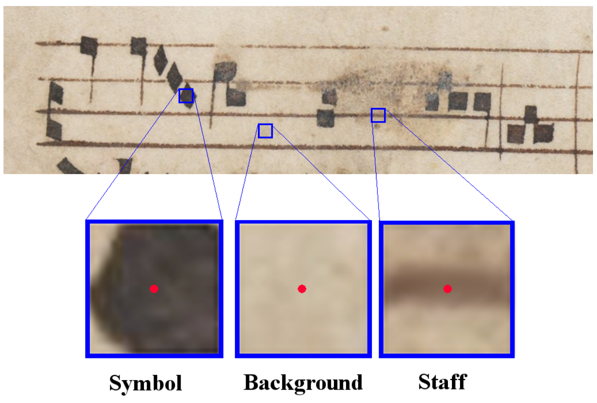


Figure 1. Example of input feature sets for pixels of interest of the three classes: *symbol*, *background*, and *staff*. Note that the pixel to be classified is located at the center of each window (highlighted in red for a better illustration).

Note that the method can work directly with color images and that the size of the neighborhood (i.e., the size of the window) is a parameter to be tuned according to the scale of the images to be processed.

3.2 Convolutional Neural Networks

Since there are no previously proposed CNN models to solve a task of this kind, we designed a new network configuration. Note, however, that the ultimate goal of this paper is not to find the best network topology—which would involve a comprehensive set of experiments to find the best set of parameters—but to demonstrate that the proposed categorization of music documents based on pixel-wise classification with CNN is feasible.

Our design is inspired by the VGG network [22], a topology widely used in the computer vision community for object recognition. This network contains several layers of convolution plus 2×2 max-pooling (16 or 19, depending on its version). By means of informal testing we simplified this network to up to 3 layers, adjusting the number of convolutional filters per layer to 64, and the size of the convolution kernels to 7.

Learning of the network weights is performed by means of stochastic gradient descent [2] with a batch size of 32, considering the adaptive learning rate proposed by Zeiler [26] (default parameterization) and a *cross-entropy* loss function. Once the CNN has learned how to distinguish among the considered categories it can be used to perform the layout analysis of a document. To do so, each pixel of the image is queried, and its feature set is forwarded and processed by the network in order to obtain its most likely category.

4. EXPERIMENTAL SETUP

4.1 Corpora

We trained and tested our approach on a set of high-resolution image scans of two different old music documents. The first corpus is a subset of 10 pages of the Einsiedeln, Stiftsbibliothek, Codex 611(89), from 1314. [1] The second corpus consists of 10 pages of the Salzinnes Antiphonal manuscript (CDM-Hsmu M2149.14), music score dated 1554–5. [2] Pages from the two manuscripts are shown in Figure 2. As a reference measure for scale, both pages depict a separation between staff lines of approximately 50 pixels.

Note that the image scans of these two manuscripts have zones with different lighting conditions that may affect the performance of the proposal we evaluate. The Einsiedeln manuscript images, in particular, present areas with severe bleed-through that may mislead the automatic recognition.

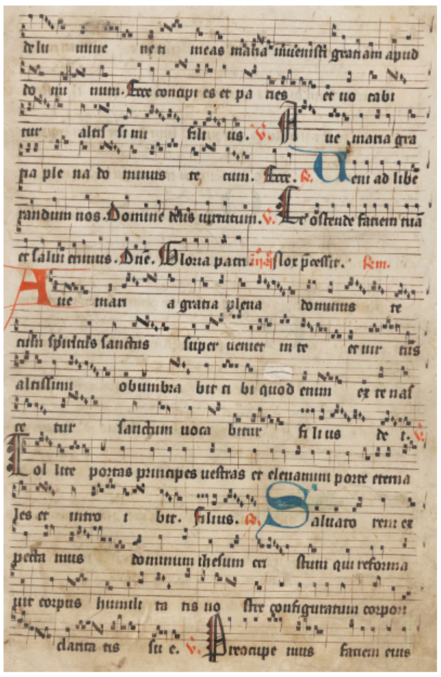
The ground-truth data from the corpora was created by manually labeling pixels into the three categories considered, as illustrated in Figure 3. Note that the class *symbol* includes both musical symbols and other types of symbols (such as lyrics). This should not be an issue as there exist successful algorithms to separate music and lyrics [3].

Taking into account the scale of the images of our corpora, an input window size of 41×41 pixels was empirically chosen, which corresponds to more than half of the space between the staff lines.

4.2 Comparative Assessment

To the best of our knowledge, there are no other algorithms that perform a direct detection of staff lines and symbols from music document images, and so we decided to compare our approach with combinations of standard binarization and staff-line removal algorithms. In order to select these algorithms, we took into account the results of the *ICDAR / GREC 2013 Competition on Music Scores: Staff Removal* [24]. In this contest, the two strategies that obtained the best performance were LRDE [11] and INESC [9].

[1] `http://www.e-codices.unifr.ch/en/sbe/0611/`
[2] `https://cantus.simssa.ca/manuscript/133/`


(a) Einsiedeln

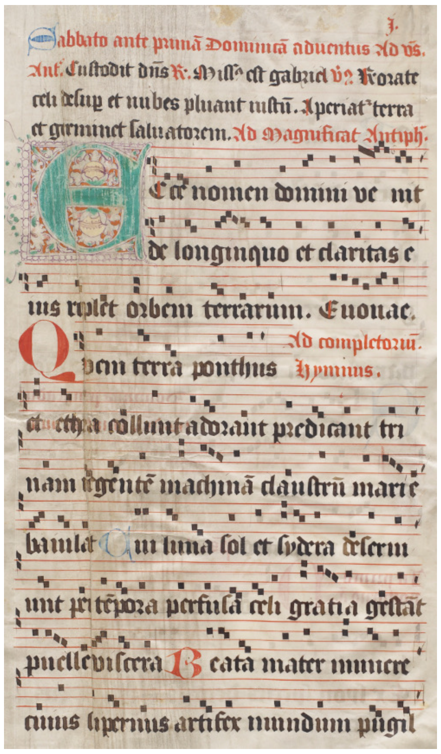
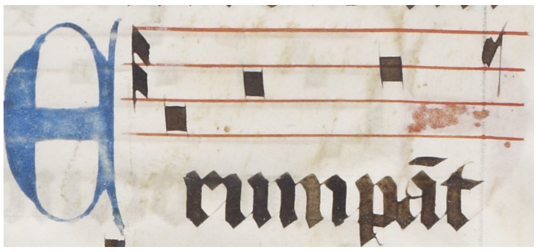

(b) Salzinnes

Figure 2. Pages from the corpora used in this work.


(a) Source image

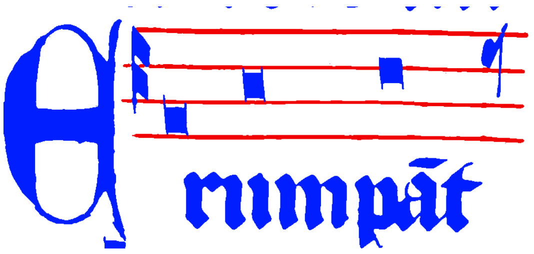

(b) Ground-truth

Figure 3. Example of ground-truth created. Background pixels are labeled in white, staff-line pixels are labeled in red, and symbol pixels are labeled in blue.

These methods were based on published approaches (described in Section 2).

As mentioned before, these methods require that the input image only contains binary values. Therefore, the following binarization strategies are considered:

Sauvola method [21] is perhaps the most widely considered binarization algorithm for document images. It is based on the assumption that foreground pixels are closer to black than background pixels. It computes a threshold at each pixel considering the mean and standard deviation of a square window centered at the pixel under consideration.

Wolf & Jolion method [25] is an extension of Sauvola's, with a change in threshold formula to normalize contrast and the mean gray-level of the considered square window.

BLIST method [19] (Binarization based in LIne Spacing and Thickness) is specially designed for binarizing music scores. It consists of an adaptive local thresholding algorithm based on the estimation of the features of the staff lines depicted in the score.

To obtain the three categories mentioned above, we assume that *background* are those pixels removed by the binarization algorithm, while *staff* are those removed by the staff-line removal algorithm from the binarized image. The remaining pixels are thus classified as belonging to the *symbol* category.

Each combination of staff-line removal and binarization methods was evaluated experimentally. To assure a fair comparison, the parameters for each method (if any) were tuned to obtain the best results in the training set.

4.3 Evaluation

To evaluate our proposal, we used a scheme of 10-fold cross-validation on each corpus. That is, at each iteration, one of the pages was used as a test set, and the other nine were used to train the network and optimize its configuration. Specifically, 30 000 samples of each of the three classes were randomly selected for training (total: 90 000), while 600 000 of each class (total: 1 800 000) were used as a validation set. Note that these partitions represent a tiny portion of the available data, as each page contains about $2 \cdot 10^7$ pixels. However, these values were considered adequate to successfully train the networks (both in accuracy and computational load) on the machines that were used for that purpose. A more clever use of all available data will be discussed when addressing future work. As a result, the training set was used to optimize the CNN through gradient descent, whereas the validation set was used to select the most appropriate epoch to stop the learning process and prevent over-fitting. The complete testing pages were finally used to measure the performance of the model created by the network during training.

Given that the number of pixels of each class is not evenly balanced in the documents, we consider the F-measure (F_1) class-wise figure of merit for quantitatively assessing the classification accuracy of the system. Taking one class at a time as reference, this metric summarizes the correctly classified elements (*True Positive*, TP), elements falsely classified as belonging to the reference set (*False Positive*, FP), and elements of the reference set misclassified as belonging to another category (*False Negative*, FP) in a single value. Then, the F_1 is formalized as:

$$\text{F}_1 = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \; .$$

Finally, in order to minimize the possibility that the differences in model performance were due to chance variation, we will perform a pairwise, non-parametric test (Wilcoxon signed rank [8]).

5. RESULTS

We show in Table 1 the average F_1 results obtained in each corpus, as well as the overall performance when the whole set of documents is taken into consideration.

As can be seen in the table, the staff-line removal algorithm is the most relevant element in the considered configurations, because the differences are smaller when varying the binarization algorithm. In particular, the LRDE approach reports poor results in both sets of documents, despite having obtained the best results in the aforementioned competition. This directly demonstrates the lack of generalization of this approach. The INESC algorithm exhibits a fair performance, especially in the Salzinnes corpus. In regard to binarization algorithms, no conclusion can be drawn since the results seem too similar and depend on the corpus.

Strategy		Dataset		
		Einsiedeln	Salzinnes	Whole
LRDE	Sauvola	58.5	78.6	68.6
	Wolf	58.7	70.6	64.6
	BLIST	59.2	74.0	66.6
INESC	Sauvola	80.3	91.6	86.0
	Wolf	83.0	90.7	86.9
	BLIST	83.8	88.0	85.9
CNN		88.0	92.6	90.3

Table 1. Average F_1 obtained in the 10-fold cross-validation scheme for each corpus and the whole set.

The approach based on CNN, which performs the process in a single step, yields the best results in all cases considered. Since these results only reflect the average performance, we used the 20 independent results (10 for each corpus) to perform statistical tests. It resulted in p-values below 0.01 in all pairwise comparisons, and so our approach is significantly better than the rest of the configurations with an alpha significance level of 99%.

In order to have a qualitative reference, Table 2 shows an example of the categorization obtained by LRDE and INESC methods on a piece of Einsiedeln documents, considering BLIST binarization (best case), as well as the categorization of the approach based on CNN. It is observed that LRDE is only able to partially detect one of the lines of staff. INESC achieves an acceptable performance but it mislabels some sections of the staff. CNN shows a prediction that is very similar to the reference one. In addition, it completes one of the staff lines that is not perfectly seen in the original document (which, in turn, may be detrimental when computing its accuracy). Also, the CNN tends to mislabel pixel close to boundaries of elements, in which is not clear the actual category of the pixel. It is expected, however, that these errors will not cause inconveniences in subsequent procedures of the recognition workflow.

All in all, we can state that a trained CNN can successfully detect the selected categories at the pixel level in images of music scores. Our approach reports the best performance among the evaluated methods although it is fair to say that it is not by a wide margin. Nevertheless, its strength can be observed in the improvements achieved in each corpus. On the Salzinnes corpus, which seems to be less degraded and simpler, the margin was narrower. However, in the Einsiedeln manuscript the improvement over the compared methods was higher. This could mean that, as the difficulty increases, our approach could be more generalizable and adaptable.

It should be emphasized that the intention of this work was not to find the most suitable combination of input feature size and network topology, but to show that this approach allows dealing with the analysis of music doc-

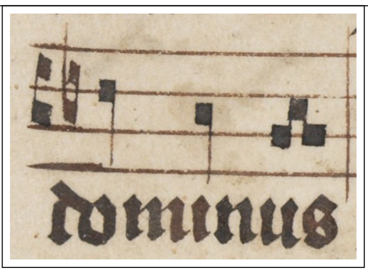
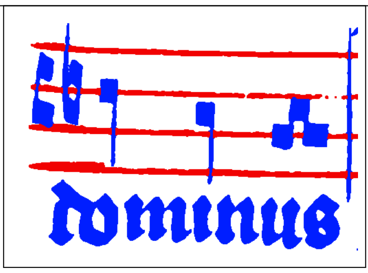
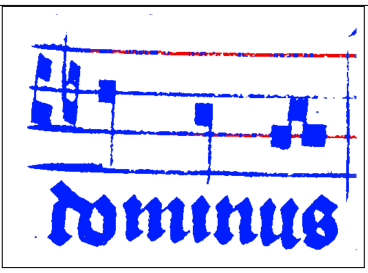
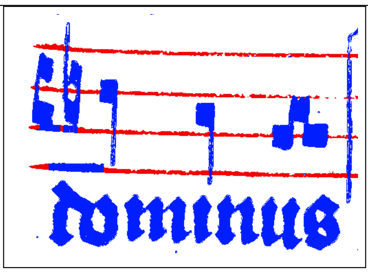
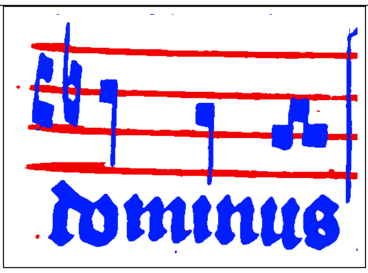
Source	Ground-truth	
		
BLIST+LRDE	**BLIST+INESC**	**CNN**
		

Table 2. Qualitative examples of categorization from Einsiedeln document, depicting the original piece along with the manually created ground-truth, and the labeling predicted by BLIST+LRDE, BLIST+INESC, and CNN. Coloring: background in white, staff lines in red, and symbols in blue.

uments successfully. Therefore, a more comprehensive search of the optimal parameters could be carried out to obtain an even better performance.

6. CONCLUSIONS

In this paper we presented a framework for detecting background, staff lines, and symbols in medieval manuscripts. Our approach was based on the classification of the different elements of the image at pixel level using machine learning. We use a CNN along with a training dataset of reasonable size that contained examples for each category.

Our results showed that the accuracy obtained is high, achieving around 90% of F_1 in the evaluated corpus. It has also been shown that our proposal is able to outperform state-of-the-art strategies based on heuristic image processing, demonstrating that CNN is a robust and generalizable alternative to those traditional approaches.

In future work, efforts should be devoted to overcoming the problem of getting enough data to train the CNN. It could be interesting to consider an incremental interactive framework in which the user does not have to label every single pixel of the image but only those erroneously labeled by a base classifier. The use of transfer learning [17] is another way to reduce the initial effort when dealing with a new type of document.

Moreover, there are several ways to improve the accuracy of the model in the future. Of course, finding a more suitable network configuration for this problem is a way of improving the results presented here. Also, since the available data is very large (i.e., a single page of ground-truth provides millions of examples of pixels labeled by humans), it would be more beneficial to train the network following a smarter strategy than choosing a random subset of the available data. For example, a random training set can be initially chosen to perform a first training iteration (as in the case of this work). After that, training documents can be evaluated so that the network is re-trained only with those pixels that would be misclassified by the current model. In this way, the network would pay special attention to the most difficult cases.

7. ACKNOWLEDGEMENT

This work was partially supported by the Social Sciences and Humanities Research Council of Canada.

Special thanks to Vi-An Tran for manually labeling the different layers in all the manuscripts used for this research.

8. REFERENCES

[1] D. Bainbridge and T. Bell. The challenge of optical music recognition. *Computers and the Humanities*, 35(2):95–121, 2001.

[2] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] J. A. Burgoyne, Y. Ouyang, T. Himmelman, J. Devaney, L. Pugin, 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*, pages 723–727, 2009.

[4] J. A. Burgoyne, L. Pugin, G. Eustace, and I. Fujinaga. A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources. In *Proceedings of the 8th International Conference on Music Information Retrieval*, pages 509–512, 2007.

[5] J. Calvo-Zaragoza, L. Micó, and J. Oncina. Music staff removal with supervised pixel classification. *International Journal on Document Analysis and Recognition*, 19(3):211–219, 2016.

[6] J. Calvo-Zaragoza, G. Vigliensoni, and I. Fujinaga. Document analysis for music scores via machine learning. In *Proceedings of the 3rd International Workshop on Digital Libraries for Musicology*, pages 37–40. ACM, 2016.

[7] C. Dalitz, M. Droettboom, B. Pranzas, and I. Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–766, 2008.

[8] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[9] 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*, 31(6):1134–1139, June 2009.

[10] A. Dutta, U. Pal, A. Fornés, and J. Lladós. An efficient staff removal approach from printed musical documents. In *20th International Conference on Pattern Recognition, ICPR 2010, Istanbul, Turkey, 23-26 August 2010*, pages 1965–1968, 2010.

[11] T. Géraud. A morphological method for music score staff removal. In *Proceedings of the 21st International Conference on Image Processing (ICIP)*, pages 2599–2603, Paris, France, 2014.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *26th Annual Conference on Neural Information Processing Systems*, pages 1106–1114, 2012.

[13] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[15] I. d. S. Montagner, R. Hirata, and N. S. T. Hirata. A machine learning based method for staff removal. In *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR)*, pages 3162–3127, 2014.

[16] K. Ntirogiannis, B. Gatos, and I. Pratikakis. ICFHR2014 competition on handwritten document image binarization (H-DIBCO 2014). In *14th International Conference on Frontiers in Handwriting Recognition*, pages 809–813, 2014.

[17] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[18] W. Piatkowska, L. Nowak, M. Pawlowski, and M. Ogorzalek. Stafflines pattern detection using the swarm intelligence algorithm. In L. Bolc, R. Tadeusiewicz, L. J. Chmielewski, and K. Wojciechowski, editors, *Computer Vision and Graphics*, volume 7594 of *Lecture Notes in Computer Science*, pages 557–564. Springer Berlin Heidelberg, 2012.

[19] T. Pinto, A. Rebelo, G. A. Giraldi, and J. S. Cardoso. Music score binarization based on domain knowledge. In *Proceedings of the 5th Iberian Conference on Pattern Recognition and Image Analysis*, pages 700–708, Las Palmas de Gran Canaria, Spain, 2011.

[20] 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*, 1(3):173–190, 2012.

[21] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *Computing Research Repository*, abs/1409.1556, 2014.

[23] B. Su, S. Lu, U. Pal, and C. L. Tan. An effective staff detection and removal technique for musical documents. In *Proceedings of the 10th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 160–164, 2012.

[24] M. Visaniy, V. Kieu, A. Fornés, and N. Journet. ICDAR/GREC 2013 music scores competition: Staff removal. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1407–1411, 2013.

[25] C. Wolf, J.-M. Jolion, and F. Chassaing. Text localization, enhancement and binarization in multimedia documents. In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 1037–1040, 2002.

[26] M. D. Zeiler. ADADELTA: An adaptive learning rate method. *Computing Research Repository*, abs/1212.5701, 2012.

OPTICAL MUSIC RECOGNITION WITH CONVOLUTIONAL SEQUENCE-TO-SEQUENCE MODELS

Eelco van der Wel
University of Amsterdam
eelcovdw@gmail.com

Karen Ullrich
University of Amsterdam
karen.ullrich@uva.nl

ABSTRACT

Optical Music Recognition (OMR) is an important technology within Music Information Retrieval. Deep learning models show promising results on OMR tasks, but symbol-level annotated data sets of sufficient size to train such models are not available and difficult to develop. We present a deep learning architecture called a Convolutional Sequence-to-Sequence model to both move towards an end-to-end trainable OMR pipeline, and apply a learning process that trains on full sentences of sheet music instead of individually labeled symbols. The model is trained and evaluated on a human generated data set, with various image augmentations based on real-world scenarios. This data set is the first publicly available set in OMR research with sufficient size to train and evaluate deep learning models. With the introduced augmentations a pitch recognition accuracy of 81% and a duration accuracy of 94% is achieved, resulting in a note level accuracy of 80%. Finally, the model is compared to commercially available methods, showing a large improvements over these applications.

1. INTRODUCTION

Optical Music Recognition (OMR) is an application of recognition algorithms to musical scores, to encode the musical content to some kind of digital format. In modern Music Information Retrieval (MIR), these applications are of great importance. The digitization of sheet music libraries is necessary first step in various data-driven methods of musical analysis, search engines, or other applications where digital formats are required.

OMR is an active area of research in MIR, and a classically hard problem. OMR systems need to deal with a large range of challenges such as low quality scans, ambiguous notation, long range dependencies, large variations in musical font, and handwritten notation. Multiple commercial applications are available, each with their own strengths and weaknesses [3], but the accuracy of these products is often too low to use without human supervision. An implementation that deals with these challenges in a satisfactory way has yet to be developed.

A traditional OMR system typically consists of multiple parts: score pre-processing, staff line identification and removal, musical object location, object classification and score reconstruction [15]. Each of these individual parts has its own difficulties, resulting in OMR systems with low confidence. More recently, there has been a trend towards less segmented systems involving machine learning methods, such as OMR without staffline removal [13] or without symbol segmentation [16]. However, a major difficulty of these algorithms is the need for large amounts of training data. Typically, scores need to be annotated on musical symbol level to train such machine learning pipelines, but large corpora of sufficiently diverse symbol-annotated scores are difficult and expensive to produce [15].

In this study, we propose a novel deep learning architecture to both move towards an end-to-end trainable OMR pipeline, and greatly reduce the data requirements for training. This is achieved by using two common deep learning architectures: *Convolutional Neural Networks* (CNN) and *Recurrent Neural Networks* (RNN). Convolutional architectures have been a popular choice of algorithm in various MIR related tasks, due to the ability to learn local structures in images, and combining them to useful features. In our method, we use a CNN to learn a feature representation of the input scores. Continuing, a *Sequence-to-Sequence* model [5, 18] is used, which is a stack of two RNN's used commonly in machine translation tasks. This model directly produces a digital representation of the score from the learned representation by the CNN. The combination of these two architectures is called a *Convolutional Sequence-to-Sequence* model. By using a Sequence-to-Sequence architecture, we cast the problem of OMR as a translation problem. Instead of training on individual segmented symbols without context, full lines of sheet music are translated simultaneously. This approach has two major advantages; Firstly, by training the algorithm on full lines of sheet music, there is no need for symbol level annotated training data. This means that in principle any corpus of sheet music with corresponding digital notation could be used for training, opening up many new possibilities for data-driven OMR systems. Secondly, because in the proposed model each of the classically segmented OMR steps is done by a single algorithm, the model can use a large amount of contextual information to solve ambiguity and long-range dependency problems.

To train the proposed model, a large corpus of monophonic sheet music is generated from a MusicXML dataset as described in Section 3. Additionally, in Section 3.2 various types of image augmentations based on real-world scenarios are proposed to enhance the models flexibility to different kinds of fonts and varying score quality. Finally in Section 5, the results of the method are discussed on both clean and augmented data, and the weaknesses of the model are examined.

2. RELATED WORK

A starting point for any OMR research is the overview paper by Rebelo et al. [15], which contains a complete introduction to OMR systems and a description of the current state of the field. The paper describes four main stages that are necessary for any OMR pipeline: Image pre-processing, musical symbol recognition, musical information reconstruction and construction of musical notation. The second component, as the name suggests, is where the main recognition work is done. Detecting and removing staff lines, segmenting individual symbols, and classifying symbols. Systems where steps are conducted by different methods we call segmented systems.

Not all methods follow this model, recent data-driven approaches suggest merging or omitting some of these segmented steps. An example of this is an approach suggested by Pugin et al. [12, 13], which applies *Hidden Markov Models* (HMM) to the recognition stage, without performing staff line removal. Shi et al. [16] incorporate a deep learning approach with *Connectionist Temporal Classification* function [6] as decoding mechanism. They pose a similar idea to the method proposed in this research, with a difference in the encoder mechanism. Instead of using both a CNN and RNN as encoder, only a CNN is used. This is less computationally expensive, but the additional RNN in the Sequence-to-sequence model can make the method proposed in this research more context aware.
Symbol classification involving neural networks has been researched by several authors [14, 19]. Convolutional architectures have been used for different OMR sub-tasks, such as staff-line detection [4] or symbol recognition [11].

In a different paper, Rebelo et al. [14] research the use of Deformable Templates [8] with various classifiers to make symbol recognition invariant to changes in musical font. This method is very similar to the Elastic Transformations [17] used in this research. However, we decide to use Elastic Transformations for ease of use and application speed.

3. DATASET

The dataset used in this research is compiled from monophonic MusicXML scores from the MuseScore sheet music archive [1]. The archive is made up of user-generated scores, and is very diverse in both content and purpose. As a result, the dataset contains a large variation in type of music, key signature, time signature, clef, and notation style.
To generate the dataset, each score is checked for monophonicity, and dynamics, expressions, chord symbols, and textual elements are removed. This process produces a dataset of about 17 thousand MusicXML scores. For training and evaluation, these scores are split into three different subsets. 60% is used for training, 15% for validation and 25% for the evaluation of the models. A specification to reproduce the data set is publicly available online. [1]

3.1 Preprocessing

From the corpus of monophonic MusicXML files, a dataset of images of score fragments and corresponding note annotations is created. Each MusicXML score is split into fragments of up to four bars, with a two bar overlap between adjacent fragments. The fragments are converted to sheet music using MuseScore [1], each image containing a single staff line. The corresponding labels are represented with a pitch and duration vector, containing all information about the notes and rests within the same four bars. Each musical symbol is represented with two values: a pitch, and a duration. Pitch values are specified by a MIDI pitch, and durations by quarterlength. In case of a rest, the pitch is a special rest indicator, which we indicate with r. The possible duration classes contain only the durations that can be specified by a single notehead. Notes with durations that require multiple noteheads are split into multiple notes. The first note will contain the pitch, and pitches of subsequent tied notes are replaced with a tie indicator, which we indicate with t

As an example, a quarter rest followed by a note with MIDI pitch 60 and a complex duration of a tied quarter note and a sixteenth note is notated as $((r, 1), (60, 1), (t, 0.25))$. Applying this method to the full score fragments produces the pitch and duration vector, and is a suitable representation for the model. A maximum of 48 events per fragment is used to put a limit on the sequence length the model has to decode. Finally, at the end of each pitch and duration vector an extra event is added to indicate the sequence has ended. This indicator is implemented as a rest with duration of zero quarter notes.

Each generated image is padded to the same width and height, and images containing notes with more than five ledger lines are discarded. These notes are extreme outliers and do not occur in normal notation. The resulting fragments have a dimension of 2261×400 pixels.

3.2 Image Augmentation

The computer generated score fragments contain no noise or variation in musical symbols. To make the proposed model robust to lower quality inputs and different kinds of musical fonts we propose four different augmentations, each simulating a real world source of input noise. Additionally, for each augmentation, we choose two separate

[1] https://github.com/eelcovdw/mono-musicxml-dataset

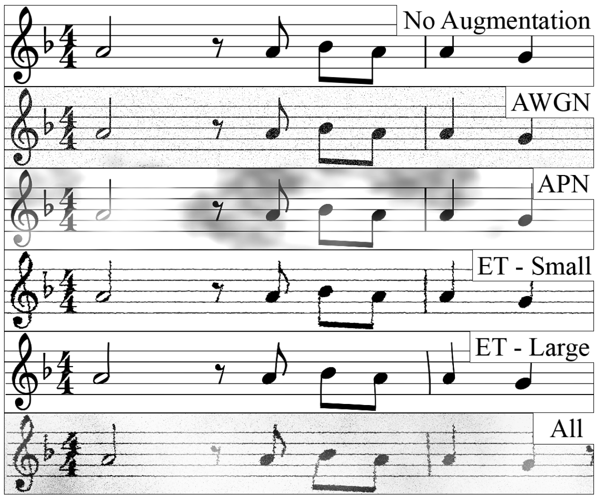


Figure 1. An example of each of the used image augmentations from the evaluation dataset. From top to bottom: No Augmentations, Additive White Gaussian Noise (AWGN), Additive Perlin Noise (APN), Small scale Elastic Transformations (ET small), Large scale Elastic Transformations (ET large), and all combined augmentations.

settings. For the augmented training data, the parameters are chosen such that the input sheet music is greatly deformed but still readable. For the augmented evaluation set, parameters are chosen such that they resemble real-world sheet music, with less deformation than the training data. The larger amount of deformation in training will force our model to learn to recognize musical symbol in any situation, and should improve the accuracy of our model on both non-augmented and augmented evaluation data.

A popular choice of augmentation is *Additive White Gaussian Noise* (AWGN). This augmentation introduces a normally distributed random deviation in pixel intensities, to mimic noise introduced by low quality scans or photos. This noise has a mean μ, which is chosen to be the same as the mean pixel intensity of the full dataset. The standard deviation σ is different between our training and evaluation set. In the training set, the σ of pixel intensities in our non-augmented data set is used. The evaluation set has a σ of half that value.

The second type of noise augmentation used is *Additive Perlin Noise* [10]. Perlin noise is a procedurally generated gradient noise, that generates lighter and darker areas in the image at larger scales than AWGN. This effect mimics quality differences in parts of the score. Some symbols might be faded and parts of staff lines less visible, and dark areas in the image are created. The mean size of generated clouds is controlled by a frequency parameter. For each augmented score, this frequency is chosen to be a random value between the size of one note head and the mean width of a full bar, to generate noise structures at different scales. The maximum intensity of the noise in our training set is chosen to be a strength of 0.8. The evaluation set uses a maximum intensity of half this value.

The final two augmentations are achieved with *Elastic Transformations* (ET) [17], which apply a smoothed field of local random affine transformations, resulting in wave-like displacements in the augmented image. An advantage of using this augmentation is that it applies a large range of possible affine and geometric transformations to each image, such as rotation, skewing, squeezing and stretching. This both enhances the diversity of the augmented data and alleviates the need to use manually defined geometric transformations.

Two parameters are used to control an elastic transformation: a strength factor σ, which reduces the strength of the distortion if a larger value is used, and a smoothing factor α, which controls the scale of deformations. A very large α will apply a nearly linear translation to the image, while an α of zero applies fully random displacements on individual pixels.

The first type of Elastic Transformation is applied on very small scales, to change the characteristics of lines and smaller symbols. Lines might appear to be drawn by pencil or pen, and the edges of symbols become less defined. α is chosen to be a random value between 2 and 8, with a σ of 0.5 for the training data, and a σ of 2 for the evaluation data.

The second type of Elastic Transformation is applied on a large scale to change the shape and orientation of musical symbols. Barlines and note stems get skewed or bent, note heads can be compressed or elongated, and many new shapes are introduced in the score. This transformation mimics the use of different musical fonts, or even handwritten notation. An α between 2000 and 3000 is used, with a σ of 40 for the training data, and 80 for the evaluation data. To maintain straight and continuous stafflines, the original algorithm is slightly adapted to reduce vertical translations of pixels by reducing the vertical component of transformations by 95%.

In Figure 1, an example of each of these four augmentation is shown, with the setting used for generating the evaluation data. The last example shows a combination of all four augmentations.

4. METHOD

We introduce the Convolutional Sequence-to-Sequence model as applied to OMR tasks, translating lines of sheet-music to a sequence of (pitch, duration) pairs. Continuing, the training and evaluation methods are defined.

4.1 Model

We define a Convolutional Sequence-to-Sequence network as a stack of three components. First, a CNN encodes the input image windows to a sequence of vector representations. Then, an encoder RNN encodes the vector sequence to a fixed size representation, containing all information from the input score. Finally, a decoder RNN decodes the fixed size representation to a sequence of output labels. The following section describes each component in detail.

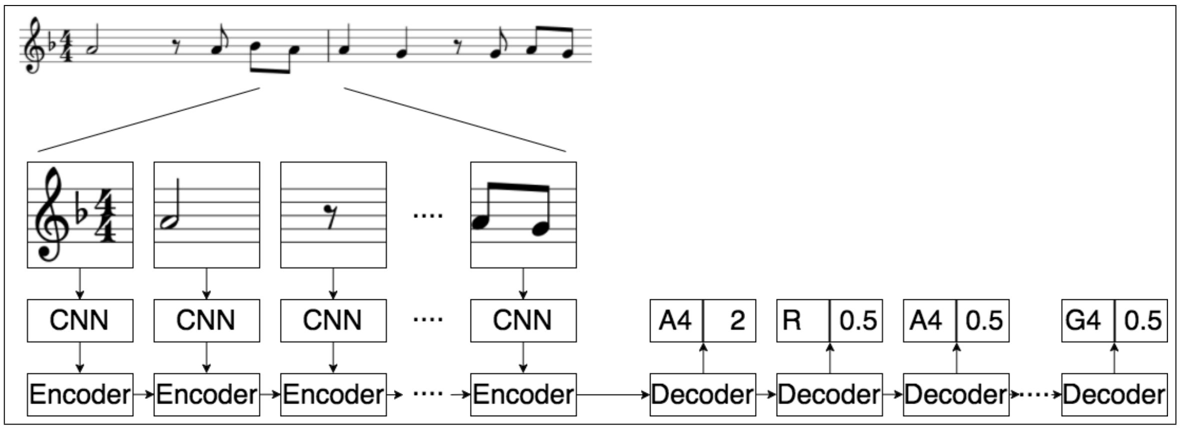


Figure 2. A diagram of the proposed Convolutional Sequence-to-Sequence model. On the left, a score fragment is processed by a CNN and Encoder RNN to a fixed size representation. This representation is used by the decoder RNN to create a sequence of (pitch, duration) pairs.

Note that, while each component is described separately, the model will be trained as a single algorithm.

Sliding window input. The image input of the algorithm is defined as a sequence of image patches, generated by applying a *sliding window* over the original input score. The implementation has two separate parameters: the window width w and window stride s. By varying w, the amount of information per window can be increased or decreased. s defines how much redundancy exists between adjacent windows. Increasing the value of w or decreasing the value of s provides the model with more information about the score, but will raise the computational complexity of the algorithm. Thus when determining the optimal parameters, a balance has to be struck between complexity and input coverage. As a rule of thumb, we use a w that is approximately twice the width of a notehead, and an s of half the value of w. This will ensure that each musical object is shown in full at least once in an input window. This gives a w of 64 pixels with a s of 32.

Convolutional neural network. To extract relevant features from the image patches, each patch is fed into a CNN. In this research, we keep the architecture of the CNN the same between different experiments, to ensure a fair comparison. First a *max-pooling* operation of 3×3 is applied on the input window for dimensionality reduction. Then, a convolutional layer of 32 5×5 kernels is applied, followed by a *relu* activation and 2×2 max-pooling operation. These three layers are repeated, and a fully-connected layer of 256 units with relu activation is applied, so each input for the encoder will be a vector of size 256.

Sequence-to-Sequence network. After extracting a vector description of each image patch, the sequence of vectors is fed into a Sequence-to-Sequence network [5,18]. This architecture consists of two RNN's. The first RNN, the *encoder*, encodes the full input sequence to a *fixed size representation*. The second RNN, the *decoder*, produces a sequence of outputs from the encoded representation. In the case of the OMR task, this sequence of outputs is the sequence of pitches and durations generated from the MusicXML files. For both encoder and decoder, a single *Long Short-Term Memory* (LSTM) [7] layer is used with 256 units. To predict both the pitch and duration, the output of the decoder is split into two separate output layers with a *softmax* activation and *categorical cross-entropy* loss.

A diagram of the full model is shown in Figure 2, where four input patches and output predictions are shown. On the left side, A sliding window is applied to a 2 bar score fragment. Each image patch is sequentially fed into the same CNN. This CNN is connected to the encoder network, creating a fixed size representation of the two input bars. The decoder uses this representation to produce the output sequence of (pitch, duration) pairs. Note that the second predicted pitch is an r pitch, representing a rest symbol.

Using the described configuration, the model has an input sequence length of 70 windows and an output sequence length of 48 units. Shorter output sequences are padded to this maximum length and the loss function is masked after last element of the sequence. The number of pitch categories is 108, and the number of duration categories is 48. In total, the model contains approximately 1.67 million parameters.

4.2 Training

Six separate models are trained, one on each of the proposed augmented data sets: No augmentations, AWGN, APN, Small ET, large ET and all augmentations. Augmentations are applied during training, and will be different each time the network is presented with a training sample. All models are trained with a batch-size of 64 using the ADAM optimizer [9], with an initial learning rate of $8*10^{-4}$ and a constant learning rate decay tuned so the rate is halved every ten epochs. Each model is trained to convergence, taking about 25 epochs on the non-augmented dataset. A single Nvidia Titan X Maxwell is used for training, which trains a model in approximately 30 hours.

4.3 Evaluation Metrics

On the evaluation data, three different metrics are calculated, similar to [3]:

- Pitch accuracy, the proportion of correctly predicted pitches.
- Duration accuracy, the proportion of correctly predicted note durations.
- Note accuracy, the proportion of predicted events where both pitch and duration are correctly predicted.

The accuracy is measured over all notes before the stop indicator, and the stop indicator is not included in the calculation of accuracy. The model is not given any a priori knowledge about how many notes are in the input fragment, so a wrong number of notes could be predicted. In case of a shorter predicted sequence, the missing notes are automatically marked as incorrect. If the predicted sequence is longer than the ground truth, the additional predicted notes are cut and only the notes within the length of the ground truth are used. This method of measuring accuracy is quite strict, as an insertion or omission of a note in the middle of a sequence could mean subsequent notes are all marked as incorrect. This should be kept in mind when evaluating the results of the model, and perhaps more descriptive metrics could be applied in future work.

5. RESULTS

5.1 Model Evaluation

The six trained models are evaluated on both a clean evaluation set, shown in Table 1, and augmented sets, shown in Table 2. The augmented evaluation sets are generated by applying the augmentations the model was trained on to the full clean evaluation set, with the parameters described in Section 3.2.

The model trained on data with all augmentations is compared against two commercially available methods in Table 3, similar to Shi et al. [16]. The comparison between the different methods on the clean dataset gives a baseline performance on digital scores, while the comparison on augmented data gives an indication of the difference of performance on real-world sheet music.

Training Augmentation	Pitch Accuracy	Duration Accuracy	Note Accuracy
None	0.79	0.92	0.76
AWGN	0.79	0.92	0.77
APN	**0.82**	0.91	0.79
ET - Small	0.78	0.91	0.76
ET - Large	0.79	**0.94**	0.78
All augmentations	0.81	**0.94**	**0.80**

Table 1. Measured accuracy on non-augmented scores. The accuracy scores for augmentations with the highest positive impact are in bold.

[2] https://www.capella-software.com/

Training Augmentation	Pitch Accuracy	Duration Accuracy	Note Accuracy
AWGN	0.79	0.90	0.75
APN	0.81	0.89	0.76
ET - Small	0.78	0.89	0.74
ET - Large	0.78	0.94	0.75
All augmentations	0.79	0.92	0.77

Table 2. Measured accuracies on scores with augmentations. Each model trained on different augmented data is evaluated on an evaluation set with corresponding augmentations.

Model	Clean	Augmented
Capella Scan 8 [2]	0.53	0.14
Photoscore 8 [3]	0.61	0.09
CS2S	0.80	0.77

Table 3. A comparison of accuracy between the proposed model (CS2S) and two popular commercially available methods.

5.2 Evaluation of Model Difficulties

To examine the difficulties the model has on different kinds of scores, three additional evaluations are performed on different subsets of the evaluation data.

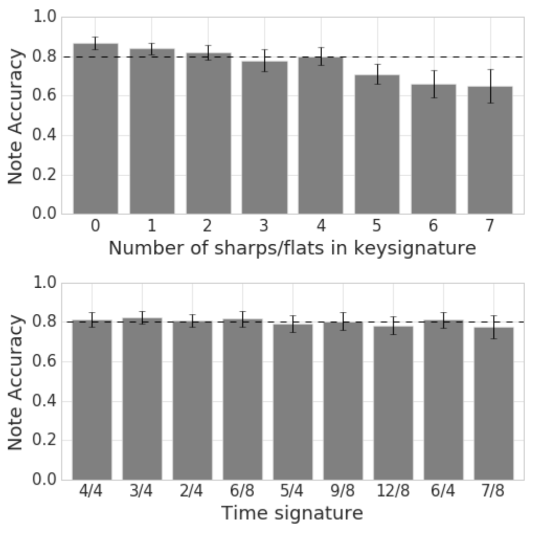


Figure 3. Top: The note-level accuracy for each number sharps/flats in the key signature. Bottom: The note-level accuracy for the most common time signatures. The dotted lines indicate the mean accuracy of 0.80.

First, an investigation into the impact of key signature on the note level accuracy on the non-augmented evaluation data is conducted. Just like human performance, the added complexity of many sharps or flats in the key signature of a fragment impacts the accuracy of the model. The results of this experiment are displayed in Figure 3 (top). At zero sharps or flats, the reported accuracy is 0.86,

[3] http://www.neuratron.com/

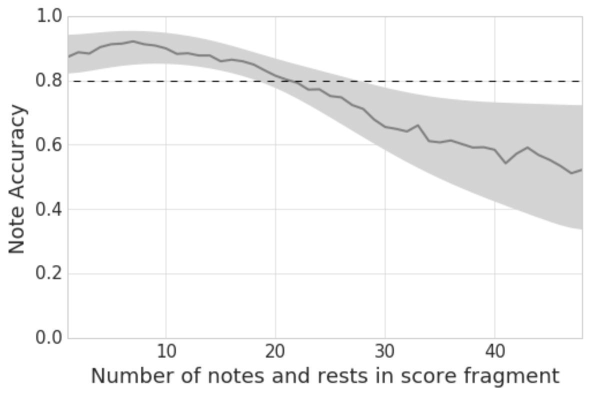


Figure 4. The mean note-level accuracy for each number of notes per fragment, with a confidence interval at a 95% level fit with a Gaussian Process.

achieving 0.06 higher than the mean accuracy of 0.80. With more than 4 sharps or flats in the key signature the note accuracy starts diminishing, down to a minimum of 0.66 for key signatures with seven sharps or flats.

Continuing, the nine most common time signatures and their accuracies are examined. While the output notation does not encode any direct information about time signature, the model could use structural information imposed by the time signature on the score to aid in note recognition. This evaluation will both look at if that is the case, and investigate which time signatures are potentially more difficult to transcribe. The results in Figure 3 (bottom) do not show a significant difference between the measured accuracy of different time signatures. The complex time signatures of $7/8$ and $5/4$ both are slightly less accurate, but this observation could be caused by a random deviation, or by features correlating with complex time signatures such as number of notes in a fragment.

As a final evaluation, we look at the correlation between number of notes in a fragment and accuracy. The model capacity and the representation between encoder and decoder are of a fixed size, which forces the model to represent more notes in the same space for fragments with a higher note density. This higher density could cause a loss in accuracy. Figure 4 shows clear evidence that this is the case; fragments containing more than 25 notes have a significantly lower accuracy than the measured mean.

6. DISCUSSION

We propose the Convolutional Sequence-to-Sequence model to deal with the difficulties OMR presents for learning systems. By using an end-to-end trainable sequential model, we completely move away from segmented symbol recognition, and perform the full OMR pipeline with a single algorithm. By incorporating Sequence-to-Sequence models into OMR, there are many new possibilities for obtaining development data. We view this aspect as the largest advantage the proposed method has over segmented models, as the acquisition of quality training data can be a limiting factor. The proposed model shows that it is robust to noisy input, an important quality for any OMR model. Additionally, the experiments show that it can deal with the large scale Elastic Transformations that essentially change the musical font. In future research, this aspect could be expanded to include handwritten notation.

A weakness of the model is pitch classification. Pooling operations introduce a degree of translation invariance, we hypothesize this invariance reduces the pitch recognition accuracy by discarding information about symbol position. However, omitting pooling operations from the model would greatly reduce the dimensionality reduction performed by the CNN. We propose incorporating a combination of convolutional layers and fully connected layers as a possible solution.

Furthermore, on more complex scores the model performs significantly worse. Both the number of sharps or flats in the key signature and the note density in the score fragment play a large role in the prediction accuracy. In future work, these problems could be addressed in multiple ways. A separate key signature recognition could be performed, and given as additional information to the model. This would take away some of the long range computations the key signature introduces and could improve the results on more complex scores.

The difficulty of translating long sequences with Sequence-to-Sequence models is a well studied problem [5, 18]. For longer sequences, the model needs to encode more information in the same fixed size representation, reducing the amount of storage available per note. A possible solution for this difficulty is proposed by Bahnadau et al. [2]: they replace the fixed size representation between encoder and decoder with an attention mechanism, a method that essentially performs a search function between the two networks. This mechanism has shown improvements to Sequence-to-Sequence models in neural machine translation, and could be used in the proposed method to alleviate some of the problems introduced with long sequences and long range dependencies.

The experiments performed in this research are exclusively on monophonic scores. The current representation of (pitch, duration) pairs does not allow for polyphonic note sequences, and in order to apply the model to polyphonic OMR tasks this representation needs to be adapted. A possible representation could be produced by using a method close like the MIDI-standard or piano roll representation.

Finally, we propose that the Convolutional Sequence-to-Sequence model could be applied to tasks outside of OMR that translate a spatial sequential representation to a sequence of labels. Within MIR, tasks like Automatic Music Transcription can be considered as such a task, where a representation of an audio signal is converted to a sequence of pitches and durations. Outside of MIR, tasks like video tagging or Optical Character Recognition are similar examples.

7. ACKNOWLEDGEMENTS

Many thanks to the reviewers for their valuable feedback, and to MuseScore for providing access to the material available in their archive. This research has been partially funded by Google.

8. REFERENCES

[1] Musescore. `https://musescore.org/`.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Donald Byrd and Megan Schindele. Prospects for improving omr with multiple recognizers. In *ISMIR*, pages 41–46, 2006.

[4] Jorge Calvo-Zaragoza, Antonio Pertusa, and Jose Oncina. Staff-line detection and removal using a convolutional neural network. *Machine Vision and Applications*, pages 1–10, 2017.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[6] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Anil K Jain and Douglas Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1390, 1997.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[10] Ken Perlin. Improving noise. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 681–682. ACM, 2002.

[11] Roberto M Pinheiro Pereira, Caio EF Matos, Geraldo Braz Junior, João DS de Almeida, and Anselmo C de Paiva. A deep approach for handwritten musical symbols recognition. In *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*, pages 191–194. ACM, 2016.

[12] Laurent Pugin. Optical music recognitoin of early typographic prints using hidden markov models. In *ISMIR*, pages 53–56, 2006.

[13] Laurent Pugin, John Ashley Burgoyne, and Ichiro Fujinaga. Map adaptation to improve optical music recognition of early music documents using hidden markov models. In *ISMIR*, pages 513–516, 2007.

[14] Ana Rebelo, G Capela, and Jaime S Cardoso. Optical recognition of music symbols. *International journal on document analysis and recognition*, 13(1):19–31, 2010.

[15] Ana Rebelo, Ichiro Fujinaga, Filipe Paszkiewicz, Andre RS Marcal, Carlos Guedes, and Jaime S Cardoso. Optical music recognition: state-of-the-art and open issues. *International Journal of Multimedia Information Retrieval*, 1(3):173–190, 2012.

[16] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[17] Patrice Y Simard, David Steinkraus, John C Platt, et al. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, volume 3, pages 958–962. Citeseer, 2003.

[18] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[19] Cuihong Wen, Ana Rebelo, Jing Zhang, and Jaime Cardoso. A new optical music recognition system based on combined neural network. *Pattern Recognition Letters*, 58:1–7, 2015.

RE-VISITING THE MUSIC SEGMENTATION PROBLEM WITH CROWDSOURCING

Cheng-i Wang
UCSD
chwl60@ucsd.edu

Gautham J. Mysore
Adobe Research
gmysore@adobe.com

Shlomo Dubnov
UCSD
sdubnov@ucsd.edu

ABSTRACT

Identifying boundaries in music structural segmentation is a well studied music information retrieval problem. The goal is to develop algorithms that automatically identify segmenting time points in music that closely matches human annotated data. The annotation itself is challenging due to its subjective nature, such as the degree of change that constitutes a boundary, the location of such boundaries, and whether a boundary should be assigned to a single time frame or a range of frames. Existing datasets have been annotated by small number of experts and the annotators tend to be constrained to specific definitions of segmentation boundaries. In this paper, we re-examine the annotation problem. We crowdsource the problem to a large number of annotators and present an analysis of the results. Our preliminary study suggests that although there is a correlation to existing datasets, this form of annotations reveals additional information such as stronger vs. weaker boundaries, gradual vs. sudden boundaries, and the difference in perception of boundaries between musicians and non-musicians. The study suggests that it could be worth re-defining certain aspects of the boundary identification in music structural segmentation problem with a broader definition.

1. INTRODUCTION

Music segmentation has been a fundamental task in automatic music content analysis. The task includes detecting boundaries between contiguous segments and labeling each detected segment within a music piece. In order to evaluate this task and train supervised machine learning algorithms, researchers have developed datasets that contain boundary timing and segment labeling annotations. In the majority of these datasets (such as *Beatles-TUT* [11], *CHARM Mazurka* [3] and *Beatles-ISO* [7], boundary timings are annotated by music experts and are defined as the time points that separate a music piece into non-overlapped contiguous sections representing meaningful song structures. These annotations provide clean-cut data for developing algorithms. During evaluation, the metrics are, in short, a measure of how close the automatically detected boundaries are to the ground truth annotations [6, 14].

Nevertheless, the instructions and rules for expert annotators to annotate these boundaries differs between datasets and the annotations inevitably conform to the subjective judgments of the annotators [13]. Moreover, the "one time point for one boundary" definition prevents the concept of short ambiguous/transitional/developing musical regions to be explored by researchers. To be more specific, it is established that different listeners will disagree on whether certain boundaries should exist or not in a music piece, and the saliences between boundaries might be different, while almost none of the existing datasets provide information about these intuitions [1, 13].

In [13], the issues of inter-annotator disagreement and the lack of multiple level annotations are discussed. The creation of the *SALAMI* dataset then attempts to solve such issues by having two versions of labels (by two annotators) and two levels (long and short time scale) of annotations in part of the dataset. The *SPAM* dataset also has five annotators with two levels of annotation for 50 songs [9]. In [8, 10], the issue of lacking support for hierarchical segmentation is discussed. Although the evaluation metrics for hierarchical segmentation problems are proposed in [8], only two datasets having two levels of hierarchy currently support this concept.

Apart from the inter-annotator agreement and between-boundaries saliency issues, the problem of not being able to model different types of boundaries is also an issue due to the current format of annotations. Sometimes the disagreements between annotators are not about whether one boundary should exist or not, but rather the timing of that boundary. The disagreement is likely because the exact change point or boundary between two larger segments is difficult to recognize if there are smaller transitional, pivotal, building, fading, or developing musical region connecting these two segments.

One of the major reasons for these limitations in existing datasets is the large amount of time and effort required by music experts to annotate songs. It tends to prevent datasets from having numerous (more than 5) annotators, and also tends to prevent detailed annotations for each song. One can alleviate the amount of effort required for annotating segmentation boundaries by crowdsourcing such task to the web. To the best of our knowledge, no methodology has been proposed utilizing crowdsourcing

Dataset	Song Name	Artist
Beatles-TUT	All You Need is Love Help! Here, There and Everywhere Strawberry Fields Forever Come Together(∗)	The Beatles
SALAMI	Smoke Machines	Atom Orr
	You Done Me Wrong	Cindy Woolf
	Out in the Cold	Carole King
	Black or White	Michael Jackson
	We will Rock You(∗)	Queen

Table 1. Song lists of the subsets from *Beatles-TUT* and *SALAMI*. Songs followed by asterisks are the hidden reference songs during the task.

to collect music segmentation boundaries. Crowdsourcing with Amazon's Mechanical Turks has been used to collect music similarity [4], music mood [5] data collection and audio sound quality evaluation [2].

In this paper, we present an preliminary study to support the above observations and address the above issues. We believe that this could lead to the creation of richer datasets with significantly less effort than previously required. In order to investigate the inter-annotator, between-boundary saliency problems and explore different types of segmentation boundaries, we used small subsets from existing datasets and annotate these via crowdsourcing. The results are a collection of annotations from at least 53 annotators (with at least 6 annotators annotated each song in full coverage) for each song for a total of 8 songs. The methodology of collecting annotations via crowdsourcing is described in section 2. The validation and analysis of the collected annotations are elaborated in section 3. Conclusions, and proposed future works are in section 4.

2. CROWDSOURCING

To perform the music segmentation boundary annotation collection task on the web, we use Amazon's Mechanical Turk (AMT). The proposed methodology is implemented as an extension of the **CAQE** (The Crowdsourced Audio Quality Evaluation Toolkit [1]) python package [2].

2.1 Data

Two subsets of songs from two music segmentation datasets were selected randomly to be annotated by the proposed methodology. The two datasets are the *Beatles-TUT* and *SALAMI* datasets. From each dataset, 5 songs were randomly selected. Among the 5 songs, 1 song is used as the hidden reference during the collection task to determine whether to accept or reject a person's annotations. Table 1 shows the 5 songs from both datasets.

2.2 Task Design

The typical off-line annotation process by music experts is to let them listen to a whole song and then annotate boundaries. This allows them to fine tune their annotations without time constraints. It also typically necessitates familiarity with an audio editing software package. Annotators on AMT, on the other hand, typically spend less time on such a task since their payment is fixed for a given task. They also typically have less (or no) experience with audio editing software packages. Therefore, the annotation process needs to be redesigned to accommodate it as an AMT based task. There are two goals of the redesign. The first goal is to simplify the annotation process so that AMT annotators could learn how to annotate quickly and repeat the process easily. The second goal is to maintain the quality of the annotations so that the results are informative and usable.

Since music structural segmentation is subjective in nature, we aim to not bias AMT annotators toward listening to specific musical cues. Therefore, the working definition used in the description of the task is kept as concise as possible and no musical terms are used. It is as follows:

> This listening task aims at collecting the **boundary timings** between parts of a song. During this task, you will be asked to listen for **when** a part of a song changes to another.

Rather than asking AMT annotators to listen to an entire song, we present them with short clips of music. For each clip, their task is to listen to the clip, determine if a boundary exists in the clip, and label the location of the boundary. We segment each song into clips of 20 seconds long with 10 seconds of overlap. The choice of 20 seconds is made according to the average length of segments defined by ground truth annotations in existing datasets as reported in [12]. The annotator is only given the option of labeling a single boundary and asked to choose the strongest boundary if they hear more than one. They also have the option of choosing no boundary.

The goal of the user interface design is to simplify the task and not bias the annotators to any clues apart from what is heard in the clip. To ensure that the annotator listened to the full clip and made an annotation decision before going to the next clip, all buttons and sliders are disabled except the Play/Pause button until the first full playback of the clip. The annotator uses the Play/Pause button and audio progress bar to listen to and navigate the clip. The annotation is done by clicking on the boundary selection slider. The darker green area surrounding the clicked location on the boundary selection slider indicates the playback region for the check selection button. The annotator has the option to simply not choose a boundary if they do not hear one. The next trial button is disabled until the annotator clicks on the change heard (submit current clicked location on the boundary selection slider) or no change heard button. A snapshot of the user interface is shown in Figure 1.

Each boundary collection task ("HIT" in AMT's terms) contains 10 clips, with 9 clips randomly selected from all non-hidden reference songs and 1 clip from the hidden reference song. The randomization of selected clips is designed such that annotators will not be presented with overlapping clips within one task (10 clips) and will cover the

[1] `https://github.com/interactiveaudiolab/CAQE`

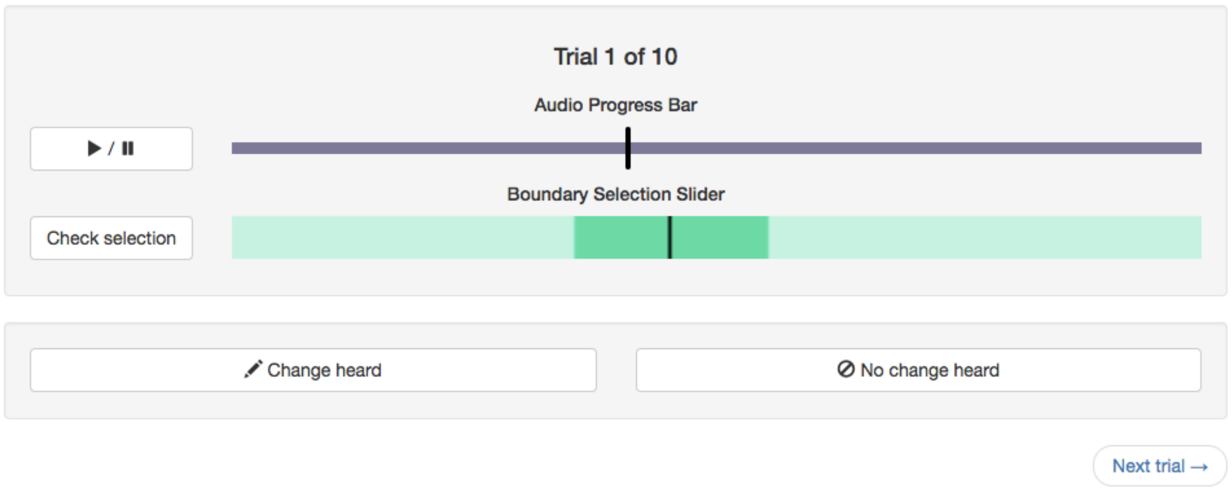


Figure 1. The user interface for the AMT annotating task. The annotator uses the Play/Pause button and audio progress bar to listen to and navigate the clip. The annotation is done by clicking on the boundary selection slider. The darker green area surrounding the clicked location on the boundary selection slider indicates the playback region for the check selection button.

full range of all songs once and once only if they finish all the tasks available to them. The order of the 10 clips within each task is also randomized. In order to further make sure that each song is covered with enough annotations, the annotation tasks are divided into two batches according to the two dataset subsets, meaning one batch for *Beatles-TUT* and one batch for *SALAMI*, and are collected separately. The batch for *Beatles-TUT* has 10 tasks (100 clips) and the batch for *SALAMI* has 8 tasks (80 clips) for each annotator.

It is mentioned in the previous section that out of the 5 songs, there is 1 song selected as a hidden reference acting as a quality check after collecting boundary annotations from the AMT annotators. To use this song as a quality check, a few clips from the song that have clear and obvious boundary regions are selected and manually annotated by the authors. The authors avoided using the ground truth annotations from the original datasets since the goal for the task is not to identify the "correct" boundaries defined by the original datasets, but rather simply annotating reasonable boundaries (that might differ from the ground truth annotations from the original datasets).

In order to take this concern into account, multiple boundary candidates for each hidden reference clip are allowed so the quality check accepts wider and reasonable results than just using ground truth annotations (only one boundary annotation for each clip) from the original datasets. After collecting boundary annotations from an AMT annotator, their boundary annotations on the hidden reference clips are compared against the author's annotations on the same clips. If the average distance between AMT annotator's boundary annotation to the closest annotation by the author for each clip is less than 3 seconds, all of the AMT annotator's annotations are accepted, otherwise they are rejected and not used for the analysis. The annotations by the authors for the hidden references could be found in the code repository [2].

In order to determine if the annotator is following basic instructions and listening on a device (speakers/headphones) with a sufficient frequency response, we insert a hearing screening before the task begins, as done in [2]. After the completion of the task, the annotator is presented with a post task survey gathering demographical, musical background, and qualitative information. For the musical background, the AMT annotator is asked to answer if they consider themself to be a musician. We also ask for qualitative feedback on the task and what the annotators were listening for.

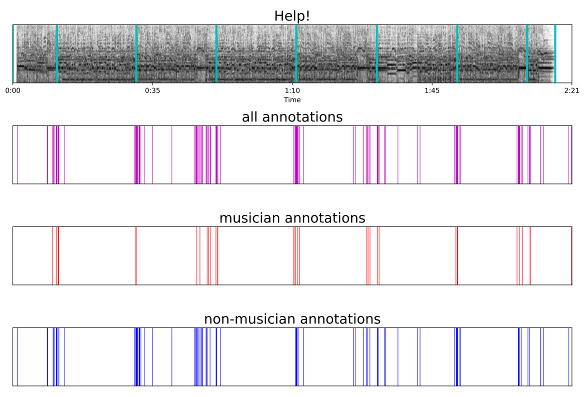
3. ANALYSIS

We consolidated the annotations from the individual clips so that we have all annotations of a given song on a common timeline. Since there is an overlap of 10 seconds between consecutive clips, there is a chance that the same boundary will be labeled twice by the same annotator (cases in which the time difference between the two annotations are less than 3 seconds). In such cases, we simply randomly discarded one of the annotations.

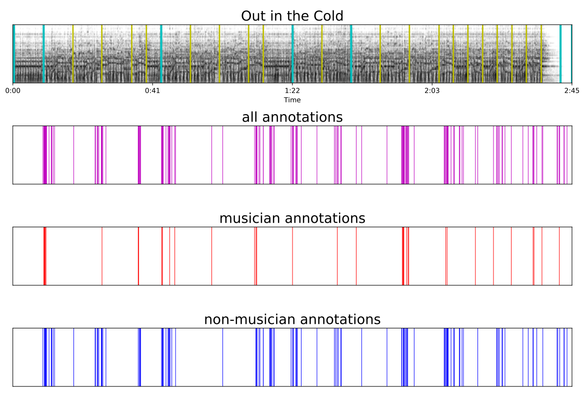
Two example songs showing the annotations from the AMT annotators along with a comparison to the ground truth annotations from the original datasets are shown in Figure 2. A correlation can be seen between the two.

In Table 2 and Table 3, overall statistics of the collected annotations and AMT annotators are shown. Though the task is only exercised on subsets of existing datasets, the number of AMT annotators and annotations easily outnumbered those of existing datasets. This observation is true even when considering only the statistics from AMT annotators that are self-identified musicians (numbers in

[2] https://github.com/wangsix/caqe_segmentation

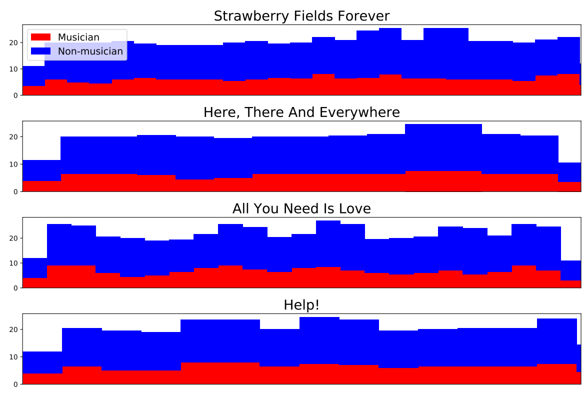


(a) Help! - Beatles

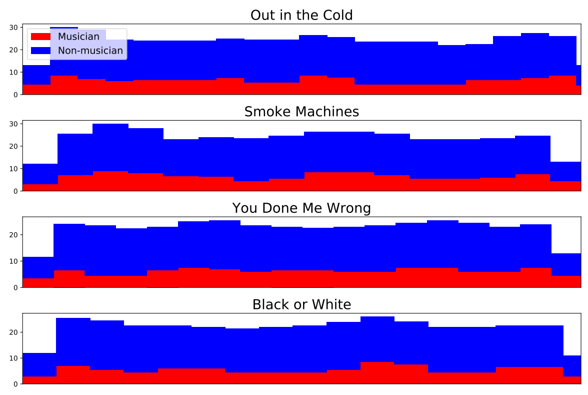


(b) Out in the Cold - Carole King

Figure 2. Two example songs with their annotations from selected subsets. The first row of each subplot is the CQT-spectrogram of the example song. The light blue lines in this row are the ground truth annotated by music experts. The yellow lines in "Out in the Cold" are the lower level ground truth by music experts. The vertical lines in the rest of the rows represent annotations from all AMT annotators, musician annotators and non-musician annotators respectively.



(a) Beatles subset



(b) Salami subset

Figure 3. Histograms of annotation count along the time line of each song. It shows a roughly evenly distributed coverage for each song being annotated.

parentheses in Table 2 and Table 3). It is also true if only complete annotations from single annotator are considered.

The seemingly low average coverage rate of each song by each annotator (right most column of Table 3) is a natural result of distributing clips randomly to AMT annotators throughout the AMT tasks. Even with the randomized task distribution, there are still at least 6 completed annotations for each song (2nd right column of Table 3).

The counts of annotations for each clip in each song are shown in Figure 3. From the histograms it could be observed that every song is fully covered by multiple annotators in a more or less evenly distributed manner. The beginning and ending of songs have less accumulated counts since they are not fully covered by overlaps.

3.1 Validation

To validate the collected annotations with the proposed methodology, the aggregated annotations for one song are treated as one segmentation boundary prediction from an arbitrary algorithm and compared to the ground truth boundary annotations from the original datasets.

The annotated timings of a song via crowdsourcing are first discretized into a binary vector with ones representing the presence of annotated boundaries. The discretization is done with a sampling rate of 22050Hz and 512 sample hop size. Then the binary vector of each song is normalized by its annotation count histogram (Figure 3) to account for different number of times each time region is annotated. After the normalization, a Gaussian window with 0.5 second standard deviation is convolved with the binary vectors obtaining a boundary detection function for each song. The boundary detection functions are renormalized to be between $[0, 1]$. A simple peak-picking strategy using the same principle as [12] with 0.5s window and 0.1 threshold is applied to select segmentation boundaries from the boundary detection function. These functions are shown in Figure 4 with the ground truth plotted against them.

The validation of such predictions are done using the standard MIREX 3 seconds structural music segmentation boundary accuracy evaluation metric. The validation results are shown in Table 4. From the F-measures and recall rate, it can be observed that the aggregated results from AMT based annotations in general agree with the music

	Annotators			Accepted Annotations	
Datasets	Accepted(musician)	Rejected	Invalid	Total	Per Annotator
Beatles-TUT	61(17)	11	89	1468	24.06
SALAMI	61(14)	41	65	1652	27.08

Table 2. Statistics of the AMT annotators and their annotations. Accepted annotators are the ones that passed the hidden reference quality check. Rejected annotators are the ones that failed the hidden reference quality check. Invalid annotators are the ones that did not pass the hearing screening or failed to submit their results.

Song	Count(musician)			Avg. Coverage
	Annotators	Annotations	Complete	Per Annotator(%)
All You Need is Love	61(17)	452(131)	6(3)	28.53
Help!	53(15)	275(80)	8(3)	30.69
Here, There and Everywhere	53(13)	264(75)	8(3)	29.93
Strawberry Fields Forever	59(16)	477(133)	7(3)	27.25
Smoke Machines	58(14)	376(103)	13(4)	25.31
You Done Me Wrong	61(14)	405(110)	12(4)	23.35
Black or White	61(14)	502(131)	14(5)	24.6
Out in the Cold	58(14)	369(92)	13(3)	23.7

Table 3. Song statistics from accepted annotations. The numbers in parentheses in columns 2 and 3 (from the left) are the numbers for self-identified musicians. The 4th column is the number of complete annotations by one annotator. The average coverage of a song per annotator (5th column) is calculated by dividing the average number of annotated clips per annotator for a given song by the total number of clips for that song.

experts annotating the original datasets. Also the self-identified musicians performed better than non-musicians in 7 cases out of 8. There might be two reasons for the higher recall rates. One reason is a potential bias due to the 20 seconds length clip. The other reason is that some of the peaks representing different levels of saliency or confidence (height of the boundary detection function), resulted in more peaks than the single level annotations by the music experts.

3.2 Inter-Annotator Analysis

In [9], the problem of subjectivity in music structural segmentation problem is studied by showing annotator effects with the two-way ANOVA factor analysis. The same analysis approach can not be applied here since the sets of annotators annotating each song are different (with overlapped annotators). In order to analyze the degree of agreement between AMT annotators, a simple measurement is devised. The agreement degree a_i of one annotation i of a clip to other annotations i' of the same clip is defined as

$$a_i = \frac{\sum_{i' \in I, i' \neq i} [1 \text{ if } i \text{ agrees with } i']}{\text{Number of items in } I}. \quad (1)$$

where I is the set of annotations of a clip annotated by all annotators. The agreement of one annotation i to another i' is established if the annotated timing of i is within 3 seconds of i'' annotated timing, or if both i and i' has empty annotation. a_i is a value between $[0, 1]$ and could be thought of as the probability of one annotation agrees with other annotations in the same 20 seconds region. Since every annotator could only annotate a clip once, Equation 1 becomes a measurement of agreement between annotators. The agreement between annotators of one song could then be measured by calculating the average of all as over one song. The inter-annotator agreement of each song is shown in Table 5. From Table 5, one can observe that although the average agreement between annotators is above 70%, the standard deviations show that the agreement between annotators is not consistent throughout the song but varying a lot from time to time within a song. This observation supports the propositions made in [10] that multiple human annotations should be used during evaluation to take human subjectivity into account.

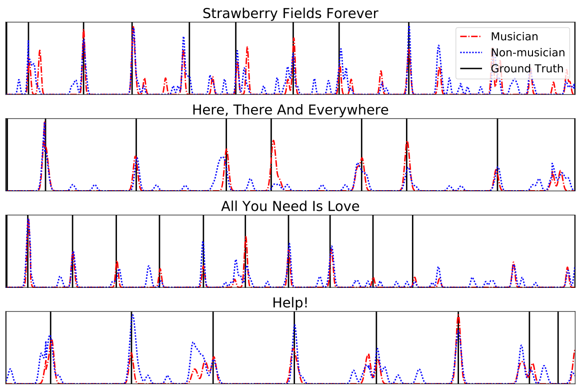
3.3 Boundaries Investigation

By qualitatively examining the AMT annotations and listening to the corresponding regions, it is evident that using a single time stamp representing the boundary between segmentations is inadequate. For example, the ground truth segmentation boundary around 0:50 in the song *Help!* by the Beatles is annotated at the beginning of the second verse when the lyrics start, but the AMT based annotations were spread across the region from 0:46 to 0:50 synchronizing with the sentence "Would you please, please, help me?". Musically, it could be correct to say that that region acts as the transition between two larger segments and the single boundary at 0:50 is inadequate to represent this musical property since the boundary is actually a prolonged region. This observation suggests that there exist different types of boundaries, with the easiest categorization being a clear-cut one versus a smooth/prolonged one. The AMT annotations of *Help!* are shown in Figure 2(a).

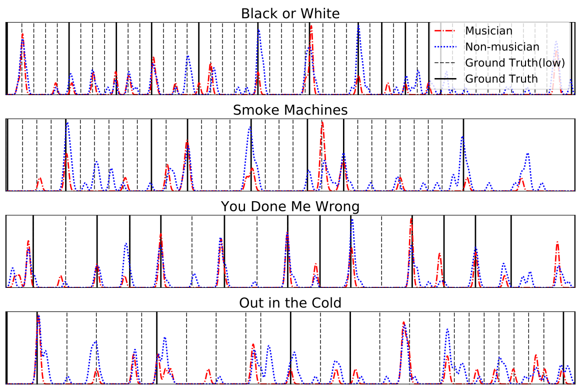
The other qualitative assessment is that the boundary detection function mentioned in section 3.1 shows that the hierarchical nature of structural segmentation boundaries exist and could be measured by the relative votes a boundary has compared to other boundaries in the same song. The boundary detection function also suggests that instead of having a discretized hierarchical representation of structural segmentation boundaries, a continuous version where

Song	Musician			Non-musician		
	Precision	Recall	F-measure	Precision	Recall	F-measure
All You Need is Love	0.73	0.85	0.79	0.5	0.69	0.58
Help!	0.5	0.78	0.61	0.33	0.89	0.48
Here, There and Everywhere	0.81	0.9	0.86	0.61	0.8	0.7
Strawberry Fields Forever	0.36	0.72	0.48	0.25	0.9	0.4
Smoke Machines	0.5	0.7	0.59	0.38	0.9	0.53
You Done Me Wrong	0.76	0.87	0.81	0.73	0.93	0.82
Black or White	0.67	0.8	0.72	0.33	0.73	0.46
Out in the Cold	0.39	0.88	0.54	0.2	0.75	0.32

Table 4. Standard 3-seconds precision, recall and F-measure evaluation metrics on the AMT annotator's annotation against ground truth from original datasets.



(a) Beatles subset



(b) Salami subset

Figure 4. The boundary detection functions obtained from AMT annotations against ground truth by music experts.

the saliency or confidence of boundaries is represented by a continuous curve might be another intuitive choice in terms of evaluating boundary detection algorithms.

4. FUTURE WORK AND CONCLUSION

In this paper, a methodology utilizing crowdsourcing for collecting alternative ground truth data for structural segmentation boundaries is proposed and validated. This methodology provides opportunities for researchers to create new segmentation boundary datasets in a fast and efficient way. To create a dataset with the proposed methodology, one has to make sure not to bias annotators toward specific aural cues and manage the distribution of clips so that annotators work on evenly distributed clips between songs and songs get evenly distributed annotations from all annotators.

Song	Avg. agreement (Std.)
All You Need is Love	0.71 (0.29)
Help!	0.75 (0.26)
Here, There and Everywhere	0.65 (0.28)
Strawberry Fields Forever	0.72 (0.29)
Smoke Machines	0.72 (0.28)
You Done Me Wrong	0.72 (0.28)
Out in the Cold	0.76 (0.31)
Black or White	0.71 (0.31)

Table 5. The average and standard deviation of inter-annotator agreement of each song.

As suggested in section 3.3, different types of boundaries exist and could be investigated given the kind of data collected by the methodology proposed in this work. These boundary types could be categorized. Also the different types of musical cues that lead to the perception of music segmentation boundaries could be investigated by another round of crowdsourcing on the annotated data focusing on surveying the reasoning behind each annotations.

5. ACKNOWLEDGMENT

This work is supported by CREL (Center for Research in Entertainment and Learning at UCSD) and Adobe Research. Special thanks to Dr. Mark Cartwright (NYU) and Dr. Ian Wehrman (Adobe Research) for their kind help.

6. REFERENCES

[1] Michael J Bruderer, Martin F Mckinney, and Armin Kohlrausch. The perception of structural boundaries in melody lines of western popular music. *Musicae Scientiae*, 13(2):273–313, 2009.

[2] Mark Cartwright, Bryan Pardo, Gautham J Mysore, and Matt Hoffman. Fast and easy crowdsourced perceptual audio evaluation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 619–623. IEEE, 2016.

[3] Nicholas Cook. Performance analysis and chopin's mazurkas. *Musicae scientiae*, 11(2):183–207, 2007.

[4] Jin Ha Lee. Crowdsourcing music similarity judgments using mechanical turk. In *ISMIR*, pages 183–188, 2010.

[5] Jin Ha Lee and Xiao Hu. Generating ground truth for music mood classification using mechanical turk. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 129–138. ACM, 2012.

[6] Mark Levy and Mark Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE transactions on audio, speech, and language processing*, 16(2):318–326, 2008.

[7] Matthias Mauch, Chris Cannam, Matthew Davies, Simon Dixon, Christopher Harte, Sefki Kolozali, Dan Tidhar, and Mark Sandler. Omras2 metadata project 2009. In *Proc. of 10th International Conference on Music Information Retrieval*, page 1, 2009.

[8] Brian McFee, Oriol Nieto, and Juan Pablo Bello. Hierarchical evaluation of segment boundary detection. In *ISMIR*, pages 406–412, 2015.

[9] Oriol Nieto. *Discovering structure in music: Automatic approaches and perceptual evaluations*. PhD thesis, PhD thesis, New York University, 2015.

[10] Oriol Nieto. 4.7 approaching the ambiguity problem of computational structure segmentation. *Computational Music Structure Analysis*, page 177, 2016.

[11] Jouni Paulus and Anssi Klapuri. Music structure analysis by finding repeated parts. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 59–68. ACM, 2006.

[12] Jean Serra, Mathias Muller, Peter Grosche, and Josep Ll Arcos. Unsupervised music structure annotation by time series structure features and segment similarity. *Multimedia, IEEE Transactions on*, 16(5):1229–1240, 2014.

[13] Jordan Bennett Louis Smith, John Ashley Burgoyne, Ichiro Fujinaga, David De Roure, and J Stephen Downie. Design and creation of a large-scale database of structural annotations. In *ISMIR*, volume 11, pages 555–560, 2011.

[14] Douglas Turnbull, Gert RG Lanckriet, Elias Pampalk, and Masataka Goto. A supervised approach for detecting boundaries in music using difference features and boosting. In *ISMIR*, pages 51–54, 2007.

SINGING VOICE SEPARATION WITH DEEP U-NET CONVOLUTIONAL NETWORKS

Andreas Jansson[1, 2], **Eric Humphrey**[2], **Nicola Montecchio**[2],
Rachel Bittner[2], **Aparna Kumar**[2], **Tillman Weyde**[1]
[1]City, University of London, [2]Spotify
`{andreas.jansson.1, t.e.weyde}@city.ac.uk`
`{ejhumphrey, venice, rachelbittner, aparna}@spotify.com`

ABSTRACT

The decomposition of a music audio signal into its vocal and backing track components is analogous to image-to-image translation, where a mixed spectrogram is transformed into its constituent sources. We propose a novel application of the *U-Net* architecture — initially developed for medical imaging — for the task of source separation, given its proven capacity for recreating the fine, low-level detail required for high-quality audio reproduction. Through both quantitative evaluation and subjective assessment, experiments demonstrate that the proposed algorithm achieves state-of-the-art performance.

1. INTRODUCTION

The field of Music Information Retrieval (MIR) concerns itself, among other things, with the analysis of music in its many facets, such as melody, timbre or rhythm [20]. Among those aspects, popular western commercial music ("pop" music) is arguably characterized by emphasizing mainly the Melody and Accompaniment aspects; while this is certainly an oversimplification in the context of the whole genre, we restrict the focus of this paper to the analysis of music that lends itself well to be described in terms of a main melodic line (foreground) and accompaniment (background) [27]. Normally the melody is sung, whereas the accompaniment is performed by one or more instrumentalists; a singer delivers the lyrics, and the backing musicians provide harmony as well as genre and style cues [29].

The task of automatic singing voice separation consists of estimating what the sung melody and accompaniment would sound like in isolation. A clean vocal signal is helpful for other related MIR tasks, such as singer identification [18] and lyric transcription [17]. As for commercial applications, it is evident that the karaoke industry, estimated to be worth billions of dollars globally [4], would directly benefit from such technology.

2. RELATED WORK

Several techniques have been proposed for blind source separation of musical audio. Successful results have been achieved with non-negative matrix factorization [26, 30, 32], Bayesian methods [21], and the analysis of repeating structures [23].

Deep learning models have recently emerged as powerful alternatives to traditional methods. Notable examples include [25] where a deep feed-forward network learns to estimate an ideal binary spectrogram mask that represents the spectrogram bins in which the vocal is more prominent than the accompaniment. In [9] the authors employ a deep recurrent architecture to predict soft masks that are multiplied with the original signal to obtain the desired isolated source.

Convolutional encoder-decoder architectures have been explored in the context of singing voice separation in [6] and [8]. In both of these works, spectrograms are compressed through a bottleneck layer and re-expanded to the size of the target spectrogram. While this "hourglass" architecture is undoubtedly successful in discovering global patterns, it is unclear how much local detail is lost during contraction.

One potential weakness shared by the papers cited above is the lack of large training datasets. Existing models are usually trained on hundreds of tracks of lower-than-commercial quality, and may therefore suffer from poor generalization. In this work we aim to mitigate this problem using weakly labeled professionally produced music tracks.

Over the last few years, considerable improvements have occurred in the family of machine learning algorithms known as image-to-image translation [11] — pixel-level classification [2], automatic colorization [33], image segmentation [1] — largely driven by advances in the design of novel neural network architectures.

This paper formulates the voice separation task, whose domain is often considered from a time-frequency perspective, as the translation of a mixed spectrogram into vocal and instrumental spectrograms. By using this framework we aim to make use of some of the advances in image-to-image translation — especially in regard to the reproduction of fine-grained details — to advance the state-of-the-art of blind source separation for music.

3. METHODOLOGY

This work adapts the *U-Net* [24] architecture to the task of vocal separation. The architecture was introduced in biomedical imaging, to improve precision and localization of microscopic images of neuronal structures. The architecture builds upon the *fully convolutional network* [14] and is similar to the *deconvolutional network* [19]. In a deconvolutional network, a stack of convolutional layers — where each layer halves the size of the image but doubles the number of channels — encodes the image into a small and deep representation. That encoding is then decoded to the original size of the image by a stack of upsampling layers.

In the reproduction of a natural image, displacements by just one pixel are usually not perceived as major distortions. In the frequency domain however, even a minor linear shift in the spectrogram has disastrous effects on perception: this is particularly relevant in music signals, because of the logarithmic perception of frequency; moreover, a shift in the time dimension can become audible as jitter and other artifacts. Therefore, it is crucial that the reproduction preserves a high level of detail. The U-Net adds additional skip connections between layers at the same hierarchical level in the encoder and decoder. This allows low-level information to flow directly from the high-resolution input to the high-resolution output.

3.1 Architecture

The goal of the neural network architecture is to predict the vocal and instrumental components of its input indirectly: the output of the final decoder layer is a soft mask that is multiplied element-wise with the mixed spectrogram to obtain the final estimate. Figure 1 outlines the network architecture. In this work, we choose to train two separate models for the extraction of the instrumental and vocal components of a signal, to allow for more divergent training schemes for the two models in the future.

3.1.1 Training

Let X denote the magnitude of the spectrogram of the original, mixed signal, that is, of the audio containing both vocal and instrumental components. Let Y denote the magnitude of the spectrograms of the target audio; the latter refers to either the vocal (Y_v) or the instrumental (Y_i) component of the input signal.

The loss function used to train the model is the $L_{1,1}$ norm [1] of the difference of the target spectrogram and the masked input spectrogram:

$$L(X, Y; \Theta) = ||f(X, \Theta) \odot X - Y||_{1,1} \tag{1}$$

where $f(X, \Theta)$ is the output of the network model applied to the input X with parameters Θ – that is the mask generated by the model.

[1] The $L_{1,1}$ norm of a matrix is simply the sum of the absolute values of its elements.

Two U-Nets, Θ_v and Θ_i, are trained to predict vocal and instrumental spectrogram masks, respectively.

3.1.2 Network Architecture Details

Our implementation of U-Net is similar to that of [11]. Each encoder layer consists of a strided 2D convolution of stride 2 and kernel size 5x5, batch normalization, and leaky rectified linear units (ReLU) with leakiness 0.2. In the decoder we use strided deconvolution (sometimes referred to as transposed convolution) with stride 2 and kernel size 5x5, batch normalization, plain ReLU, and use 50% dropout to the first three layers, as in [11]. In the final layer we use a sigmoid activation function. The model is trained using the ADAM [12] optimizer.

Given the heavy computational requirements of training such a model, we first downsample the input audio to 8192 Hz in order to speed up processing. We then compute the Short Time Fourier Transform with a window size of 1024 and hop length of 768 frames, and extract patches of 128 frames (roughly 11 seconds) that we feed as input and targets to the network. The magnitude spectrograms are normalized to the range $[0, 1]$.

3.1.3 Audio Signal Reconstruction

The neural network model operates exclusively on the magnitude of audio spectrograms. The audio signal for an individual (vocal/instrumental) component is rendered by constructing a spectrogram: the output magnitude is given by applying the mask predicted by the U-Net to the magnitude of the original spectrum, while the output phase is that of the original spectrum, unaltered. Experimental results presented below indicate that such a simple methodology proves effective.

3.2 Dataset

As stated above, the description of the model architecture assumes that training data was available in the form of a triplet (original signal, vocal component, instrumental component). Unless one is in the extremely fortunate position as to have access to vast amounts of unmixed multi-track recordings, an alternative strategy has to be found in order to train a model like the one described.

A solution to the issue was found by exploiting a specific but large set of commercially available recordings in order to “construct” training data: *instrumental versions* of recordings.

It is not uncommon for artists to release instrumental versions of tracks along with the original mix. We leverage this fact by retrieving pairs of (original, instrumental) tracks from a large commercial music database. Candidates are found by examining the metadata for tracks with matching duration and artist information, where the track title (fuzzily) matches except for the string “Instrumental” occurring in exactly one title in the pair. The pool of tracks is pruned by excluding exact content matches. Details about the construction of this dataset can be found in [10].

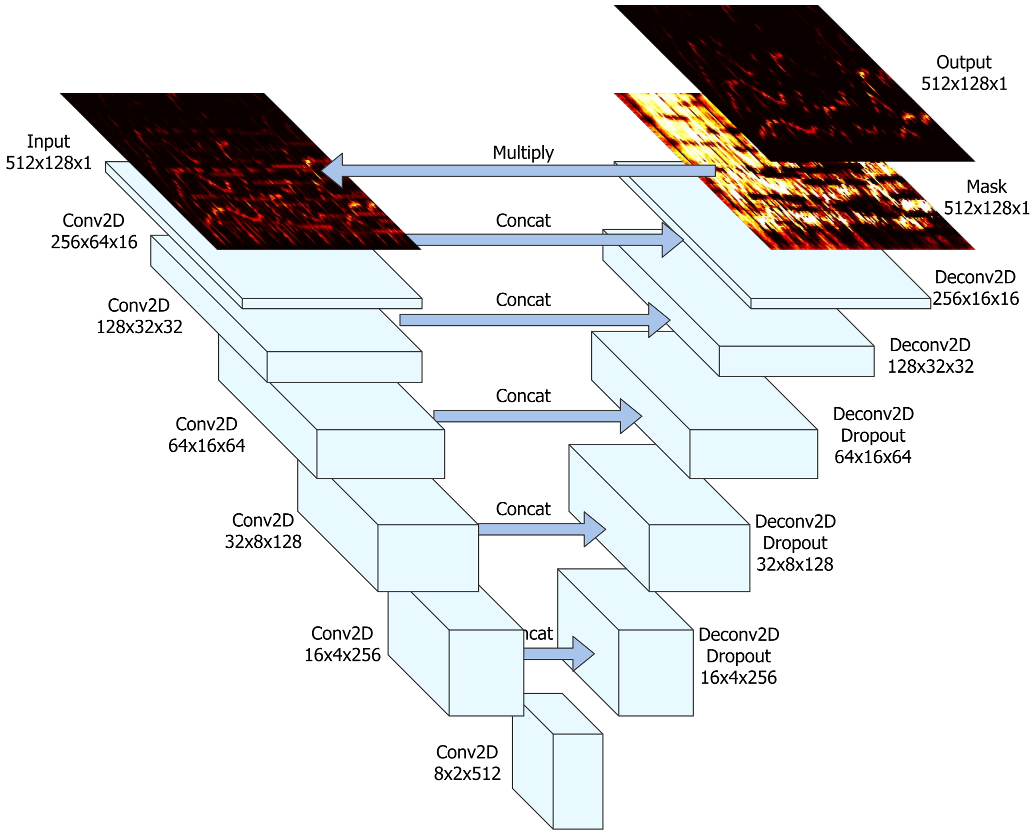


Figure 1. Network Architecture

Genre	Percentage
Pop	26.0%
Rap	21.3%
Dance & House	14.2%
Electronica	7.4%
R&B	3.9%
Rock	3.6%
Alternative	3.1%
Children's	2.5%
Metal	2.5%
Latin	2.3%
Indie Rock	2.2%
Other	10.9%

Table 1. Training data genre distribution

The above approach provides a large source of X (mixed) and Y_i (instrumental) magnitude spectrogram pairs. The vocal magnitude spectrogram Y_v is obtained from their half-wave rectified difference. A qualitative analysis of a large handful of examples showed that this technique produced reasonably isolated vocals.

The final dataset contains approximately 20,000 track pairs, resulting in almost two months worth of continuous audio. To the best of our knowledge, this is the largest training data set ever applied to musical source separation. Table 1 shows the relative distribution of the most frequent genres in the dataset, obtained from the catalog metadata.

4. EVALUATION

We compare the proposed model to the *Chimera* model [15] that produced the highest evaluation scores in the 2016 MIREX Source Separation campaign[2]; we make use of their web interface[3] to process audio clips. It should be noted that the Chimera web server is running an improved version of the algorithm that participated in MIREX, using a hybrid "multiple heads" architecture that combines deep clustering with a conventional neural network [16].

For evaluation purposes we built an additional baseline model; it resembles the U-Net model but without the skip connections, essentially creating a convolutional encoder-decoder, similar to the "Deconvnet" [19].

We evaluate the three models on the standard iKala [5] and MedleyDB dataset [3]. The iKala dataset has been used as a standardized evaluation for the annual MIREX campaign for several years, so there are many existing results that can be used for comparison. MedleyDB on the other hand was recently proposed as a higher-quality, commercial-grade set of multi-track stems. We generate isolated instrumental and vocal tracks by weighting sums of instrumental/vocal stems by their respective mixing co-

[2] `www.music-ir.org/mirex/wiki/2016:Singing_Voice_Separation_Results`
[3] `danetapi.com/chimera`

	U-Net	Baseline	Chimera
NSDR Vocal	**11.094**	8.549	8.749
NSDR Instrumental	**14.435**	10.906	11.626
SIR Vocal	**23.960**	20.402	21.301
SIR Instrumental	**21.832**	14.304	20.481
SAR Vocal	**17.715**	15.481	15.642
SAR Instrumental	**14.120**	12.002	11.539

Table 2. iKala mean scores

	U-Net	Baseline	Chimera
NSDR Vocal	**8.681**	7.877	6.793
NSDR Instrumental	**7.945**	6.370	5.477
SIR Vocal	**15.308**	14.336	12.382
SIR Instrumental	**21.975**	16.928	20.880
SAR Vocal	**11.301**	10.632	10.033
SAR Instrumental	**15.462**	15.332	12.530

Table 3. MedleyDB mean scores

Model	Mean	SD	Min	Max	Median
U-Net	**14.435**	3.583	4.165	21.716	**14.525**
Baseline	10.906	3.247	1.846	19.641	10.869
Chimera	11.626	4.151	-0.368	20.812	12.045
LCP2	11.188	3.626	2.508	19.875	11.000
LCP1	10.926	3.835	0.742	19.960	10.800
MC2	9.668	3.676	-7.875	22.734	9.900

Table 4. iKala NSDR Instrumental, MIREX 2016

Model	Mean	SD	Min	Max	Median
U-Net	**11.094**	3.566	2.392	20.720	**10.804**
Baseline	8.549	3.428	-0.696	18.530	8.746
Chimera	8.749	4.001	-1.850	18.701	8.868
LCP2	6.341	3.370	-1.958	17.240	5.997
LCP1	6.073	3.462	-1.658	17.170	5.649
MC2	5.289	2.914	-1.302	12.571	4.945

Table 5. iKala NSDR Vocal, MIREX 2016

efficients as supplied by the MedleyDB Python API[4]. We limit our evaluation to clips that are known to contain vocals, using the melody transcriptions provided in both iKala and MedleyDB.

The following functions are used to measure performance: Signal-To-Distortion Ratio (SDR), Signal-to-Interference Ratio (SIR), and Signal-to-Artifact Ratio (SAR) [31]. Normalized SDR (NSDR) is defined as

$$\text{NSDR}(S_e, S_r, S_m) = \text{SDR}(S_e, S_r) - \text{SDR}(S_m, S_r) \quad (2)$$

where S_e is the estimated isolated signal, S_r is the reference isolated signal, and S_m is the mixed signal. We compute performance measures using the *mir_eval* toolkit [22].

Table 2 and Table 3 show that the U-Net significantly outperforms both the baseline model and Chimera on all three performance measures for both datasets. In Figure 2 we show an overview of the distributions for the different evaluation measures.

Assuming that the distribution of tracks in the iKala hold-out set used for MIREX evaluations matches those in the public iKala set, we can compare our results to the participants in the 2016 MIREX Singing Voice Separation task.[5] Table 4 and Table 5 show NSDR scores for our models compared to the best performing algorithms of the 2016 MIREX campaign.

In order to assess the effect of the U-Net's skip connections, we can visualize the masks generated by the U-Net and baseline models. From Figure 3 it is clear that while the baseline model captures the overall structure, there is a lack of fine-grained detail observable.

4.1 Subjective Evaluation

Emiya et al. introduced a protocol for the subjective evaluation of source separation algorithms [7]. They suggest asking human subjects four questions that broadly correspond to the SDR/SIR/SAR measures, plus an additional question regarding the overall sound quality.

As we asked these four questions to subjects without music training, our subjects found them ambiguous, e.g., they had problems discerning between the absence of artifacts and general sound quality. For better clarity, we distilled the survey into the following two questions in the vocal extraction case:

- Quality: "Rate the vocal quality in the examples below."
- Interference: "How well have the instruments in the clip above been removed in the examples below?"

For instrumental extraction we asked similar questions:

- Quality: "Rate the sound quality of the examples below relative to the reference above."
- Extracting instruments: "Rate how well the instruments are isolated in the examples below relative to the full mix above."

Data was collected using CrowdFlower[6], an online platform where humans carry out micro-tasks, such as image classification, simple web searches, etc., in return for small per-task payments.

In our survey, CrowdFlower users were asked to listen to three clips of isolated audio, generated by U-Net, the baseline model, and Chimera. The order of the three clips was randomized. Each question asked one of the Quality and Interference questions. In the Interference question we also included a reference clip. The answers were given according to a 7 step Likert scale [13], ranging from "Poor" to "Perfect". Figure 4 is a screen capture of a CrowdFlower question.

[4] github.com/marl/medleyDB

[5] http://www.music-ir.org/mirex/wiki/2016:Singing_Voice_Separation_Results

[6] www.crowdflower.com

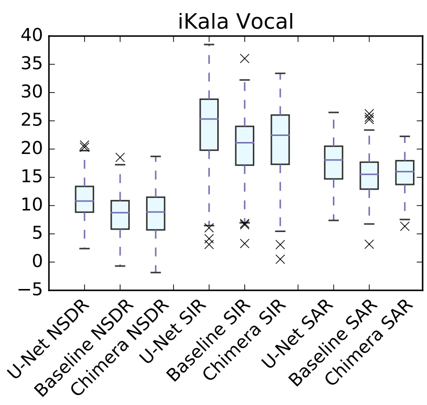


Figure 2. iKala vocal and instrumental scores

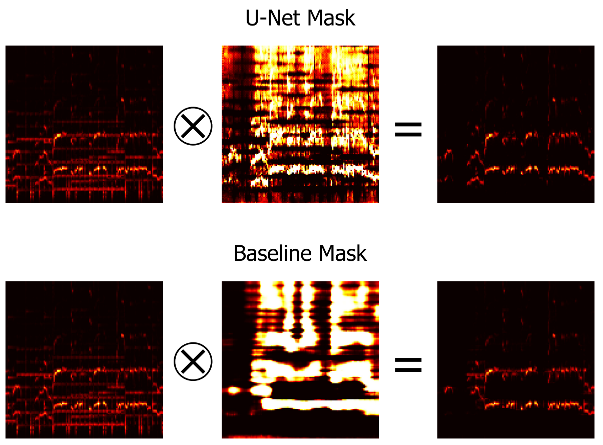
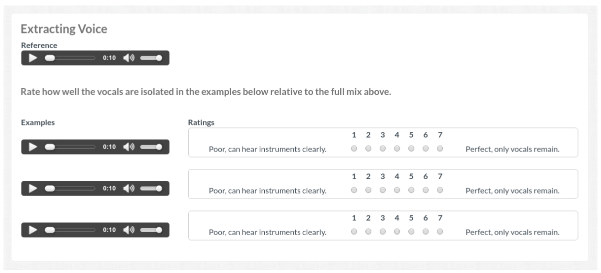


Figure 3. U-Net and baseline masks



Extracting Voice

Reference

Rate how well the vocals are isolated in the examples below relative to the full mix above.

Examples

Ratings

1 2 3 4 5 6 7
Poor, can hear instruments clearly. Perfect, only vocals remain.

1 2 3 4 5 6 7
Poor, can hear instruments clearly. Perfect, only vocals remain.

1 2 3 4 5 6 7
Poor, can hear instruments clearly. Perfect, only vocals remain.

Figure 4. CrowdFlower example question

To ensure the quality of the collected responses, we interspersed the survey with "control questions" that the user had to answer correctly according to a predefined set of acceptable answers on the Likert scale. Users of the platform are unaware of which questions are control questions. If they are answered incorrectly, the user is disqualified from the task. A music expert external to our research group was asked to provide acceptable answers to a number of random clips that were designated as control questions.

For the survey we used 25 clips from the iKala dataset and 42 clips from MedleyDB. We had 44 respondents and 724 total responses for the instrumental test, and 55 respondents supplied 779 responses for the voice test [7].

Figure 5 shows mean and standard deviation for answers provided on CrowdFlower. The U-Net algorithm outperforms the other two models on all questions.

5. CONCLUSION AND FUTURE WORK

We have explored the U-Net architecture in the context of singing voice separation, and found that it brings clear improvements over the state-of-the-art. The benefits of low-level skip connections were demonstrated by comparison to plain convolutional encoder-decoders.

A factor that we feel should be investigated further is the impact of large training data: work remains to be done to correlate the effects of the size of the training dataset to the quality of source separation.

We have observed some examples of poor separation on tracks where the vocals are mixed at lower-than-average volume, uncompressed, suffer from extreme application of audio effects, or otherwise unconventionally mixed. Since the training data consisted exclusively of commercially produced recordings, we hypothesize that our model has learned to distinguish the kind of voice typically found in commercial pop music. We plan to investigate this further by systematically analyzing the dependence of model performance on the mixing conditions.

Finally, subjective evaluation of source separation algorithms is an open research question. Several alternatives exist to 7-step Likert scale, e.g. the ITU-R scale [28]. Tools like CrowdFlower allow us to quickly roll out surveys, but care is required in the design of question statements.

6. REFERENCES

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] Some of the audio clips we used for evaluation can be found on `http://mirg.city.ac.uk/codeapps/vocal-source-separation-ismir2017`

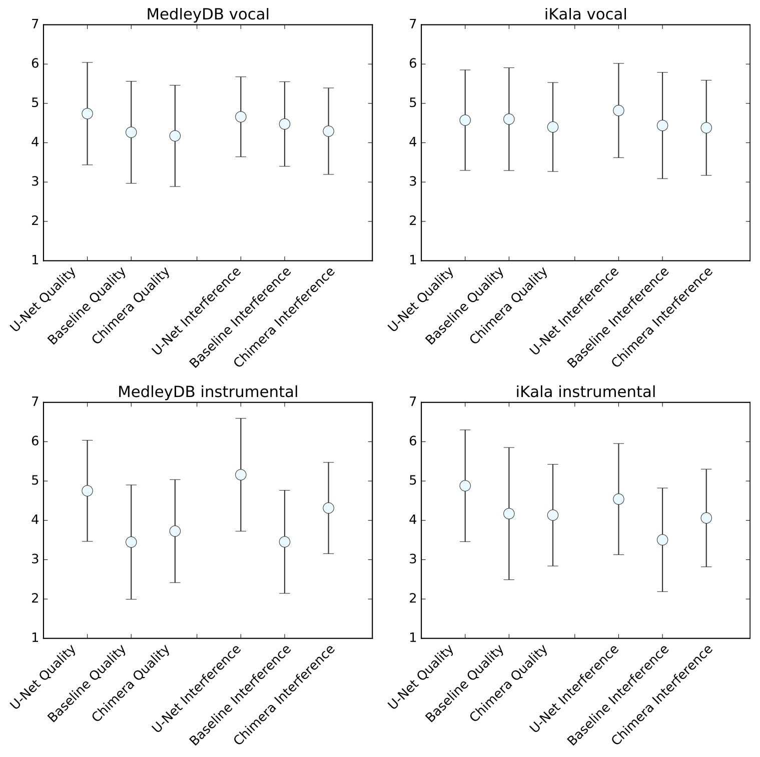


Figure 5. CrowdFlower evaluation results (mean/std)

[2] Aayush Bansal, Xinlei Chen, Bryan Russell, Abhinav Gupta, and Deva Ramanan. Pixelnet: Towards a general pixel-level architecture. *arXiv preprint arXiv:1609.06694*, 2016.

[3] Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A multitrack dataset for annotation-intensive MIR research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 155–160, 2014.

[4] Kevin Brown. *Karaoke Idols: Popular Music and the Performance of Identity*. Intellect Books, 2015.

[5] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation with the iKala dataset. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 718–722. IEEE, 2015.

[6] Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. Monoaural audio source separation using deep convolutional neural networks. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 258–266. Springer, 2017.

[7] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann. Subjective and objective quality assessment of audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2046–2057, 2011.

[8] Emad M Grais and Mark D Plumbley. Single channel audio source separation using convolutional denoising autoencoders. *arXiv preprint arXiv:1703.08019*, 2017.

[9] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, pages 477–482, 2014.

[10] Eric Humphrey, Nicola Montecchio, Rachel Bittner, Andreas Jansson, and Tristan Jehan. Mining labeled data from web-scale collections for vocal activity detection in music. In *Proceedings of the 18th ISMIR Conference*, 2017.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[15] Yi Luo, Zhuo Chen, and Daniel PW Ellis. Deep clustering for singing voice separation. 2016.

[16] Yi Luo, Zhuo Chen, John R Hershey, Jonathan Le Roux, and Nima Mesgarani. Deep clustering and conventional networks for music separation: Stronger together. *arXiv preprint arXiv:1611.06265*, 2016.

[17] Annamaria Mesaros and Tuomas Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010(1):546047, 2010.

[18] Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007, Vienna, Austria, September 23-27, 2007*, pages 375–378, 2007.

[19] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[20] Nicola Orio et al. Music retrieval: A tutorial and review. *Foundations and Trends® in Information Retrieval*, 1(1):1–90, 2006.

[21] Alexey Ozerov, Pierrick Philippe, Frdric Bimbot, and Rmi Gribonval. Adaptation of bayesian models for single-channel source separation and its application to voice/music separation in popular songs. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1564–1578, 2007.

[22] Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel 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, Taipei, Taiwan, October 27-31, 2014*, pages 367–372, 2014.

[23] Zafar Rafii and Bryan Pardo. Repeating pattern extraction technique (REPET): A simple method for music/voice separation. *IEEE transactions on audio, speech, and language processing*, 21(1):73–84, 2013.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[25] Andrew JR Simpson, Gerard Roma, and Mark D Plumbley. Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 429–436. Springer, 2015.

[26] Paris Smaragdis, Cedric Fevotte, Gautham J Mysore, Nasser Mohammadiha, and Matthew Hoffman. Static and dynamic source separation using nonnegative factorizations: A unified view. *IEEE Signal Processing Magazine*, 31(3):66–75, 2014.

[27] Philip Tagg. Analysing popular music: theory, method and practice. *Popular music*, 2:37–67, 1982.

[28] Thilo Thiede, William C Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John G Beerends, and Catherine Colomes. Peaq-the itu standard for objective measurement of perceived audio quality. *Journal of the Audio Engineering Society*, 48(1/2):3–29, 2000.

[29] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[30] Shankar Vembu and Stephan Baumann. Separation of vocals from polyphonic audio recordings. In *ISMIR 2005, 6th International Conference on Music Information Retrieval, London, UK, 11-15 September 2005, Proceedings*, pages 337–344, 2005.

[31] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing*, 14(4):1462–1469, 2006.

[32] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE transactions on audio, speech, and language processing*, 15(3):1066–1074, 2007.

[33] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

SKETCHING SONATA FORM STRUCTURE IN SELECTED CLASSICAL STRING QUARTETS

Louis Bigo[1] **Mathieu Giraud**[1] **Richard Groult**[2] **Nicolas Guiomard-Kagan**[2] **Florence Levé**[2,1]
[1] CRIStAL, UMR 9189, CNRS, Université de Lille, France
[2] MIS, Université de Picardie Jules Verne, Amiens, France
{louis,mathieu,richard,nicolas,florence}@algomus.fr

ABSTRACT

Many classical works from 18th and 19th centuries are *sonata forms*, exhibiting a piece-level tonal path through an exposition, a development and a recapitulation and involving two thematic zones as well as other elements. The computational music analysis of scores with such a large-scale structure is a challenge for the MIR community and should gather different analysis techniques. We propose first steps in that direction, combining analysis features on symbolic scores on patterns, harmony, and other elements into a structure estimated by a Viterbi algorithm on a Hidden Markov Model. We test this strategy on a set of first movements of Haydn and Mozart string quartets. The proposed computational analysis strategy finds some pertinent features and sketches the sonata form structure in some pieces that have a simple sonata form.

1. INTRODUCTION

1.1 Sonata Forms

Sonata form is a large-scale structure that can be found in many works from early Classical (18th century) to late Romantic period (end of 19th century). Sonata forms can be found in almost all first movements (and, often, in other movements) on Haydn, Mozart and Beethoven works.

Figure 1 shows an example of a very reduced sonata form in a piano sonatina by Kuhlau. Basically, a sonata form is built on a *piece-level tonal path* involving a *primary thematic zone (P)* and a contrasting *secondary thematic zone (S)*. It contains the following parts [15]:

- an *exposition*, often repeated, containing the thematic zone P in the main tonality (denoted by I), and the thematic zone S in an auxiliary tonality (usually, but not always, the tonality of the dominant of I, denoted by V);
- a *development* (D) characterized by tonal instability, in which the existing themes are transformed and possibly new themes are introduced, finished by a *retransition* (R), that focus back to the main tonality;
- a *recapitulation* of the themes P and S, both in the tonality of the tonic, possibly including elements that were added throughout the development.

Several striking events are found between these sections, in particular *cadences*. The transition (TR) between the P and S zones often ends on a *Medial Caesura* (MC), that is often a Half Cadence (HC) with additional break features [14]. The S zone generally concludes with a Perfect Authentic Cadence (PAC), and is followed by concluding patterns (C) without thematic content.

There are many possible variations on this basic structure. Somehow, the "regular" sonata form does not exist, and is merely a reconstruction. Some forms even do not have two contrasting themes but rather a "continuous exposition", such as in several Haydn string quartets or in the first movement of Mozart's "The Hunt" K 458 [15].

More than a rigid framework between sections, what constitutes the essence of a sonata form is a *high-level balance* in the whole piece: the tonal tension (the auxiliary tonality) and the rhetorical tension (textures, themes) created by the exposition and the development are resolved during the recapitulation. The development of sonata form was consubtantial to the emergence of instrumental music, this high-level balance enabling the design of musical works at a larger scale than before.

1.2 Sonata Forms, Musicology and Pedagogy

The term "Sonata form" was first coined in mid-1820s, in the A. B. Marx's *Berliner allgemeine musikalische Zeitung*, and later formalized in [22] and [6], even if some underlying principles were already known before [19, 29]. Somes authors conducted in-depth analyses of corpora with sonata forms, such as [12, 27] for Beethoven's String quartets or [32] for Beethoven's Piano sonatas.

In the last decades, authors proposed systematic theories on those forms [3, 4, 11, 21, 23, 28, 30]. The work of Hepokoski and colleagues [13, 14], culminating in the book [15], will be used here as a reference. These works formalized the notion of *rotations* organizing musical techniques throughout the piece.

Music research on sonata forms is thus still active today, two centuries after the climax of compositions in sonata

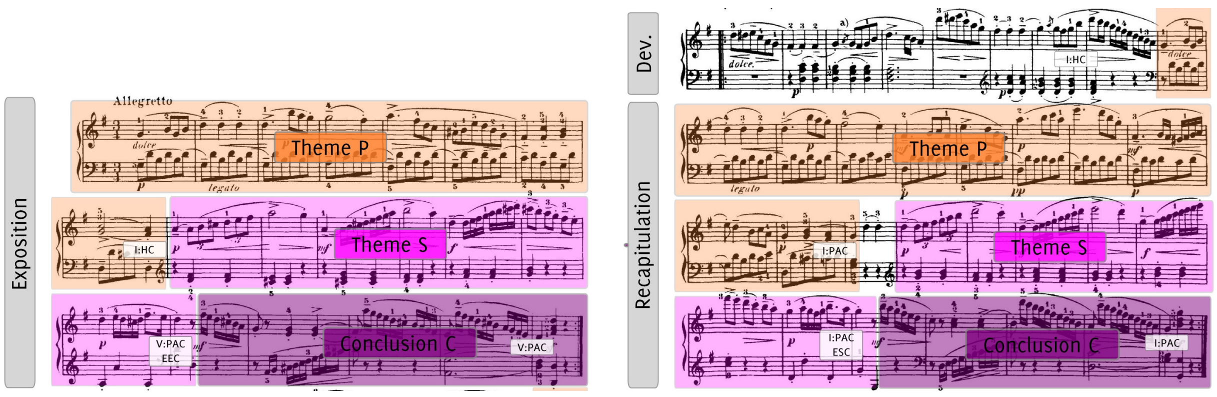


Figure 1. Allegretto of the piano sonatina Op. 55, no. 2 by Kuhlau. HC/PAC/EEC/ESC describe cadences and structure endings using the notations from [15]. This movement has very short sections, a tiny development (Dev) that is almost only a retransition, and almost no transition between themes P and S. It nevertheless features the characteristic *tonal path*: S (and C) is in the dominant tonality (V, D major) during the exposition and comes back to the main tonality (I, G major) during the recapitulation. Theme S and conclusion C are exactly transposed between the exposition and the recapitulation. The theme P is both times in the main tonality, but lasts 8 measures in the exposition and 11 in the recapitulation.

form. Such studies help to understand some principles of compositions and to have a new look on the history of music. Finally, sonata forms are one of the focus of lectures in music history, analysis or in composition.

1.3 Sonata Forms and MIR

Several works in the MIR community target in sonata forms, for example to test pattern extraction [25], tonality estimation [34], classification on n-grams, interval and metrical analyses [18]. However, there are very few works focusing on the sonata form *structure*. There will never be a "definitive analysis" of some piece in sonata form – even between musicians, one may choose to focus on some aspects. Anyway, some analytical viewpoints on the sonata forms make consensus and can be the focus of MIR research. On audio signals, Jiang and Müller computed correlations to detect exposition/recapitulation on the first movements of 28 Beethoven piano sonatas with self-similarity matrices [17]. They also trace transpositions and harmonic changes during the different parts. Weiß and Müller propose a model of the "tonal complexity" and map it on sections of sonata forms [35].

On symbolic data, we proposed in [7] a first approach to detect the exposition/recapitulation based on pattern matching. Baratè and al. proposed a model of the sonata form strucure trough Petri Nets, but without any algorithm [2].

We argue that sonata forms are very stimulating examples for MIR research, going from simple cases (repeated pattern with a tonal path in a sonatina, as in Figure 1) to very elaborated constructions (such as Beethoven piano sonatas) with many deviations from the norm [36]. The key point of an analysis of sonata forms – a large-scale tonal path – combines local-level features (themes, harmony) with a piece-level analysis.

An example of the complexity is the detection of Medial Caesura (MC) that marks the break between the two thematic zones. The MC is often marked by a half-cadence, but also by a long preparation, a "triple hammer blow" and then a silence on all voices [14]. However these events are not always found – and such events can also appear outside of a MC. To our knowledge, not any study in MIR tried to detect MC in sonata forms.

More generally, sketching an analysis of large-scale structures such as sonata forms is challenging for any analyst. A student in music analysis or a music theorist considers different elements and, through diligent *analytical choices*, summarize them into a coherent analysis. Our computational strategy to analyze sonata forms in symbolic scores takes inspiration from this approach. We propose to detect several analysis "features" using or extending MIR techniques (Section 2) and then to combine them to sketch the large-scale structure (Section 3). We test this strategy on a corpus of ten Haydn and Mozart string quartets and discuss the results (Sections 4 and 5).

2. ANALYSIS FEATURES

The following paragraphs list analysis features on which we will build to sketch the structure of the sonata form. Figures 2b and Figure 3 show these features on a first movement of a string quartet by Mozart. Such musical features are common in textbook or lecture descriptions of the sonata form. Their selection was done according to *whether their presence or absence could be characteristic of one (or several) section(s) in a sonata form.*

The following paragraphs lists these features, noted with boxes such as $\boxed{P}$. The detection of some of these features is taken from previous works [8]. Note that these features are already relatively mid-level or high-level MIR features, and their detection is often a challenge by itself that will not be detailed and evaluated here. Although not perfect, these methods detect features that can be combined as observable symbols produced by a Hidden Markov Model (Section 3).

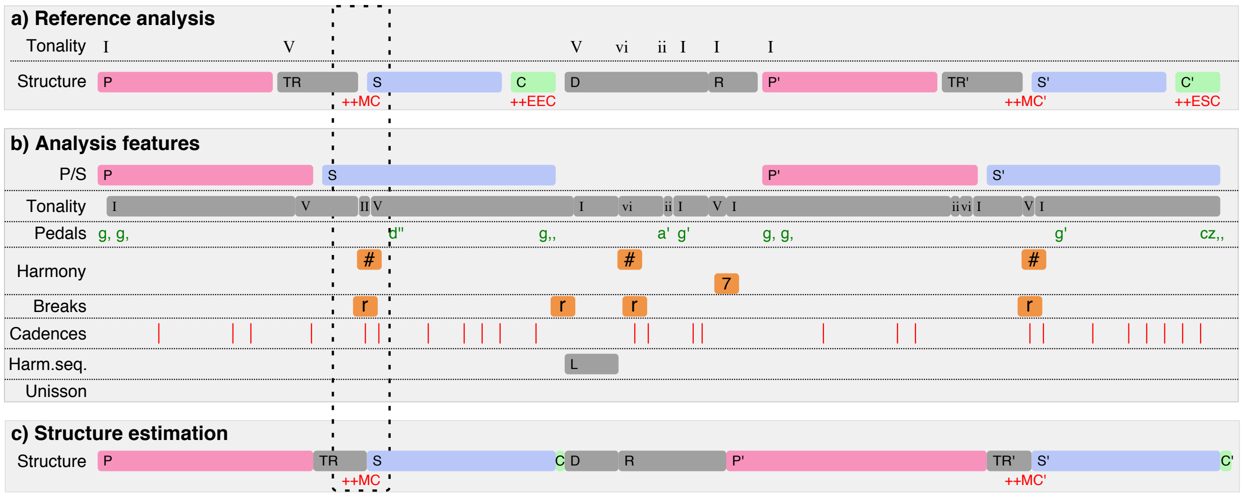


Figure 2. First movement of the String Quartet no. 4 in C major by W. A. Mozart (K157). (a.) Reference analysis indicating the main sections, following notations of [15], with the cadential structure endings, in particular the MC (Medial Caesura) between the P/TR and S zones. (b.) Analysis features as described in Section 2. (c.) Structure estimation by the HMM described in Section 3.2. The section with a dotted frame, around the first MC, is detailed on Figure 3.

2.1 Thematic Features

In a "regular" sonata form, the two thematic zones are strong markers of the form. We detect here repeated themes by computing a similarity score [8], which is based on equations similar to the Mongeau-Sankoff algorithm that uses dynamic programming [24]. The score function favors diatonic similarities, and only allows here alignment of two notes having the same duration.

- **P theme.** $\boxed{\text{P}}$ The P theme is searched using the score function, forbidding any transposition, and by comparing the start of the piece with other parts. The pattern is extracted only from the highest voice (first violin), but successive occurrences may be found in other voices. The first searched pattern begins at the start of the piece and ends at most at $1/3$ of the length of the piece. If no repeated pattern is found, the search is done again, starting from 2 measures after. The P theme must start in the first 10 measures and its length has to be more than 1 measure.

- **S theme.** $\boxed{\text{S}}$ The S theme is searched after the first P theme, again for at least one more occurrence. The S theme must start before $1/3$ of the length of the piece, end before $1/2$ of the length of the piece, and its length has to be between at least 4 measures. This time, the cost forces to find some pattern with a dominant transposition between the first occurrence and the following one. Once again, if no repeated pattern is found, the search is done by starting from a further position.

These features were introduced in [7] and may be related to the approach taken by [17] on audio signals. The selected ratios ($1/3$, $1/2$) reflect a generic balance of the structure of the sonata form. The score function could be improved by further research, in particular to allow more variations between the statements of the themes.

2.2 Harmonic Features

As tonal path is the most striking element of a sonata form, some features specifically focus on the harmony. Indeed, even without detection of full P/S themes, the harmony alone should give hints on analyzing sonata forms.

- **Tonality.** $\boxed{\text{I}}$ $\boxed{\text{A}}$ $\boxed{\text{O}}$ We detect local tonalities on 2-measures windows with a Krumhansl-Schmukler algorithm [20] used with the pitch profiles improved by Temperley [31]. Tonalities are then output relatively against the main (most present) tonality of the piece: main $\boxed{\text{I}}$, auxiliary $\boxed{\text{A}}$ and other $\boxed{\text{O}}$ tonalities. As our goal is not to infer precisely the tonality but to give a hint of the tonal context that will be used next in a probabilistic model, we do not use any algorithm improving this detection such as the full algorithm of [31].

- **Authentic cadences.** $\boxed{\text{AC}}$ Cadences are markers between sections. Moreover, cadences appear more likely in conclusive sections (C). We detect candidates of simple Perfect Authentic Cadences (PAC) and rooted Imperfect Authentic Cadences (rIAC) by checking harmonies over any V-I bass movement on strong beats using the algorithm of [8]. To take foreign notes into account, the V chord, characterized by the leading tone and possibly the seventh, has to be found somewhere while the bass holds the dominant. As this detection is here solely based on the harmony, it may induce some false positives. This is the case on Figure 3, where two successive V-I bass movements are interpreted as PACs even if they do not correspond to any phrase ending.

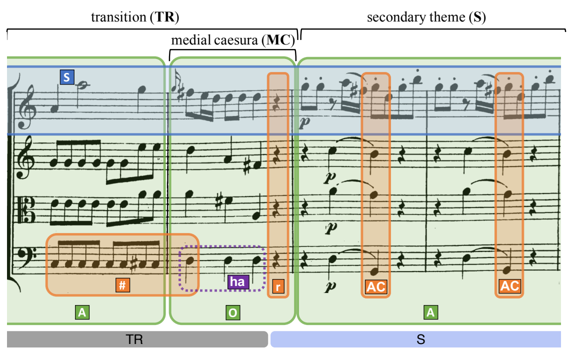


Figure 3. First medial caesura (MC) in the first movement of the String Quartet no. 4 in C major by W. A. Mozart (K157), measures 29 to 32. See Figure 2 for an overview of this movement. The MC ends the transition (TR) and is before the beginning of the secondary theme (S). The computed analysis retrieves several features within this region: the thematic pattern S $\boxed{\text{S}}$ – falsely detected even before the MC, see discussion in the text – tonality regions corresponding to auxiliary $\boxed{\text{A}}$ and – falsely detected – other $\boxed{\text{O}}$ tonalities, a chromatic upward bass movement $\boxed{\#}$ and a full rest $\boxed{\text{r}}$. Two spurious cadences $\boxed{\text{AC}}$ are also detected at the beginning of the secondary theme. Although not taken into account in the present work, the extract includes a *triple hammer blow* $\boxed{\text{ha}}$ characteristic of the medial caesura.

- **Preparation of half-cadences.** We detect both *chromatic upward bass movements* $\boxed{\#}$ (one chromatic semitone followed by one diatonic semitone, contiguous notes with the same pitch being taken as one note, see Figure 3) and putative *diminished seven chords* $\boxed{7}$ (any diminished seventh or augmented second interval between two notes sounding at the same time). These feature are often found in the preparation of half-cadences, and especially for the preparation of the Medial Caesura.

- **Pedals.** $\boxed{\text{ped}}$ We detect pedals during more than 1 measure in one voice. Pedals are often found during development and conclusion sections. On the contrary, they are often not found in thematic P/S zones, except at the very beginning of the piece.

2.3 Other Features

These features combine melody and harmony and/or other music elements.

- **Full rests.** $\boxed{\text{r}}$ We look for rests that occur in all voices simultaneously. Such rests are often found at key places: after the MC, after the exposition, and just before the recapitulation.

- **Unisons.** $\boxed{\text{uni}}$ We detect unisons between all the voices using the algorithm presented in [10]. Unisons are strong markers that often also break the musical flow: They are also likely to be found in structural breaks.

- **Long harmonic sequences.** $\boxed{\text{L}}$ We detect harmonic sequences by at least three successive occurrences of melodic patterns in all four voices, using the algorithm presented in [9] and reporting sequences during at least 5 measures. Such long harmonic sequences, often modulating, can be found in the development.

3. STRUCTURE ESTIMATION THROUGH FEATURE COMBINING

Analysis features are sampled at regular intervals to give sequences of symbols (Figure 2b). We propose to gather these features into a sonata form structure (Figure 2c). The following paragraphs present the Hidden Markov Models (HMM) framework we use and then the particular HMM designed for sonata forms.

3.1 Hidden Markov Models with Multiple Outputs

Markov model. We consider a finite alphabet of symbols $\mathcal{A} = \{\alpha_1, \alpha_2 ...\}$ that will be here the analysis features. The Markov model $\mathcal{M} = (Q, \pi, \tau, T, E)$ on $\mathcal{A}$ is defined by a set of n states $Q = \{q_1, ... q_n\}$ corresponding here to sections of the sonata form, the initial state probabilities $\pi = (\pi_1, ... \pi_n)$, and the final state probabilities $\tau = (\tau_1, ... \tau_n)$. $T(i \rightarrow j)$ is the *transition probability* – state q_i goes to state q_j – and $E(i \rightsquigarrow \alpha)$ is the *emission probability* – state q_i emits feature α. All probabilities are between 0 and 1, and the probabilities arrays sum to 1.

Given an integer t, we call a t-tuple $P = (p_1, ... p_t) \in [1, n]^t$ a *path* in $\mathcal{M}$. This path goes through the t states $q_{p_1} ... q_{p_t}$. We also consider a sequence of symbols $w = \alpha_1 ... \alpha_{t-1} \in \mathcal{A}^{t-1}$. The probability that the model $\mathcal{M}$ follows a path P while outputting the sequence w, one state outputting one character at each step, is given by:

$$p(P, w) \ = \ \pi_{p_1} \cdot \Pi_{i=1}^{t-1} (E(p_i \rightsquigarrow \alpha_i) \cdot T(p_i \rightarrow p_{i+1})) \ \cdot \ \tau_{p_t}$$

Outputting mutiple symbols. Several features can be predicted at the same step. We thus now consider that a state may output simultaneously a set of symbols $A = \{\alpha_1 ... \alpha_a\} \subset \mathcal{A}$. If these emissions are independent events, the probability that the state q_i outputs the set A is

$$E(i \rightsquigarrow A) \ = \ \Pi_{\alpha \in A} E(i \rightsquigarrow \alpha) \ \cdot \ \Pi_{\alpha \in \mathcal{A} \setminus A} (1 - E(i \rightsquigarrow \alpha))$$

We now consider a path P as before and a sequence of sets of symbols $W = A_1 ... A_{t-1}$. The probability $p(P, W)$ that the model $\mathcal{M}$ follows a path P while outputting the sequence W is given by the same equation, replacing $E(p_i \rightsquigarrow \alpha_i)$ by $E(p_i \rightsquigarrow A_i)$.

HMM. Now we consider $\mathcal{M}$ as a Hidden Markov Model (HMM). The path P is unknown but we *observe* a sequence of sets of symbols W.

Finding the most probable path P that maximizes $p(P, W)$ is done by the classical Viterbi algorithm [26, 33] that first uses a forward stage to compute the probability of being in a state while outputting $A_1 ... A_j$, and then that finds back the optimal path in a backward pass.

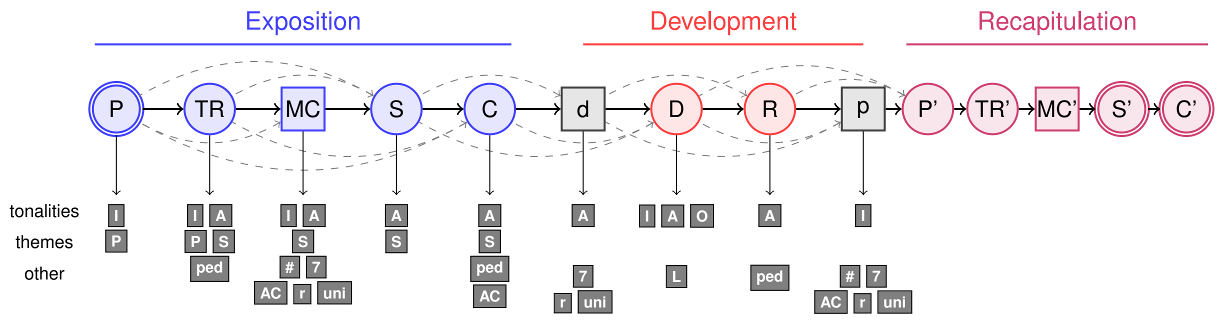


Figure 4. Hidden Markov Model sketching a regular sonata form structure from analysis features. The initial state is P, the final states are S' and C'. The square states (MC/MC' "medial caesura", d "transition to development", p "retransition to primary theme") are transient states intended to last one or a few quarters, and are characterized by break features ($\boxed{\#}$, $\boxed{7}$, $\boxed{\text{AC}}$, $\boxed{\text{r}}$, $\boxed{\text{uni}}$). Each state has a (not shown) loop transistion over itself. The horizontal straight transitions have the second highest probabilities, and the curved dashed transitions enable to skip some states with a low probability. Only the main emissions are shown here: the states may also emit other symbols with a low probability. For clarity, auxiliary transitions and emissions are not shown in the recapitulation. They are the same than in the exposition, except that the tonality emissions focus on the main tonality $\boxed{\text{I}}$.

3.2 A HMM to Sketch Sonata Form Structure

Figure 4 depicts the HMM created to sketch the sonata form structure. The HMM uses the alphabet $\{\boxed{\text{P}}, \boxed{\text{S}}, \boxed{\text{I}}, \boxed{\text{A}}, \boxed{\text{O}}, \boxed{\#}, \boxed{7}, \boxed{\text{ped}}, \boxed{\text{r}}, \boxed{\text{uni}}, \boxed{\text{L}}, \boxed{\text{AC}}\}$ containing the analysis features presented in the previous section. The features are sampled at every quarter note. The 14 states $Q = \{\text{P}, \text{TR}, \text{MC}, \text{S}, \text{C}, \text{d}, \text{D}, \text{R}, \text{p}, \text{P}', \text{TR}', \text{MC}', \text{S}', \text{C}'\}$ were selected to match the various sections of the "regular" sonata form as well as some transitions $\{\text{MC}, \text{d}, \text{p}, \text{MC}'\}$ between these sections.

As discussed in the introduction, even if such a "regular" sonata form is a fiction, some pieces do follow this structure: The proposed states intend to match these pieces. As the model is very simple – and as the detection of the features is far from perfect – the goal is not to perfectly match these 14 stages to actual sections of sonata forms, but rather to sketch the structure. We defined so many states to try to follow actual structures – for example, TR and MC states definitely imply different music events: in TR, focalization on the auxiliary tonality $\boxed{\text{A}}$, possibly with some $\boxed{\text{ped}}$, and, in MC, conclusion with a cadence, possibly $\boxed{\text{AC}}$, possibly with additional events: $\boxed{\#}$, $\boxed{7}$, $\boxed{\text{r}}$.

Transitions and emission probabilities of the selected symbols were choosen manually by a trial-and-error process (see discussion in Section 5). Each state has a loop transition over itself with a very high probability (0.8). The emission probabilities were drafted according to what was described in the previous section. Some adjustments were made to take into account limits of some feature detection. For example, the feature $\boxed{\text{S}}$ is often detected outside of S, as on Figure 3. Indeed, sometimes some few measures before the MC in the recapitulation are exactly a transposition of the same passage in the exposition. Thus, in the HMM, $\boxed{\text{S}}$ can also be emitted by the states TR and MC. The model with all probabilities can be downloaded at `algomus.fr/sonata`.

4. CORPUS AND RESULTS

4.1 Corpus and Reference Analysis

Experiments were done in python3, within the music21 framework [5] extended with analytic labels [1]. Pieces were given as `.krn` Humdrum files [16] downloaded from `kern.humdrum.org`. The corpus $\mathcal{C}_{10}$ contains 10 first movements of classical string quartets composed by Haydn and Mozart (see Table 1). All these are in major mode. The selected Mozart quartets are mostly early works (K80 and the three Milanese quartets K155, K156, K157) that have a simple sonata form, even if the first movement of K80 is adagio. Although they have the typical tonal path, the two Haydn quartets 54-3 and 64-4 do not exhibit a clear S (or S') theme. We denote by $\mathcal{C}_8$ the set of the 8 remaining pieces. These pieces were analyzed following principles of [15] to determine P/TR/S/C sections as well as MCs. At least two curators checked every reference analysis. These analyses are available under an open-source license from `algomus.fr/datasets`.

4.2 Results of the Proposed Strategy

Table 1 shows the structure estimation of the HMM on all pieces in $\mathcal{C}_{10}$. As written above, the 14-state HMM was not intended to fit perfectly with the structure, but rather to give hints on the sonata form structure. Moreover, some sections are difficult to predict, or even to define: for example, the start of the transition (TR) is often "blurred" in the end of P. Note also that the features and model we proposed do not separate well the S themes from the conclusions C. We thus propose here to focus the evaluation on four key events of the sonata form (start of S, D, P', S'):

- **MC+S.** In the exposition, the MC followed by the start of S is perfectly or approximately found in 4 pieces in $\mathcal{C}_8$. In non-regular structures ($\star$), a S may be falsely detected, because the feature $\boxed{\text{S}}$ may report transposed sections of the theme P.

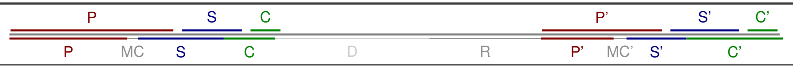
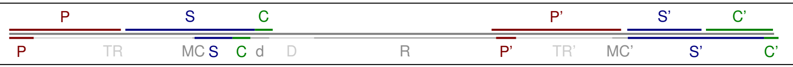
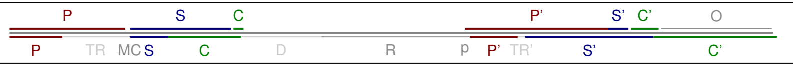
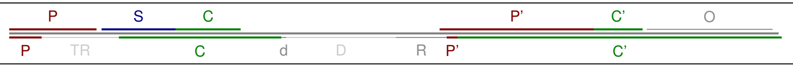
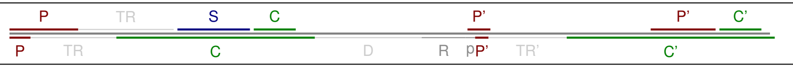
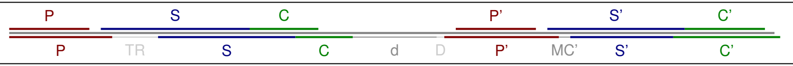
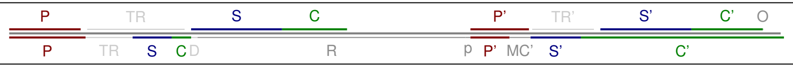
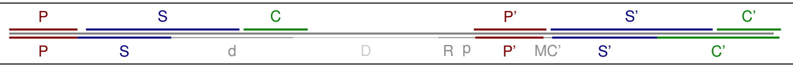
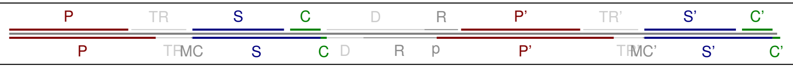
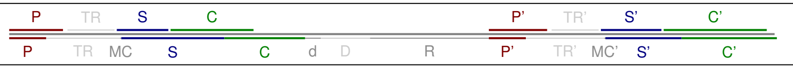
	Reference (top) and computed (bottom) analyses	MC+S	D	P'	MC'+S'
Haydn op. 33 no. 2		−	+	+	−
Haydn op. 33 no. 3		−	=	+	+
Haydn op. 33 no. 5		+	+	+	−
Haydn ⋆ op. 54 no. 3		·	−	=	·
Haydn ⋆ op. 64 no. 4		·	=	?	·
Mozart K80 no. 1		−	=	+	=
Mozart K155 no. 2		−	−	+	−
Mozart K156 no. 3		=	−	+	+
Mozart K157 no. 4		+	+	−	+
Mozart K387 no. 14		+	−	+	+

Table 1. Structure detection on ten first movements of Haydn and Mozart string quartets. The top lines are the reference analyses and the bottom line the structure found by the HMM. The four columns MC+S, D, P' and MC'+S' evaluate the prediction of the *start* of these events or sections: $+$ (perfect or almost, that is at most 1 measure shifted from the reference), $=$ (approximate match, between 2 and 3 measures), $-$ (not found, or too far from the reference, at least 4 measures). We do not evaluate S positions $(\cdot)$ for pieces marked with $\star$, as they do not follow a "regular" bithematic sonata form structure with a clear secondary theme.

- **D**. The start of the development is perfectly or approximately found in 6 pieces in $\mathcal{C}_{10}$. This detection is usually grounded by the feature r.

- **P'**. The start of the recapitulation is perfectly or approximately found in 8 pieces in $\mathcal{C}_{10}$, mainly driven by the feature P. Haydn op. 64 no. 4 has partial repeats of the P theme during the recapitulation, and Mozart K157 has a long retransition that is falsely detected as a P theme due to the feature I.

- **MC'+S'.** In the recapitulation, the start of S' is approximately found in 5 pieces in $\mathcal{C}_8$. It is again often grounded on the break features.

Sonata structure sketch. Back on the motivation of this study, the predicted sonata form structure seems quite good for Mozart K157 and K387: starts and durations of sections are quite precisely detected. For Mozart K80, K156 and Haydn 33-3 and 33-5, the structure is coarsely detected, but bad lengths or shifts in some predicted sections are not satisfying. Note that, on K80, even if the thematic features are not detected (data not shown), the path estimated by the HMM is still sensible, mainly due to tonalities as well as break events.

The bad results on the other pieces mostly come from a wrong detection of the start of S/S'. This suggest that features helping the prediction of the MC as well as the HMM should be improved.

5. DISCUSSION

The music analysis of large-scale structures, such as the sonata forms, requires to gather different analytical elements into some coherent analysis. Taking inspiration from what the analysts do, we proposed a strategy to sketch such sonata structures, designing a HMM modeling music knowledge over analysis features. The proposed strategy manages to sketch the structure of some "regular" sonata forms in string quartets, finding the most important sections (P/S, D, P'/S') and sometimes detecting the location of the Medial Caesura (MC).

This strategy should now be evaluated on a larger corpus. More general perspectives include both the improvement of individual feature detection – conceiving or using MIR techniques that may be used to analyze any tonal music, in classical music but also in jazz or pop repertoires – and also improvement of the HMM. Other HMM topologies could analyze more elaborated variations of sonata forms – especially continuous exposition. Analyzing late Mozart quartets or some romantic quartets will also be very challenging.

In the present work, we manually designed transition and emission probabilities. These probabilities could also be learned on larger corpora, but the number of parameters to learn makes such a learning difficult. A solution to benefit both from human expertise and machine learning could be also to learn the weights of only manually selected emissions and transitions.

Acknowledgements. We thank Elise Janvresse, Marc Rigaudière, and the anonymous reviewers for their insightful comments. This work was supported by a grant from the French Research Agency (ANR-11-EQPX-0023 IR-DIVE).

6. REFERENCES

[1] Guillaume Bagan, Mathieu Giraud, Richard Groult, and Emmanuel Leguy. Modélisation et visualisation de schémas d'analyse musicale avec music21. In *Journées d'Informatique Musicale (JIM 2015)*, 2015.

[2] Adriano Baratè, Goffredo Haus, and Luca A Ludovico. Music analysis and modeling through Petri nets. In *International Symposium on Computer Music Modeling and Retrieval (CMMR 2005)*, pages 201–218, 2005.

[3] William E. Caplin. *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven*. Oxford University Press, 2000.

[4] William E. Caplin. The classical sonata exposition: Cadential goals and form-functional plans. *Tijdschrift voor Muziektheorie*, 6(3):195–209, 2001.

[5] Michael Scott Cuthbert and Christopher Ariza. music21: A toolkit for computer-aided musicology and symbolic music data. In *International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 637–642, 2010.

[6] Carl Czerny. *School of Practical Composition*. R. Cocks, London, 1848.

[7] Laurent David, Mathieu Giraud, Richard Groult, Corentin Louboutin, and Florence Levé. Vers une analyse automatique des formes sonates. In *Journées d'Informatique Musicale (JIM 2014)*, 2014.

[8] Mathieu Giraud, Richard Groult, Emmanuel Leguy, and Florence Levé. Computational fugue analysis. *Computer Music Journal*, 39(2), 2015.

[9] Mathieu Giraud, Richard Groult, and Florence Levé. Detecting episodes with harmonic sequences for fugue analysis. In *International Society for Music Information Retrieval Conference (ISMIR 2012)*, 2012.

[10] Mathieu Giraud, Florence Levé, Florent Mercier, Marc Rigaudière, and Donatien Thorez. Modeling texture in symbolic data. In *International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014.

[11] Robert O. Gjerdingen. *Music in the Galant Style*. Oxford University Press, 2007.

[12] Theodor Helm. *Beethovens Streichquartette: Versuch einer technischen Analyse dieser Werke im Zusammenhange mit ihren geistigen Gehalt*. Leipzig, 1885.

[13] James Hepokoski. Beyond the sonata principle. *J. of the American Musicological Society*, 55(2):91, 2002.

[14] James Hepokoski and Warren Darcy. The medial caesura and its role in the eighteenth-century sonata exposition. *Music Theory Spectrum*, 19(2):115–154, 1997.

[15] James Hepokoski and Warren Darcy. *Elements of Sonata Theory: Norms, Types, and Deformations in the Late-Eighteenth-Century Sonata*. Oxford University Press, 2006.

[16] David Huron. Music information processing using the Humdrum toolkit: Concepts, examples, and lessons. *Computer Music Journal*, 26(2):11–26, 2002.

[17] Nanzhu Jiang and Meinard Müller. Automated methods for analyzing music recordings in sonata form. In *International Society for Music Information Retrieval Conference (ISMIR 2013)*, pages 595–600, 2013.

[18] Stephanie Klauk and Frank Zalkow. Das italienische Streichquartett im 18. Jahrhundert. Möglichkeiten der semiautomatisierten Stilanalyse. urn:nbn:de:101:1-201609071562, 2016.

[19] Heinrich Christoph Koch. *Versuch einer Einleitung zur Composition (volume 3)*. Leipzig, 1793.

[20] Carol L. Krumhansl and Edward J. Kessler. Tracing the dynamic changes in perceived tonal organisation in a spatial representation of musical keys. *Psychological Review*, 89(2):334–368, 1982.

[21] Steve Larson. Recapitulation recomposition in the sonata-form first movements of Haydn's string quartets: Style change and compositional technique. *Music Analysis*, 22(1-2):139–177, 2003.

[22] Adolph Bernhard Marx. *Die Lehre von der musikalischen Komposition (volumes 2 et 3)*. Breitkopf & Härtel, Leipzig, 1838, 1845.

[23] Jan Miyake. *The Role of Multiple New-key Themes in Selected Sonata-form Exposition*. PhD thesis, Univ. of New York, 2004.

[24] Marcel Mongeau and David Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, 1990.

[25] Oriol Nieto and Morwaread Mary Farbood. Perceptual evaluation of automatically extracted musical motives. In *International Conference on Music Perception and Cognition (ICMPC 2012)*, pages 723–727, 2012.

[26] Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[27] Phillip Radcliffe. *Beethoven's String Quartets*. E. P. Dutton, 1965.

[28] Leonard Ratner. *Classical Music: Expression, Form, and Style*. Schirmer, 1980.

[29] Anton Reicha. *Traité de haute composition musicale (volume 2)*. Paris, 1826.

[30] Charles Rosen. *Sonata Forms*. W. W. Norton, 1980.

[31] David Temperley. What's key for key ? the Krumhansl-Schmuckler key-finding algorithm reconsidered. *Music Perception*, 17(1):65–100, 1999.

[32] Donald Francis Tovey. *A Companion to Beethoven's Pianoforte Sonatas: Complete Analyses*. AMS Press (reedited in 1998), 1931.

[33] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[34] Christof Weiß and Julian Habryka. Chroma-based scale matching for audio tonality analysis. In *Conference on Interdisciplinary Musicology (CIM 2014)*, 2014.

[35] Christof Weiß and Meinard Müller. Quantifying and visualizing tonal complexity. In *Conference on Interdisciplinary Musicology (CIM 2014)*, 2014.

[36] Frans Wiering and Emmanouil Benetos. Digital musicology and MIR: Papers, projects and challenges. In *International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013. Late-breaking session.

Author Index