LESS IS MORE: FASTER AND BETTER MUSIC VERSION IDENTIFICATION WITH EMBEDDING DISTILLATION

Furkan Yesiler[1] **Joan Serrà**[2] **Emilia Gómez**[3,1]
[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Dolby Laboratories, Barcelona, Spain
[3] European Commission, Joint Research Centre, Seville, Spain
furkan.yesiler@upf.edu

ABSTRACT

Version identification systems aim to detect different renditions of the same underlying musical composition (loosely called cover songs). By learning to encode entire recordings into plain vector embeddings, recent systems have made significant progress in bridging the gap between accuracy and scalability, which has been a key challenge for nearly two decades. In this work, we propose to further narrow this gap by employing a set of data distillation techniques that reduce the embedding dimensionality of a pre-trained state-of-the-art model. We compare a wide range of techniques and propose new ones, from classical dimensionality reduction to more sophisticated distillation schemes. With those, we obtain 99% smaller embeddings that, moreover, yield up to a 3% accuracy increase. Such small embeddings can have an important impact in retrieval time, up to the point of making a real-world system practical on a standalone laptop.

1. INTRODUCTION

The concept of music versions is as old as the concept of music itself. Before the existence of recorded music, listening to a piece mostly meant listening to a version of it. Nowadays, with the advancements in recording technologies, most music we listen to comes in recorded form. Nevertheless, musicians keep creating their own versions of existing songs for various reasons, including commercial ones (for example, to attract new audiences), political ones (to connect people or make a stance), and artistic ones (to re-imagine a song with a personal touch).

Version identification (VI) is the task of automatically detecting different renditions of the same underlying musical composition. VI systems are mainly focused on retrieval, aiming to find all renditions of a query song in a reference database. Although creating new versions is common practice, defining the characteristics that enable us to perceive the links connecting different renditions of the same piece is not a straightforward task [1, 2]. Based on quantitative evidence, many successful systems exploit tonal and melodic descriptors that are invariant to the typical differences among versions, including the differences in timbre, tempo, structure, lyrics, and so on [3–5]. By further processing such descriptors, the ultimate goal is to obtain representations that allow inferring links among versions.

The accuracy-scalability trade-off stands as a key challenge in version retrieval. The early, alignment-based systems [6, 7] incorporated musical know-how to capture the similarities among versions, resulting in strong performances. However, due to the scarcity of data, and their dependence on complex representations and computationally-intensive algorithms, they ended up in limited evaluation environments and, ultimately, not being suitable for industrial-size databases. With the release of the Million Song Dataset [8], researchers were further encouraged to address the scalability issue by exploring embedding-based systems that encode songs into more compact vectors. Although offering significant improvements for the scalability aspect, the performance of such systems failed to match their predecessors [9]. Recent embedding-based systems that use deep learning techniques pave the way to encapsulate the similarities among versions in ways that are both efficient and accurate [5, 10–12].

The main use case of version retrieval in commercial settings is to detect copyright infringement cases in media streaming platforms and live performance venues or events. Such application scenarios require having fast and scalable solutions. For example, more than 500 h of video content are uploaded to YouTube every minute [1], and handling the music licensing aspect of that requires having accurate and scalable systems that can identify the cases where a video includes a copyrighted piece of music.

In this paper, we investigate a number of ways to improve the scalability of existing embedding-based VI systems that use neural networks as encoders. Specifically, our goal is to reduce the size of embedding vectors without compromising the accuracy of the systems. Since embeddings can be pre-computed, reducing their size is crucial to improve data storage and, more importantly, retrieval

[1] https://www.cnbc.com/2018/03/14/with-over-1-billion-users-heres-how-youtube-is-keeping-pace-with-change.html

time. For this purpose, we consider three core state-of-the-art strategies, namely unsupervised dimensionality reduction, neural network pruning, and knowledge distillation. Apart from introducing a number of techniques from other fields to VI research, we also consider a novel knowledge distillation loss for metric learning, which aims to optimize a clustering evaluation metric. Moreover, inspired by transfer learning applications, we propose a technique called latent space reconfiguration, to show that learning a compact and efficient latent space is facilitated by using a pre-trained feature extractor due to its stronger priors, compared with using a randomly-initialized one. Our experiments suggest that the performance of a pre-trained network can be preserved, or even improved, while shrinking the embedding vectors down to less than 1% of their original sizes. We evaluate our approach on a publicly-available test set, and share our code, instructions for using a newly-contributed training dataset and supplementary materials (SM) on Github [2].

2. RELATED WORK

2.1 Version identification

Like many other systems in music information retrieval (MIR), VI systems extract audio descriptors to obtain relevant information from signals, including mid-level ones, such as pitch class profiles (PCP) [7, 13–16] and predominant melody [4, 10, 17], or low-level ones, such as the constant-Q transform (CQT) [18–20]. To achieve invariance against the changes in musical characteristics, further processing steps have been proposed, including beat-synchronous features for handling tempo variations [13,15, 21], and the optimal transposition index for handling pitch transpositions [6, 7, 14, 15]. Many rule-based VI systems use alignment algorithms to then compare these representations, resulting in long retrieval times.

Embedding-based VI systems are designed to obtain compact representations that speed up the retrieval phase. Compact embeddings reduce the required storage and facilitate similarity estimation through the use of efficient nearest-neighbor libraries implementing common metrics like Euclidean or cosine distances. Early attempts of such systems use techniques like the 2D Fourier transform, principal component analysis, and linear discriminant analysis for encoding and dimensionality reduction operations [21, 22].

Current deep learning-based systems learn non-linear transformations that map the feature representations into embedding vectors of various sizes, ranging from 300 to 16,000. Xu et al. [23] and Yu et al. [11] train their convolutional networks with a classification loss, but obtain the embedding vectors to use in the retrieval phase from the penultimate layer of their networks. Doras and Peeters [10], and Yesiler et al. [5] formulate the network training as a metric learning setting, in which they use the triplet loss for optimizing distances among training samples.

[2] https://github.com/furkanyesiler/re-move

2.2 Metric learning

This line of research is concerned with learning functions that produce low distance values between semantically similar data points, and high values otherwise. The early approaches include learning Mahalanobis distances [24] with linear [25, 26] or non-linear [27] transformations. Parametrizing such transformations with neural networks was pioneered by Chopra et al. [28] and Salakhutdinov and Hinton [29], and the research domain combining these two concepts is often called deep metric learning.

Deep metric learning methods offer new solutions regarding how to exploit the semantic relations among data, often by formulating or revising loss functions. The triplet loss [30], similar to LMNN [26], manipulates the distances between genuine and impostor pairs with an energy-based approach. The ProxyNCA loss [31], similar to NCA [25], and NormalizedSoftmax loss [32], similar to cross entropy loss, maximize the likelihoods of samples being close to particular class proxies. The group loss [33] incorporates the idea that similar elements should belong in the same class by using replicator dynamics [34]. Although there is a variety of deep metric learning losses, each one with distinctive advantages, the triplet loss variants remain the popular (and almost unique) choice in MIR research.

2.3 Model reduction

2.3.1 Neural network pruning

Pruning a large neural network can preserve the original performance while eliminating more than 90% of its weights [35–38]. The main challenge is to identify the importance of connections and weights, and previous techniques explored the use of absolute weight magnitudes [36–38] and the Hessian of the loss function [35]. Pruning operations can be performed layer- or network-wise, in a one-shot or an iterative fashion, and combined with quantization or clustering. To the best of our knowledge, network pruning has not been considered for VI, nor further explored in MIR systems in general.

2.3.2 Knowledge distillation

Bucilă et al. [39], and later Hinton et al. [40], explored the idea where a small neural network (the student model) is trained with the guidance from a wide, deep, and better-performing network (the teacher model). In the metric learning context, some works explored this idea with a slightly changed formulation: Classical knowledge distillation methods use teacher networks to guide the students on individual examples, but metric learning methods exploit similarity relations among samples. For this, researchers proposed methods that match a number of properties between the embeddings obtained from the teacher and the student models, including the ranks of retrieved samples [41], distances between samples [42], class likelihood distributions [43], and absolute positions of embeddings in the latent space [44]. With few exceptions [45,46], distillation techniques are largely under-explored in MIR, and we believe that no attempt has been done within VI.

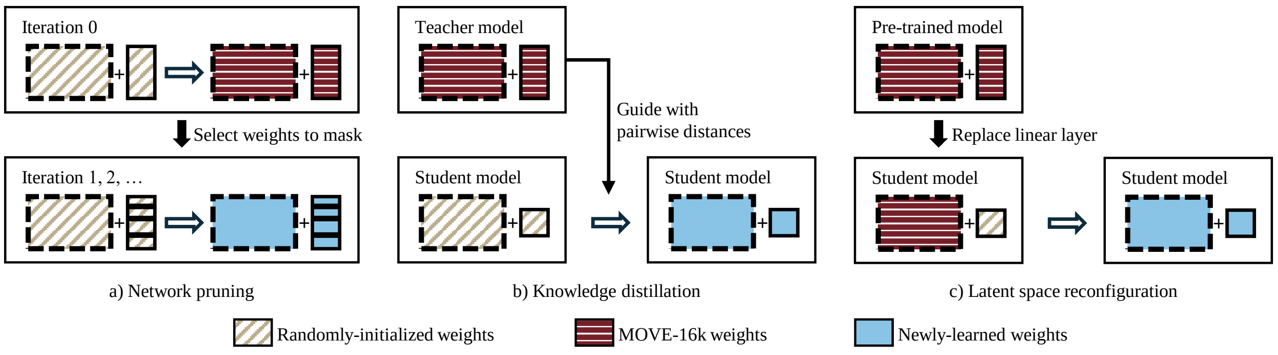

Figure 1: An overview of neural network-based embedding distillation methods. The hollow arrows denote training process, the boxes with dashed and with solid outlines denote feature extractors and fully-connected layers, respectively.

3. METHODOLOGY

3.1 Embedding distillation

Our study focuses on a set of techniques for improving the scalability of existing VI systems in the retrieval phase by reducing the size of the embeddings, rather than building a novel network architecture. [3] We hypothesize that a high-capacity encoder is still needed to extract the essential information from complex and noisy signals such as current tonal representations. However, once a reliable encoder is obtained, it can be used for training a second model that outputs embeddings with a lower dimensionality, ideally without compromising accuracy. Due to the goal of having smaller embeddings that yield a similar performance, we call this set of methods embedding distillation techniques.

3.2 Data

Our models are developed with the Da-TACOS training set that we make available under Creative Commons BY-NC-SA 4.0 license together with this publication. It features a training partition of 83,904 songs in 14,499 cliques (or unique works), and a validation partition of 14,000 songs in 3,500 cliques. The annotations for the songs are obtained using the API of `secondhandsongs.com`. We share the crema-PCP features [47] and the related metadata. Further detail on the contributed dataset is available in SM.

3.3 Model architecture and training details

Our methods require to start from a pre-trained and sufficiently reliable model. For this, we take advantage of the publicly-available MOVE model [5], together with its pre-trained weights. Nonetheless, we believe all methods introduced here can be applied to other embedding-based systems using neural networks (initial results are available in SM).

MOVE uses crema-PCP features $\mathbf{X} \in [0,1]^{12\times T}$ as input, where T is the number of frames, pre-computed with non-overlapping windows of 93 ms. The model outputs embedding vectors $\mathbf{v} = f(\mathbf{X}) \in \mathbb{R}^d$, where d is the embedding size. The original work reports results for d between 128 and 32 k, and shows a clear accuracy drop for $d < 2048$ (the final model employs a rather high dimensionality $d = 16\,\text{k}$). In contrast, the dimensionalities we consider in this work are $d = \{128, 256, 512, 2048\}$.

To find a suitable learning rate and an optimizer for each experiment setting, we perform a grid search over both stochastic gradient descent and Ranger [4] optimizers and initial learning rates in {0.0001, 0.001, 0.01, 0.1}, using our validation set. The full training lasts for 70 epochs, and we decrease the learning rate by a factor of 10 at epochs 50 and 60. We save the model weights that result in the best performance on the validation set. The remaining training details and design decisions follow the ones made by the MOVE authors [5]. All neural network models are trained using PyTorch [48], and the hyper-parameter values used for each experiment can be found at our repository.

3.4 Methods for embedding distillation

3.4.1 *Classical unsupervised techniques*

Before going into complex solutions, we investigate the benefits of using classical dimensionality reduction techniques for embedding distillation. For this, we use principal component analysis (PCA), independent component analysis (ICA), and Gaussian random projection (GRP) techniques. Each model is fit using the training set embeddings obtained with MOVE-16k, and applied to the evaluation set embeddings. We use the implementations from the scikit-learn library [49] and change only the number of target components.

3.4.2 *Pruning*

Based on the approach of Han et al. [37], we study whether we can prune the dimensions of the latent space constructed by MOVE-16k in an iterative way. Although pruning the weights of all layers throughout the network is the most common practice, the underlying idea can be applied

[3] To illustrate the benefits of using smaller embeddings, consider computing distances between a query and a reference database with 10 M embeddings. This takes us (with a simple brute-force, double-loop Euclidean distance function) 0.32 s using $d = 256$, but the elapsed time increases up to 4.75 s for $d = 4k$, and to 18.43 s for $d = 16k$ (the embedding size of MOVE [5]). Although the absolute values are subject to change based on computational resources, for real-world applications on portable devices, such differences in magnitude for the retrieval time (from 0.32 to 18.43 s) may drastically affect user experience and product appeal.

[4] https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer

to only the final linear layer of the model in order to obtain embeddings with fewer dimensions. Denoting the weights of the linear layer of MOVE-16k as $\mathbf{W} \in \mathbb{R}^{d \times f}$, where d is the size of the embeddings and f is the number of input connections to the linear layer, we compute the mean of the absolute values per row for $\mathbf{W}$ and sort the rows based on these mean values. At the end of each iteration m ($m \in \{0, 1, ...\}$), the weights of the top 50% rows are restored to their initial values from Iteration 0 and retrained. The weights of the bottom 50% rows are zeroed-out and not considered for the next iterations (Figure 1(a)) .

3.4.3 Knowledge distillation

This set of experiments consider MOVE-16k as a teacher model, and our goal is to train from scratch a student model of the same size but with a lower embedding dimensionality. Our approach is formulated in a deep metric learning setting where the guidance of the teacher model is shaped by the distances among samples (Figure 1(b)). In the experiments described next, the weights of the teacher model are frozen, and the weights of the student model are initialized randomly.

Distance matching — Perhaps the most intuitive way of guiding the student model is to match the distances obtained from the student with the ones from the teacher, allowing the two models to have different embedding sizes. In our implementation, we pass the samples in each mini-batch to both models, compute in-batch pairwise distances, and use the mean absolute error between the pairwise distance matrices from the teacher model and the student model to train the latter:

$$L_i^{\mathrm{DM}} = \sum_j \left| D(\mathbf{v}_i^{\mathrm{s}}, \mathbf{v}_j^{\mathrm{s}}) - D(\mathbf{v}_i^{\mathrm{t}}, \mathbf{v}_j^{\mathrm{t}}) \right|, \tag{1}$$

using

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

where $\| \; \|$ represents the Euclidean norm, and $\mathbf{v}_i^{\mathrm{s}}$ and $\mathbf{v}_i^{\mathrm{t}}$ the embeddings of song i obtained with the student and teacher models, respectively.

Cluster matching — Our second knowledge distillation scheme aims to obtain a student model that constructs clusters with both low intra-class and high inter-class distances. Assuming the teacher model holds this desired property, we take advantage of this information to guide the student model. To the best of our knowledge, this distillation criterion has not been explored in previous deep metric learning research.

Our criterion exploits internal cluster evaluation metrics [50]. In the experiments reported here, we use the Davies-Bouldin (DB) index [51], but other cluster evaluation metrics can be used:

$$L_i^{\mathrm{DB}} = \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{D(c_i, c_j)} \right), \tag{3}$$

where σ_i denotes the average intra-class distance, computed with a suitable distance measure D, and c_i denotes the centroid for class i. The DB index yields low values for configurations that have low intra-class and high inter-class distances.

In our implementation, we pre-compute the class centroids using the MOVE-16k embeddings from the entire training set. To match the dimensions of the centroids with the student model embeddings, we train a linear projection simultaneously with the student model. The intra-class and inter-centroid distances are computed using only the samples present in the mini-batch and their respective centroids. After computing DB scores for each class in the mini-batch, we average them to obtain the final loss value.

3.4.4 Latent space reconfiguration

Transfer learning applications take advantage of the strong priors learned by the feature extractor part of successful, high-capacity models that are trained on large datasets. Inspired by this idea, we hypothesize that, by using the feature extractor of a pre-trained model, we can obtain a better-structured and lower-dimensional latent space that cannot be obtained by training a randomly-initialized model from scratch.

To test this idea, we use the pre-trained convolutional layers of MOVE-16k as the feature extractor, remove the final linear layer, and learn a new latent space with a randomly-initialized linear layer using a metric learning loss function (Figure 1(c)). Note that the original MOVE-16k model is trained with a triplet loss, meaning that it learned a distance metric parametrized by a neural network. Our approach uses the non-linear part of that metric, and 'reconfigures' the latent space and the distance metric by optimizing a second loss function (hence the name latent space reconfiguration). Our motivation is based on two assumptions: (1) training losses play an important role in shaping the latent space where the embeddings lie, and (2) embeddings with lower dimensionalities may be sufficient to successfully represent semantically meaningful information, as long as the dimensions are effectively utilized. Note that although this technique follows the same procedure as transfer learning, the latter requires, by definition, distinct source and target tasks (or datasets), which is not the case for the proposed technique. Focusing on metric learning schemes, the term latent space reconfiguration denotes the process of starting with an already learned distance metric and modifying it to represent the semantic relations in a more compact embedding space.

In our experiments, we consider 4 loss functions which are explained below. The weights of the feature extractor are frozen during the first epoch and updated with a lower learning rate during the rest of the training. Batch normalization is applied after the linear layer as in MOVE-16k. Apart from using the loss functions below for latent space reconfiguration, we also use them individually and train models from scratch with the same settings to set baseline models.

Triplet loss — We follow the triplet loss formulation used by Yesiler et al. [5]. Distances among vectors are computed using D as specified in Eqn (2):

$$L_i^{\mathrm{T}} = \max \left(D\left(\mathbf{v}_i, \mathbf{v}_+\right) - D\left(\mathbf{v}_i, \mathbf{v}_-\right) + m, 0 \right), \tag{4}$$

where $\mathbf{v}_i$ corresponds to the anchor, $\mathbf{v}_+$ to the positive sample, $\mathbf{v}_-$ to the negative sample, and $m = 1$ is a margin hyper-parameter. For selecting which triplets to use, we follow the hard-positive, hard-negative mining strategy used by the authors.

ProxyNCA loss — Our implementation of ProxyNCA loss [31] also uses the normalized squared Euclidean distance metric from Eqn (2). Every class in our training set is represented with one proxy vector that is initialized randomly and trained simultaneously with the model parameters. In mathematical notation, the ProxyNCA loss can be expressed as:

$$L_i^{\mathrm{P}} = -\log\left(\frac{\exp(-D(\mathbf{v}_i, y))}{\sum_{z\in Z}\exp(-D(\mathbf{v}_i, z))}\right), \tag{5}$$

where $y \in \mathbb{R}^d$ denotes the proxy vector for the class of $\mathbf{v}_i$ and Z denotes the set of proxies for all classes different than the one of $\mathbf{v}_i$.

NormalizedSoftmax loss — As proposed in [32], we implement this function using the cosine distance. We randomly initialize one proxy per class and update their parameters at each training step. We use

$$L_i^{\mathrm{N}} = -\log\left(\frac{\exp(\langle \mathbf{v}_i, y\rangle/\tau)}{\sum_{z\in Z}\exp(\langle \mathbf{v}_i, z\rangle/\tau)}\right), \tag{6}$$

where $\langle\ \rangle$ denotes cosine similarity, $y \in \mathbb{R}^d$ the proxy for the positive class, Z the set of proxies for all classes, and $\tau = 0.05$ the temperature parameter.

Group loss — Following the approach of [33], we use Pearson's correlation coefficient as the similarity metric and replace the negative values with 0. We perform three iterations for refining the class probabilities and, unlike the original implementation, we select one anchor per class in the mini-batch. The main loss is regular cross-entropy:

$$L_i^{\mathrm{G}} = -\log\left(\frac{\exp(l_i^c)}{\sum_{t\in C}\exp(l_i^t)}\right), \tag{7}$$

where l_i^c denotes the logit of sample i with respect to its positive class c, and C denotes the set of all classes in the training set. However, in group loss, logits are updated with replicator dynamics using pairwise similarities [33].

4. RESULTS

4.1 Evaluation methodology

For development, we use the newly available dataset mentioned in Section 3.2 and detailed in SM. Results are then evaluated on Da-TACOS benchmark subset [9], which contains a non-intersecting set of cliques with respect to our training and validation data. Da-TACOS contains 1,000 cliques with 13 songs each and 2,000 noise songs that do not belong to any other clique and are not queried. Following common practice, we report the performance of our models using mean average precision (MAP) and mean rank of the first relevant item (MR1) metrics.

Method	d			
	128	256	512	2048
Baselines (no reduction, training from scratch)				
Triplet	0.459	0.469	0.478	0.487
ProxyNCA	0.168	0.185	0.212	0.206
NormalizedSoftmax	0.445	0.470	0.475	0.422
Group	0.265	0.271	0.269	0.271
Unsupervised				
PCA	0.494	**0.507**	**0.507**	**0.507**
ICA	0.456	0.425	n/a	n/a
GRP	0.429	0.465	0.485	**0.502**
Knowledge distillation				
Distance matching + Triplet	0.492	**0.499**	**0.503**	**0.500**
Cluster matching + Triplet	0.424	0.471	0.465	0.455
Latent space reconfiguration				
Triplet	0.485	0.491	0.494	**0.506**
ProxyNCA	0.424	0.465	0.485	**0.502**
NormalizedSoftmax	**0.513**	**0.524**	**0.525**	**0.525**
Group	0.465	0.483	**0.495**	**0.511**

Table 1: MAP for different embedding sizes d when training from scratch (top) and when using pre-trained models and embedding distillation (middle-bottom). MAPs for the original MOVE-4k and MOVE-16k baselines are 0.495 and 0.507, respectively (values equal to or above MOVE-4k are highlighted in bold).

4.2 Embedding distillation experiments

Table 1 presents the exhaustive list of results for the methods described in Section 3. The baseline results (top block) show that, when training from scratch, changing the loss function of a network causes significant accuracy differences. It should be noted that all alternative losses we consider achieve state-of-the-art performances in computer vision datasets. Nevertheless, our results suggest that they may not generalize across other types of data or tasks, or that they may be oversensitive to hyper-parameters or specific architectural decisions.

For unsupervised dimensionality reduction (second block of Table 1), we find that PCA successfully projects the information contained in MOVE-16k embeddings, even when using 256 dimensions. This suggests that, although achieving state-of-the-art performance, MOVE-16k embeddings contain redundant information that can be drastically compressed. GRP reaches a similar performance as PCA with $d = 2048$, but the resulting performance decreases when using lower-dimensional embeddings.

The initial experiments on pruning reached the same performance as MOVE-16k after one iteration, that is, after reducing the dimensionality by 50%. However, further pruning iterations drastically decreased MAP, up to the point of yielding non-useful embeddings. Therefore, we decided to stop iterating and not to report the corresponding results.

Among the considered knowledge distillation techniques (third block, Table 1), the additional distance matching loss clearly increases the model performance

	d	MAP	MR1
2DFTM [21]	50	0.275	155
Dmax [16]	5.5 k	0.322	132
SiMPle [14]	2.2 k	0.332	142
Qmax [7]	5.5 k	0.365	113
Qmax* [52]	5.5 k	0.373	104
EarlyFusion [15]	8.5 k	0.426	116
LateFusion [16]	5.5 k	0.454	177
MOVE [5]	4 k	0.495	42
MOVE [5]	16 k	0.507	40
Re-MOVE	**256**	**0.524**	**38**

Table 2: Comparison with existing VI systems using Da-TACOS (taken from [9]). When not explicit, embedding sizes d are estimated for a song duration of 3.5 min (see text). Results for the proposed methodology are highlighted in bold.

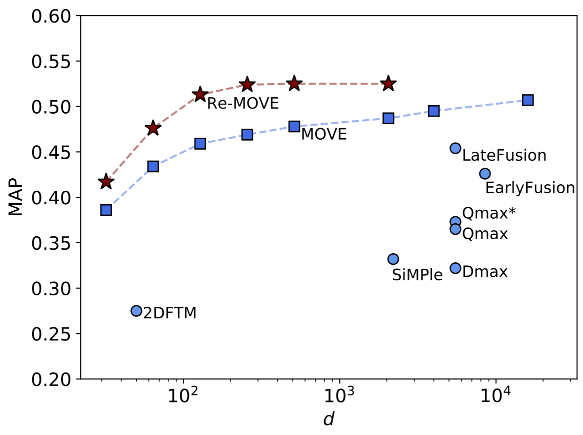

Figure 2: MAP with respect to embedding dimensionality d for Re-MOVE (red stars), MOVE (blue squares), and other existing approaches (blue circles). Notice the logarithmic axis.

compared with the case where only the triplet loss is optimized. However, the same advantage is not observed with cluster matching using DB loss. We hypothesize that this may be related to training an extra linear projection for compressing the centroid embeddings to match the size of the embeddings obtained with the student model.

Latent space reconfiguration results seem to justify our hypothesis regarding the use of strong priors of a pre-trained feature extractor (last block, Table 1). All considered alternatives outperform their baseline counterparts. Moreover, we find that using probabilistic losses such as NormalizedSoftmax and Group for latent space reconfiguration even outperforms the original model while reducing the embedding size by a large margin ($128/16000 = 0.8\%$). Notice that, in addition to these advantages, latent space reconfiguration does not suffer from the setbacks of network pruning and knowledge distillation methods, namely training a model for multiple iterations and using two models simultaneously during training, respectively.

4.3 Comparison with the state of the art

Lastly, Table 2 compares our best result with state-of-the-art methods. The second column, d, shows the size of the smallest representation (per song) required for each method to estimate pairwise similarities (equivalent to the embedding dimensionality). As the results for the first 7 methods are computed with the publicly-available acoss library [9], we use those implementations for estimating the embedding sizes[5]. As the sizes of some representations depend on the song duration (SiMPle, Qmax, Dmax, LateFusion) or tempo (EarlyFusion), we use 3.5 min and 102 bpm estimates, which correspond to average song duration and bpm of the songs in Da-TACOS, respectively.

Re-MOVE, which stands for ‘reduced MOVE’, denotes the model trained with latent space reconfiguration using NormalizedSoftmax. With $d = 256$, it demonstrates relative performance increases of 3%, 6%, and 15% when compared with MOVE-16k, MOVE-4k, and LateFusion systems, respectively (Table 2). We also find that Re-MOVE improves over MOVE for a wide range of dimensionalities $d \in [32, 2048]$ (Figure 2). Along with its state-of-the-art performance, Re-MOVE provides a crucial advantage in terms of scalability, which positions it as the most viable system from a practical point of view.

5. CONCLUSION

In this work, we have studied a set of techniques for reducing the embedding sizes of existing VI systems, which we consider under the name embedding distillation. We have claimed that by using a pre-trained and high-capacity network, we can train a second network that yields smaller embedding vectors without a decrease in performance. To investigate this idea, we have studied a wide range of techniques, including classical dimensionality reduction, neural network pruning, and knowledge distillation methods. Moreover, we have introduced latent space reconfiguration, which is a technique that builds upon the non-linear part of a distance metric learned by a pre-trained network to construct a compact latent space with fewer dimensions. Our results show that it is possible to reduce the embedding dimensionality of a model while maintaining, or even surpassing, its performance.

As future work, we plan to investigate further techniques for compressing entire networks rather than just embedding vectors. We emphasized the importance of having smaller embeddings for real-world applications, and we plan to demonstrate it further in carefully-designed version retrieval scenarios that mimic real-world use cases. Lastly, we believe that optimizing the existing methods to make them applicable in industrial scenarios is a valuable research direction, and we hope to facilitate bridging the gap between academy and industry in MIR research.

[5] For 2DFTM, the acoss library uses a 450-dimensional embedding while the authors apply PCA to reduce the dimensionality to 50.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] D. R. Madoery, "Los procedimientos de producción musical en música popular," *Revista del ISM*, vol. 1, no. 7, pp. 76–93, 2000.

[2] K. Mosser, "Cover songs: ambiguity, multivalence, polysemy," *Philosophy Faculty Publications*, 2008.

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

[4] J. Salamon, J. Serrà, and E. Gómez, "Melody, bass line, and harmony representations for music version identification," in *Proc. of the Int. World Wide Web Conf. (WWW): Int. Workshop on Advances in Music Information Research (AdMIRe)*, 2012, pp. 887–894.

[5] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 21–25.

[6] J. Serrà, E. Gómez, P. Herrera, and X. Serra, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 16, no. 6, pp. 1138–1151, 2008.

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

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

[9] 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. Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 327–334.

[10] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 107–114.

[11] 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 Conf. on Artificial Intelligence (IJCAI)*, 2019, pp. 4846–4852.

[12] F. Zalkow and M. Müller, "Learning low-dimensional embeddings of audio shingles for cross-version retrieval of classical music," *Applied Sciences*, vol. 10, no. 1, p. 19, 2020.

[13] 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)*, vol. IV, 2007, pp. 1429–1432.

[14] 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. Society for Music Information Retrieval Conf. (ISMIR)*, 2016, pp. 23–29.

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

[16] 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.

[17] 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.

[18] Z. Rafii, B. Coover, and J. Han, "An audio fingerprinting system for live version identification using image processing techniques," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 644–648.

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

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

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

[22] E. J. Humphrey, O. Nieto, and J. P. Bello, "Data driven and discriminative projections for large-scale cover song identification," in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2013, pp. 149–154.

[23] 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.

[24] P. C. Mahalanobis, “On the generalized distance in statistics,” *Proc. of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[25] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, “Neighbourhood components analysis,” in *Advances in Neural Information Processing Systems 17 (NeurIPS)*, 2004, pp. 513—-520.

[26] K. Q. Weinberger, J. Blitzer, and L. K. Saul, “Distance metric learning for large margin nearest neighbor classification,” in *Advances in Neural Information Processing Systems 18 (NeurIPS)*, 2005, pp. 1473–1480.

[27] D. Kedem, S. Tyree, F. Sha, G. R. Lanckriet, and K. Q. Weinberger, “Non-linear metric learning,” in *Advances in Neural Information Processing Systems 25 (NeurIPS)*, 2012, pp. 2573–2581.

[28] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *Proc. of the IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 539—546.

[29] R. Salakhutdinov and G. Hinton, “Learning a nonlinear embedding by preserving class neighbourhood structure,” in *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, vol. 2, 2007, pp. 412–419.

[30] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: a unified embedding for face recognition and clustering,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.

[31] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, “No fuss distance metric learning using proxies,” in *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 360–368.

[32] A. Zhai and H. Wu, “Classification is a strong baseline for deep metric learning,” in *Proc. of the British Machine Vision Conference (BMVC)*, 2019, p. 91.

[33] I. Elezi, S. Vascon, A. Torcinovich, M. Pelillo, and L. Leal-Taixe, “The group loss for deep metric learning,” *ArXiv: 1912.00385*, 2019.

[34] J. Weibull, *Evolutionary Game Theory*. Cambridge, MA: The M.I.T. Press, 1995.

[35] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” in *Advances in Neural Information Processing Systems 2 (NeurIPS)*, 1989, pp. 598—605.

[36] S. J. Hanson and L. Y. Pratt, “Comparing biases for minimal network construction with back-propagation,” in *Advances in Neural Information Processing Systems 1 (NeurIPS)*, 1988, pp. 177—-185.

[37] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural networks,” in *Advances in Neural Information Processing Systems 28 (NeurIPS)*, 2015, pp. 1135—1143.

[38] J. Frankle and M. Carbin, “The lottery ticket hypothesis: Finding sparse, trainable neural networks,” in *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2019.

[39] C. Bucilă, R. Caruana, and A. Niculescu-Mizil, “Model compression,” in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 535—-541.

[40] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *ArXiv: 1503.02531*, 2015.

[41] Y. Chen, N. Wang, and Z. Zhang, “Darkrank: Accelerating deep metric learning via cross sample similarities transfer,” in *Proc. of the AAAI conference on Artificial Intelligence*, 2018, pp. 2852–2859.

[42] W. Park, D. Kim, Y. Lu, and M. Cho, “Relational knowledge distillation,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3967–3976.

[43] J. Han, T. Zhao, and C. Zhang, “Deep distillation metric learning,” in *Proc. of the ACM Multimedia Asia*, 2019.

[44] L. Yu, V. O. Yazici, X. Liu, J. van de Weijer, Y. Cheng, and A. Ramisa, “Learning metrics from teachers: Compact networks for image embedding,” in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2907–2916.

[45] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, “DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm,” in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2018, pp. 431–437.

[46] C.-W. Wu and A. Lerch, “Automatic drum transcription using the student-teacher learning paradigm with unlabeled music data,” in *Proc. of the Int. Society for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 613–620.

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

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019, pp. 8024–8035.

[49] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[50] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *IEEE Int. Conf. on Data Mining*, 2010, pp. 911–916.

[51] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[52] 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. Society for Music Information Retrieval Conf. (ISMIR)*, 2009, pp. 225–230.