ACCURATE AND SCALABLE VERSION IDENTIFICATION USING MUSICALLY-MOTIVATED EMBEDDINGS

Furkan Yesiler[†] *Joan Serrà*[*‡] *Emilia Gómez*[†§]

[†] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[‡] Dolby Laboratories, Barcelona, Spain
[§] Joint Research Centre, European Commission, Sevilla, Spain

ABSTRACT

The version identification (VI) task deals with the automatic detection of recordings that correspond to the same underlying musical piece. Despite many efforts, VI is still an open problem, with much room for improvement, specially with regard to combining accuracy and scalability. In this paper, we present MOVE, a musically-motivated method for accurate and scalable version identification. MOVE achieves state-of-the-art performance on two publicly-available benchmark sets by learning scalable embeddings in an Euclidean distance space, using a triplet loss and a hard triplet mining strategy. It improves over previous work by employing an alternative input representation, and introducing a novel technique for temporal content summarization, a standardized latent space, and a data augmentation strategy specifically designed for VI. In addition to the main results, we perform an ablation study to highlight the importance of our design choices, and study the relation between embedding dimensionality and model performance.

***Index Terms*—** Cover song identification, deep learning, music embedding, network encoder.

1. INTRODUCTION

Version identification (VI) commonly refers to the task of determining, by computational means, whether two audio renditions correspond to the same underlying musical composition [1]. Being more challenging than traditional audio fingerprinting [2], VI goes beyond near-exact duplicate detection to embrace additional perceptual differences that, despite having a contrasting imprint in the signal, convey the same musical entity [3]. Such is the case of changes in instrumentation, musical key, tempo, timing, structure, or lyrics, to name a few [1]. Besides digital rights management, VI has application to music organization, retrieval, navigation, and understanding.

Traditional VI systems generally approach the task with a pipeline consisting of three main stages [4]. Firstly, as many other content-based retrieval methods, VI systems use feature extraction to obtain relevant information from the audio signal. Representations like predominant melody, pitch class profiles (PCP), or the constant-Q transform (CQT) have proven useful for this initial step [5–7]. Secondly, traditional VI systems use various post-processing strategies for achieving transposition, tempo, timing, or structure invariance [8–10]. Thirdly, for estimating similarity between pairs of songs, VI systems use segmentation strategies or local alignment methods, which also introduce invariance with regard to musical piece structure [10–12]. Further approaches have explored combining the information obtained from different features and/or different alignment schemes with early or late fusion techniques [12–14]. These, together with some previous solutions, achieve good performance in different evaluation contexts but, nonetheless, have difficulties in scaling to datasets above tens of thousands of songs [15]. With the release of the SHS dataset [16], researchers explored scalable approaches based on audio hashprints, the 2D Fourier transform, or motif-finding strategies [9, 17, 18], but those achieved a limited success compared to their predecessors.

Recent deep learning approaches for VI aim to provide systems that are both accurate and scalable. In general, they focus on learning accurate, low-dimensional embeddings of recordings for, later, estimating similarities with basic distance metrics, with the intention to exploit existing scalable nearest-neighbor libraries. Xu et al. [19] and Yu et al. [20] train their convolutional networks in a multi-class classification fashion, where each version group (or clique) is considered a unique class, and use PCP and CQT as their inputs, respectively. For evaluation, they use the representations obtained from the penultimate layer of their network as embeddings. Beyond classification-based strategies, deep metric learning approaches with contrastive and triplet losses are becoming popular for VI. Qi et al. [21] use a convolutional network with PCPs as input and a triplet loss as the objective function. As an alternative to using PCP variants, Doras & Peeters [22] use a 2D predominant melody representation as input to their convolutional network, which is also trained with a triplet loss but using an online semi-hard triplet mining strategy.

In this paper, we propose a music embedding method that allows for both accurate and scalable VI. We call it MOVE: musically-motivated version embeddings. MOVE achieves state-of-the-art results on two publicly-available benchmark sets and, since it is based on Euclidean distances, allows for efficient retrieval and indexing using existing libraries. The architecture of MOVE introduces a number of improvements, including (1) a relatively novel input representation that has not been explored in the context of deep metric learning for VI, (2) a multi-channel adaptive attention mechanism that is an alternative to previously-used temporal aggregation strategies, and (3) a non-parametric batch normalization at the last layer to yield a standardized embedding space. The training of MOVE, like other recent VI systems, is done with a triplet loss. However, in contrast to those, it uses an online hard triplet mining strategy. In order to learn invariances with respect to the modifiable musical characteristics, MOVE is trained with a VI-specific data augmentation strategy. To gain insight, we perform an ablation study and also investigate the role of embedding dimensionality. To enable further research, we evaluate our method on publicly-available datasets and provide our code at `https://github.com/furkanyesiler/move`.

[*] Work done while at Telefónica Research, Barcelona.

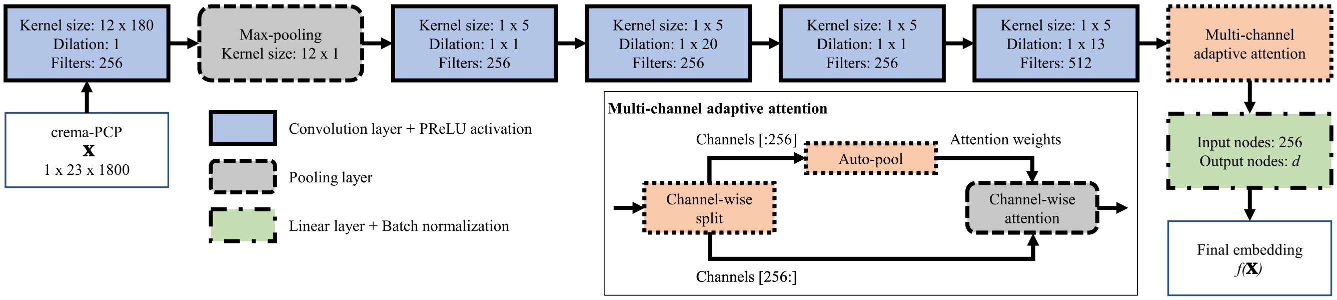

Fig. 1. Block diagram of MOVE's architecture.

2. MUSICALLY-MOTIVATED VERSION EMBEDDINGS

2.1. Input

We use as input a relatively novel PCP variant: crema-PCP. This representation is constructed by using the output of an intermediate step of the crema chord estimation model [23]. For each frame, the crema model estimates the root, the bass, and the pitch classes, which are later combined to output a single chord. Specifically, crema-PCP is constructed by taking the sigmoid activation values of pitch classes for each frame, and considering them as the energy values of each pitch class [23]. Although being a fairly new approach, crema-PCP has been shown to outperform elaborate PCP representations in some benchmarking experiments [15]. We use the pre-trained model available at `https://github.com/bmcfee/crema` (version 0.1.0) and denote the obtained output by $\mathbf{X} \in [0,1]^{12\times T}$, where T is the number of frames using non-overlapping windows of 93 ms. For training, we take random patches of $T = 1800$ frames after applying data augmentation (see below) to a full song. At inference time, we give entire tracks to the model without picking random patches of a particular length (preliminary experiments showed that the below-proposed temporal pooling strategy was also effective with entire tracks at inference time).

2.2. Network architecture

MOVE consists of 5 convolutional blocks with PReLU activation functions and no padding, interleaved by two different pooling layers (Figure 1). A linear layer followed by non-parametric batch normalization produces the final embedding. With the current best setup, the total number of parameters is 6.3 M. We now motivate and present the key components of MOVE.

Transposition-invariant architecture — Following the strategy proposed by Xu et al. [19], we increase the dimension of the crema-PCP inputs $\mathbf{X}$ from $12\times T$ to $23\times T$ by concatenating two copies of $\mathbf{X}$ in the pitch dimension and removing the last pitch class. The first convolutional layer, with a kernel size of 12×180 traverses the input, going through all possible transpositions in the pitch dimension, and the subsequent max-pooling layer, with a kernel size of 12×1, keeps the transposition with the highest activation value (convolutions in MOVE have no padding).

Expanding the receptive field — The 4 convolutional blocks after max-pooling are designed to encode higher-level information and to increase the receptive field of the model (Figure 1). On the one hand, with the layers that have no dilation, we aim to encode higher-level nonlinearities without expanding the temporal context. On the other hand, with the layers that have dilations 20 and 13, we increase the receptive field, which after max-pooling is less than 17 s, to approximately 30 s. Notice that this temporal span could be already sufficient to detect musical piece versions, at least from a human perspective. However, to process an even larger time span, and to be able to deal with different lengths T at test time, we still perform an additional step.

Summarizing temporal content — We consider the convolutional part of our network as a feature extractor that processes the input to obtain a representation that is invariant to the modifiable musical characteristics mentioned in Section 1. In order to summarize the values of each feature in the temporal dimension, unlike previous approaches that use average- or max-pooling variants [20, 22], we propose a multi-channel adaptive attention mechanism, which combines multi-channel temporal attention [24] with auto-pool [25]. The first idea is to let the network compute (and learn) the importance of each time step independently for each feature with an attention-like mechanism [24]. The second idea is to apply a non-linear, learnable pooling function with a scaling parameter before the softmax function [25] such that, depending on the value of such parameter, the function pivots between average- and max-pooling. In practice, temporal summarization is done by calculating channel-wise attention weights, which correspond to the first half of the filters of the last convolutional layer, using the auto-pool function, and utilizing the result to weight the last half of the filters of the same layer. Splitting the hidden representation channel-wise into two halves, $\mathbf{H} = [\mathbf{H}_a\ \mathbf{H}_b]$, this corresponds to

$$\mathbf{H}' = \sum_{t=1}^{T} \sigma(\alpha \mathbf{H}_a) \odot \mathbf{H}_b,$$

where the sum is taken across the temporal dimension, σ corresponds to the softmax function, α is a learnable parameter which we initialize to 0 (equivalent to average-pooling), and $\odot$ is the element-wise product.

Standardizing embedding components — For deep metric learning approaches using a triplet loss, it is highly important to take into account the volume of the hyper-dimensional space where the embeddings lie, specially during training. For instance, if the magnitude of the distances and the margin are disproportionate, the training process may not be able to structure the latent space in an effective way. With these motivations in mind, we propose to use non-parametric batch normalization after the linear layer that finalizes the encoding process. By doing so, we aim to obtain zero-mean and unit-variance components in our embeddings, yielding a statistically-standardized latent space volume. This, together with dimension-normalized Euclidean distances, may also allow us to develop some intuition regarding the loss values and the corresponding margin.

2.3. Training strategy

MOVE is trained by minimizing the triplet loss

$$L = \max\left(D\left(\mathbf{X}_A, \mathbf{X}_P\right) - D\left(\mathbf{X}_A, \mathbf{X}_N\right) + m, 0\right) \qquad (1)$$

using

$$D(\mathbf{X}_i, \mathbf{X}_j) = \frac{1}{d}\left\|f(\mathbf{X}_i) - f(\mathbf{X}_j)\right\|^2,$$

where $\|\ \|$ corresponds to the Euclidean norm and $f(\mathbf{X})$ denotes an embedding of size d produced by our model. Equation 1 aims to make the distance between an anchor A and a positive example P smaller than the distance between the same anchor A and a negative example N under a margin m. We now present our decisions regarding training data, data augmentation, triplet mining, and hyper-parameters.

Training data — We use a private collection of 97,905 songs that are divided into 17,999 cliques. The annotations of the songs are under the Creative Commons BY-NC 3.0 license, and obtained with the API of `secondhandsongs.com`. The related metadata can be found at our repository. For training and validation, we created two disjoint sets of cliques, with 14,499 cliques containing 83,905 songs and 3,500 cliques containing 14,000 songs, respectively. All audio files are encoded in MP3 format and their sample rate is 44.1 kHz.

Data augmentation — In order to enhance the learning of MOVE, we apply to each example a data augmentation function specifically designed for VI. Based on the modifiable musical characteristics specified in Section 1 and elsewhere, such function sequentially and independently applies transposition in the pitch dimension, time stretching, and time warping with probabilities 1, 0.3, and 0.3, respectively. Transposition uses the octave-equivalent characteristics of PCP representations and randomly rolls $\mathbf{X}$ in the pitch dimension between 0 and 11 bins. Time stretching uses one-dimensional interpolations in the temporal domain, with a random factor between 0.7 and 1.5. Time warping consists of three mutually-exclusive functions, which either silence, duplicate, or remove frames with probabilities 0.3, 0.4, and 0.3, respectively (silence corresponds to zeroing-out the entire frame). Once selected, these functions are applied on a per-frame basis with a probability of 0.1, 0.15, and 0.1, respectively. All random numbers are sampled using a uniform distribution.

Triplet mining — As discussed in previous works that employ a triplet loss, the characteristics of the triplets in each mini-batch may have drastic effects on learning performance [26, 27]. For our model, we employ an online hard triplet mining strategy [27]. In our implementation, we choose 16 unique cliques and 4 songs per clique, forming a mini-batch of 64. For the cliques that have less than 4 songs, we choose among the already chosen songs of the same clique. Within a mini-batch, we consider all the examples as anchors (A), and select the positive/negative example that has the maximum/minimum distance to the anchor (P and N, respectively; Equation 1). Although Schroff et al. [26] point out that the hardest examples may lead to local minima early in the training, our triplets can be considered "moderate" [27], as they are selected only from the current mini-batch, and therefore do not strictly correspond to the hardest triplets in the dataset. This presumably avoids the aforementioned local minima.

Hyper-parameters and optimization — We train our network for 120 epochs with plain stochastic gradient descent, using an initial learning rate of 0.1 and decreasing it by a factor of 5 at epochs 80 and 100. An epoch is completed when our data loader goes through all possible cliques. However, an important detail to note is that we include the cliques with size between 6 and 9 twice, the ones with size between 10 and 13 three times, and the ones with size 14 or above four times. This is done to increase the probability of every song being introduced to the network at least once per epoch. The margin value m for the triplet loss is 1. As mentioned, we use patches of $T = 1800$ frames for training and an initial auto-pool parameter $\alpha = 0$. If not already specified in Figure 1, the remaining hyper-parameters and implementation details can be found at our GitHub repository [1]. We study the impact of the embedding dimension d in the next section.

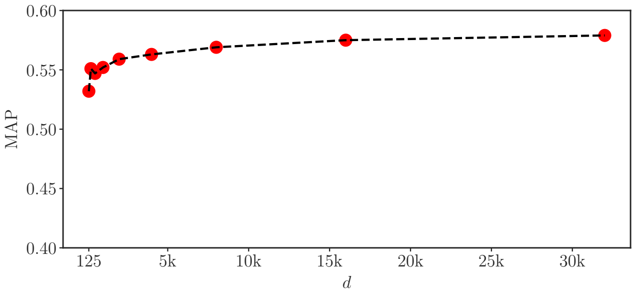

Fig. 2. MAP with respect to embedding dimension d on validation data.

3. RESULTS

3.1. Evaluation methodology

For studying the effect of the embedding dimension and performing the ablation study, we train with a subset of our training set with 8,817 cliques and 44,909 songs in total, and report the performance scores on our validation set. For comparison to the previous work, we utilize the entire training set. To report performance, we use mean average precision (MAP) and mean rank of the first relevant result (MR1). For all the experiments presented in this section, we use the models obtained after the last epochs.

For comparing the performance of MOVE with the state of the art, we use two additional datasets. The first dataset, the benchmark subset of Da-TACOS [15], contains a total of 15,000 songs, with 1,000 cliques of 13 songs each, and 2,000 songs not belonging to any other clique (acting as noise and not queried). The second dataset, YouTubeCovers (YTC) [28], contains 50 cliques with 7 songs each, and comes split into a training and a test set with 250 and 100 songs each, respectively. To compare the performance of our model on YTC with previous works, we follow their approach of only querying the test set to retrieve the versions in the reference set [18–20, 29]. Moreover, in this case, we remove from our training data the 17 cliques that overlap with YTC. After that, both Da-TACOS and YTC do not contain any overlapping cliques with respect to our training/validation data.

3.2. Effect of the embedding dimension

For any embedding system, the size of the embeddings d is a crucial hyper-parameter, as it can have an important effect on model performance. Therefore, we decide to study the model performance on the validation set with respect to it (Figure 2). For this set of experiments, we consider $d = \{128, 256, 512, 1\,\mathrm{k}, 2\,\mathrm{k}, 4\,\mathrm{k}, 8\,\mathrm{k}, 16\,\mathrm{k}, 32\,\mathrm{k}\}$. We observe that performance continues to increase with the embedding dimensionality until it saturates at $d = 16\,\mathrm{k}$. We can place a knee in the curve between $d = 512$ and $d = 2\,\mathrm{k}$.

[1] `https://github.com/furkanyesiler/move`

	MAP	MR1
MOVE	0.575	156
Data augmentation		
1: Without data augmentation	0.540	180
Transposition invariance		
2: Without transposition invariance	0.154	399
Summarizing temporal content		
3: Only multi-channel attention	0.575	153
4: Only auto-pool	0.563	145
5: Max-pooling	0.561	152
6: Average-pooling	0.491	197
Triplet mining strategies		
7: Semi-hard mining	0.545	135
8: Random mining	0.427	167

Table 1. Ablation study. Performance on the validation set using $d = 16\,$k.

3.3. Ablation study

We now analyze the performance of the main components of our network by comparing them to their potential alternatives (Table 1). With that, we aim to quantify the importance of each decision. The first aspect we assess is the effect of the proposed data augmentation strategy (1). We find that removing data augmentation yields a relative decrease of 6% in MAP. The second aspect that we evaluate is the importance of the transposition-invariant architecture explained in Section 2.2 (2). As an alternative, we consider the case where we do not pre-process the input by changing its shape, and remove the max-pooling layer after the first convolution. Although trained with a much smaller learning rate (10^{-4}) and the Adam optimizer, the model was not able to properly learn an effective representation, even though multiple transpositions were present in the data augmentation function. The third aspect we consider is temporal summarization (3–6). We observe that the introduction of the auto-pool parameter α to multi-channel attention does not really change the results (3). In contrast, substituting the proposed multi-channel attention by auto-, max-, or average-pool clearly has an impact (4–6). The final aspect we analyze is the effect of the triplet mining strategy (7–8). To do so, we train our network with online semi-hard (7) and random (8) mining strategies. For semi-hard mining, we pick a random positive example for each anchor and then select a negative example that satisfies the condition $D\left(\mathbf{X}_A, \mathbf{X}_N\right) \leq D\left(\mathbf{X}_A, \mathbf{X}_P\right)$. In case no such negative example exists, we pick a random one. For random mining, we randomly select one positive and one negative example for each anchor. We see that semi-hard and random mining produce a relative MAP decrease of 5 and 26%, respectively. Overall, our ablation study shows that all introduced variations have a positive impact in performance. The only exception is the mixing of the auto-pool parameter with multi-channel attention, which nonetheless does not substantially affect the performance.

3.4. Comparison with the state-of-the-art

Finally, we compare the performance of MOVE with the state of the art (Table 2). The results on Da-TACOS show that MOVE clearly outperforms all considered VI systems. Importantly, this does not only happen for systems that, like MOVE, use a single input representation and alignment, but also for complex systems that employ early or late fusion strategies. The relative MAP difference with respect to LateFusion [14], the most competing system, is over 10%. We also see that, although the best performance is achieved with a relatively large embedding dimension of $16\,$k, a smaller embedding size of $4\,$k can still outperform the state of the art. The results on YTC support the claim that MOVE achieves a new state-of-the-art performance (Table 2). However, we caution about the use of YTC to report VI performance, as differences measured with this dataset may not be significant due to the relatively small number of query and reference tracks (cf. [1]). As an example, MOVE with $d = 4\,$k shows a similar result as the setting with $d = 16\,$k on YTC, while in larger datasets, the latter clearly outperforms the former.

	MAP	MR1
Results on Da-TACOS		
2DFTM [17]	0.275	155
SiMPle [18]	0.332	142
Dmax [14]	0.322	132
Qmax [10]	0.365	113
Qmax* [30]	0.373	104
EarlyFusion [12]	0.426	116
LateFusion [14]	0.454	177
MOVE w/ $d = 4\,$k (ours)	0.495	42
MOVE w/ $d = 16\,$k (ours)	**0.507**	**40**
Results on YTC		
SiMPle [18]	0.591	8
2DFTM sequences [29]	0.648	8
InNet [19]	0.660	6
SuCo-DTW [31]	0.800	**3**
CQT-TPPNet [20]	0.859	**3**
MOVE w/ $d = 4\,$k (ours)	**0.889**	**3**
MOVE w/ $d = 16\,$k (ours)	0.888	**3**

Table 2. Comparison of state-of-the-art VI systems (best results are highlighted in bold). Results on Da-TACOS are taken from [15].

4. CONCLUSION

In this work, we have proposed MOVE, a method for accurate and scalable version identification using musically-motivated embeddings. MOVE achieves state-of-the-art performance on two publicly-available benchmark sets for VI. After motivating the components of its architecture and training strategy, both designed while incorporating a certain degree of domain knowledge, we performed an ablation study to justify our decisions. We have also studied the relation between the embedding size and the performance of our model. As future work, we plan to investigate different input representations. Since some early and late fusion methods incorporate several musical dimensions to outperform their isolated components, we intend to explore possibilities where we can mimic the same idea to improve MOVE's performance. Moreover, considering that our method outperforms traditional VI systems that are built with a certain notion of similarity in mind (for instance, local alignment between tonal features), a future study investigating the similarity concept learned by our model could provide meaningful insight regarding the links that bind various versions originated from the same musical composition.

5. ACKNOWLEDGMENTS

This work is supported by the MIP-Frontiers project, the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 765068, and by TROMPA, the Horizon 2020 project 770376-2.

6. REFERENCES

[1] J. Serrà, *Identification of versions of the same musical composition by processing audio descriptions*, Ph.D. thesis, Universitat Pompeu Fabra, Spain, 2011.

[2] P. Cano, E. Batlle, J. Haitsma, and T. Kalker, "A review of audio fingerprinting," *Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 271–284, 2005.

[3] P. Grosche, M. Müller, and J. Serrà, "Audio content-based music retrieval," in *Multimodal Music Processing*, M. Müller, M. Goto, and M. Schedl, Eds., vol. 3 of *Dagstuhl Follow-Ups*, chapter 9, pp. 157–174. Dagstuhl Publishing, Wadern, Germany, 2012.

[4] J. Osmalskyj, *A combining approach to cover song identification*, Ph.D. thesis, University of Liege, Belgium, 2017.

[5] M. Marolt, "A mid-level melody-based representation for calculating audio similarity," in *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, 2006, pp. 280–285.

[6] F. Kurth and M. Muller, "Efficient index-based audio matching," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 382–395, 2008.

[7] E. Humphrey, O. Nieto, and J. P. Bello, "Data driven and discriminative projections for large-scale cover song identification," in *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, 2013, pp. 4–9.

[8] D. P. W. Ellis and G. E. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, vol. IV, pp. 1429–1432.

[9] T. J. Tsai, T. Prätzlich, and M. Müller, "Known-artist live song ID: a hashprint approach," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 427–433.

[10] J. Serrà, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, pp. 093017, 2009.

[11] E. Gómez and P. Herrera, "The song remains the same: identifying versions of the same piece using tonal descriptors," in *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, 2006, pp. 180–185.

[12] C. J. Tralie, "Early MFCC and HPCP fusion for robust cover song identification," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 294–301.

[13] R. Foucard, J. Durrieu, M. Lagrange, and G. Richard, "Multimodal similarity between musical streams for cover version detection," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2010, pp. 5514–5517.

[14] N. Chen, W. Li, and H. Xiao, "Fusing similarity functions for cover song identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2629–2652, 2018.

[15] F. Yesiler, C. Tralie, A. Correya, D. F. Silva, P. Tovstogan, E. Gómez, and X. Serra, "Da-TACOS: a dataset for cover song identification and understanding," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, 2019.

[16] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, 2011, pp. 628–634.

[17] T. Bertin-Mahieux and D. P. W. Ellis, "Large-scale cover song recognition using the 2D Fourier Transform magnitude," in *Proc. of the Int. Soc. on Music Information Retrieval Conf. (ISMIR)*, 2012, pp. 241–246.

[18] D. F. Silva, M. Y. Chin-Chia, G. E. A. P. A. Batista, and E. J. Keogh, "SiMPle: assessing music similarity using subsequences joins," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 23–29.

[19] X. Xu, X. Chen, and D. Yang, "Key-invariant convolutional neural network toward efficient cover song identification," in *Proc. of the IEEE Int. Conf. on Multimedia and Expo (ICME)*, 2018, pp. 1–6.

[20] Z. Yu, X. Xu, X. Chen, and D. Yang, "Temporal pyramid pooling convolutional neural network for cover song identification," in *Proc. of the Int. Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 4846–4852.

[21] X. Qi, D. Yang, and X. Chen, "Triplet convolutional network for music version identification," in *Multimedia Modeling*, K. Schoeffmann, T. H. Chalidabhongse, C. W. Ngo, S. Aramvith, N. E. O'Connor, Y.-S. Ho, M. Gabbouj, and A. Elgammal, Eds. 2018, pp. 544–555, Springer.

[22] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR), in print*, 2019.

[23] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 188–194.

[24] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series," in *Artificial Intelligence Research and Development*, vol. 308 of *Frontiers in Artificial Intelligence and Applications*, pp. 120–129. IOS Press, Amsterdam, The Netherlands, 2018.

[25] B. McFee, J. Salamon, and J. P. Bello, "Adaptive pooling operators for weakly labeled sound event detection," *IEEE/ACM Trans. Audio, Speech and Language Processing*, vol. 26, no. 11, pp. 2180–2193, 2018.

[26] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: a unified embedding for face recognition and clustering," in *Proc. of the IEEE Int. Computer Vision Conference (ICCV)*, 2015, pp. 815–823.

[27] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *ArXiv: 1703.07737*, 2017.

[28] D. F. Silva, V. M. A. de Souza, and G. E. A. P. A. Batista, "Music shapelets for fast cover song recognition," in *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, 2015, pp. 441–447.

[29] P. Seetharaman and Z. Rafii, "Cover song identification with 2D Fourier transform sequences," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 616–620.

[30] J. Serrà, M. Zanin, C. Laurier, and M. Sordo, "Unsupervised detection of cover song sets: accuracy improvement and original identification," in *Proc. of the Int. Conf. on Music Information Retrieval (ISMIR)*, 2009, pp. 225–230.

[31] D. F. Silva, F. V. Falcão, and N. Andrade, "Summarizing and comparing music data and its application on cover song identification," in *Proc. of Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2018, pp. 732–739.