MUSICAL AUDIO SIMILARITY WITH SELF-SUPERVISED CONVOLUTIONAL NEURAL NETWORKS

Carl Thomé[1] **Sebastian Piwell**[1] **Oscar Utterbäck**[1]

[1] Epidemic Sound, Stockholm, Sweden

firstname.lastname@epidemicsound.com

ABSTRACT

We have built a music similarity search engine that lets video producers search by listenable music excerpts, as a complement to traditional full-text search. Our system suggests similar sounding track segments in a large music catalog by training a self-supervised convolutional neural network with triplet loss terms and musical transformations. Semi-structured user interviews demonstrate that we can successfully impress professional video producers with the quality of the search experience, and perceived similarities to query tracks averaged 7.8/10 in user testing. We believe this search tool will make for a more natural search experience that is easier to find music to soundtrack videos with.

1. INTRODUCTION

Finding songs to soundtrack with is crucial for video production. Rather than browsing by track metadata (genres, moods, musical key, instrumentation), we believe browsing by example is an intuitive alternative as the user does not need to provide textual descriptions [1]. Instead, by providing example songs and how they *sound*, we want to let video producers find similar tracks in a music catalog.

By having an algorithmic notion of what constitutes music similarity for video production, we could simply measure pairwise distances from a query audio and recommend neighboring catalog tracks. However, it is difficult to design such algorithms for a general setting [2, 3]: the search can become overly strict, retrieving poor matches that break the rules, or entirely miss good matches not covered by the rules. Thus, many systems rely on learning similarities by an objective function and example data [2, 4, 5] to relax assumptions before the retrieval task.

In this work, we've trained similarity learning models by triplet loss terms, with the ambition of finding a distance measure where similar sounding clips should be close and dissimilar sounding clips should be distant [6, 7]. In order to design our notion of similarity, we use music production knowledge to design rules for what constitutes a positive and negative example in the triplet loss formulation.

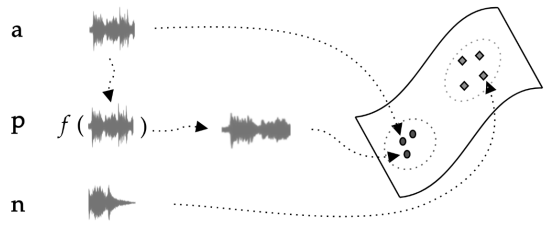

Figure 1. An overview of our self-supervised similarity learning approach for musical audio. We construct training data by transforming audio clips with a randomized audio effects chain of musical transformations.

Our contribution is empirical evidence that self-supervised similarity learning can be valuable for music similarity search in a large scale production environment in an industry setting.

2. DESIGN

In order to retrieve similar sounding tracks from a given query song, we need to find parameters θ for an encoding function e_θ such that pairwise distances between query songs and tracks in the catalog will agree with video producers, meaning

$$d(e_\theta(\mathbf{a}), e_\theta(\mathbf{p})) < d(e_\theta(\mathbf{a}), e_\theta(\mathbf{n}))$$

where d is Euclidean distance, and the anchor $\mathbf{a}$ is any query song. $\mathbf{n}$ is a negative example song not deemed similar to the anchor, and $\mathbf{p}$ is a positive example song perceived as similar to the anchor. However, we lack annotated triplets $(\mathbf{a}, \mathbf{n}, \mathbf{p})$ from video producers, so we've designed an audio effects chain of musical transformations f that we believe e_θ should be approximately invariant for (see Figure 1), as in

$$e_\theta(\mathbf{x}_0) \approx e_\theta(f(\mathbf{x}_0)).$$

During training the audio effects are applied stochastically by turning them on and off and tweaking their settings per audio clip, and the goal is to steer the learned similarity by including this domain knowledge. [1]

[1] This is a powerful benefit of our approach as future developers can tune the system for changing requirements without having to understand historical annotators, since the similarity notion is explicit in code.

2.1 Data

From our music catalog of 40,000 tracks, we randomly assign 80% of tracks to training, 10% for validation and 10% for testing by hashing track ids. We use validation tracks for model development (learning rate annealing, early stopping and model selection) and testing tracks exclusively for estimating generalization upon deployment. We downsample and sum each full mix track to monophonic 16 kHz with librosa [8], and sample ten random clips at 10.0 seconds each from every track with Dataflow [9], such that long tracks don't dominate in model training.

2.2 Encoder

The encoder structure consists of a spectral transform from audio signals to Mel spectrograms, with online feature standardization by batch normalization, and a standard ConvNet for embedding computation. The spectral transform has a DFT size of 2048 samples, windowed by 50% overlapping Hann windows. The Mel bands are triangular filters starting from 20 Hz up to the Nyquist limit as provided in TensorFlow [10]. Magnitudes are log scaled to promote timbral features and not only pitch and rhythm. To project into RGB space, an initial 2D convolution runs between the spectrogram and the ConvNet. We view the ConvNet variant itself as a categorical hyperparameter during model development but the default classification head is replaced with a fully-connected layer with 128 output units and a unit normalizing activation function such that embeddings are on the surface of a hypersphere in $\mathbb{R}^{128}$ with radius one.

2.3 Objective

The most important puzzle piece in our system is how the encoder receives parameters θ. We rely on an objective of competing triplet loss terms.

For every input clip, we transform it into a positive by applying the stochastic audio effects chain f, consisting of a random time shift, time stretch, pitch shift, reverb and additive noise from a truncated Gaussian. The time stretching and pitch shifting is implemented as a phase vocoder while the reverb is implemented as multiplication in the frequency domain with exponentially decaying white noise.

We perform online triplet mining [11] and deem audio clips within a minibatch as positives if they share track metadata. We weigh triplet loss terms for transformed clips, clips from the same track, and genre and mood membership, with weights 1.0, 0.5, 0.1, 0.1 respectively. Model parameter and gradient estimate updates are computed with Adam [12] for minibatches of 256 audio clips.

3. EVALUATION

To know that encoders produce meaningful points in the embedding space such that neighboring tracks are perceived as similar, we conducted qualitative tests. However, for choosing models to send out for testing, we relied on offline metrics in [7] and computed average precision per category (one per genre, one per mood, etc.) on a ranked list of pairwise distances between embeddings where we put a one if both embeddings belong to the same category, and a zero if they belong to different categories.

3.1 Quantitative

Intuitively we want a clear separation between embedded audio clips from vastly different musical genres and moods. Similarly, we likely want clips from the same track to cluster together. [2]

We have computed mean average precision (mAP) as in [7] in Table 1 for our encoder, a random encoder, and a baseline encoder. The random encoder draws an embedding from a uniform distribution per clip and uses that as its representation. The baseline encoder computes 20 MFCC coefficients from 128 band Mel spectrograms and decorrelates the MFCC dimensions with incremental PCA with librosa and scikit-learn [8, 13].

	Genre	Mood	Track
Random	0.3143	0.3320	0.0495
Baseline	0.4011	0.3684	0.6054
ConvNet	**0.7095**	**0.5087**	**0.8490**

Table 1. Mean average precision (mAP) scores on test tracks for different sets of annotations and encoders.

3.2 Qualitative

Acknowledging that music similarity is hard to evaluate without humans [5], we conducted semi-structured interviews with five professional video producers in which they explored a production grade web app for music discovery and were tasked with finding similar music from three starting tracks of their own choosing.

The mean opinion score when asked how similar they perceived the search results to be to their chosen query tracks was 7.8/10. The interviews were recorded and transcribed and can be shared upon request.

4. DISCUSSION

After seeing video producers interact with the music similarity search engine, we believe self-supervised similarity learning on musical audio makes for a natural search experience to find music to soundtrack videos with. By circumventing annotation needs and not requiring users to express what they want beyond providing listenable examples, we can help video producers find suitable tracks easier and faster.

In future work, we will explore stereophonic signals, multi-track recordings, alternative loss functions [14] and musical transformations [15], and ways of granting user control to the similarity measure [16].

[2] However, if we get perfect scores we arguably haven't accomplished anything meaningful, as we then could simply resort to using class memberships directly to recommend tracks.

5. REFERENCES

[1] P. Grosche, M. Müller, and J. Serra, "Audio content-based music retrieval," in *Dagstuhl Follow-Ups*, vol. 3. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2012.

[2] J. Schluter and C. Osendorfer, "Music similarity estimation with the mean-covariance restricted boltzmann machine," in *10th International Conference on Machine Learning and Applications and Workshops*, vol. 2. IEEE, 2011, pp. 118–123.

[3] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, "On rhythm and general music similarity," in *International Symposium on Music Information Retrieval (ISMIR)*, 2009, pp. 525–530.

[4] M. Slaney, K. Weinberger, and W. White, "Learning a metric for music similarity," in *International Symposium on Music Information Retrieval (ISMIR)*, vol. 148, 2008.

[5] S. Dieleman and B. Schrauwen, "Learning content-based metrics for music similarity," in *5th International Workshop on Machine Learning and Music (MML-2012)*, 2012, pp. 13–14.

[6] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2840–2848.

[7] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 126–130.

[8] B. McFee, V. Lostanlen, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, J. Mason, D. Ellis, E. Battenberg, S. Seyfarth, R. Yamamoto, K. Choi, viktorandreevichmorozov, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, and T. Kim, "librosa/librosa: 0.8.0," Jul. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3955228

[9] T. Akidau, R. Bradshaw, C. Chambers, S. Chernyak, R. J. Fernández-Moctezuma, R. Lax, S. McVeety, D. Mills, F. Perry, E. Schmidt *et al.*, "The dataflow model: a practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing," in *Proceedings of the VLDB Endowment*, vol. 8, 2015, pp. 1792–1803.

[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference for Learning Representations*, 2015.

[13] 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," pp. 2825–2830, 2011.

[14] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3875–3879.

[15] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *arXiv preprint arXiv:2103.09410*, 2021.

[16] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6–10.