BYTECOVER2: TOWARDS DIMENSIONALITY REDUCTION OF LATENT EMBEDDING FOR EFFICIENT COVER SONG IDENTIFICATION

Xingjian Du[1†], *Ke Chen*[2†], *Zijie Wang*[1,3], *Bilei Zhu*[1], *Zejun Ma*[1]

[1] Bytedance AI Lab [2] University of California San Diego [3] Zhejiang University

ABSTRACT

Convolutional neural network (CNN)-based methods have dominated the recent research of cover song identification (CSI). A typical example is the ByteCover system we proposed, which has achieved state-of-the-art results on all the mainstream datasets of CSI. In this paper, we propose an upgraded version of ByteCover, termed *ByteCover2*, which further improves ByteCover in both identification performance and efficiency. Compared with ByteCover, ByteCover2 is designed with an additional PCA-FC module, which integrates the capability of principal component analysis (PCA) and fully-connected (FC) neural network for dimensionality reduction of the audio embedding, allowing ByteCover2 to perform CSI in a more precise and efficient way. We evaluated ByteCover2 on multiple datasets in different dimension sizes and training settings, where ByteCover2 beat all the compared methods including ByteCover, even with a dimension size of 128, which is 15 times smaller than that of ByteCover.

Index Terms— Cover song identification, instance-batch normalization (IBN), BNNeck, PCA-FC module, multi-loss training.

1. INTRODUCTION

Cover song identification (CSI) is an important task in the field of music information retrieval (MIR), which aims to identify alternative versions of a given music performance. CSI has a series of downstream applications, such as music recommendation, music copyright protection and music spotlight detection. However, in spite of its importance in MIR, the different musical elements of cover versions (e.g. tempo, key, instrumentation) from the origin music make the CSI problem quite challenging.

Recently, with the development of artificial intelligence, neural-network-based CSI models (e.g., [1–3]) have achieved superior identification performance compared to traditional methods (e.g., [4–7]). Convolutional neural networks (CNNs) have been widely used in these models because its receptive field design can help capture discriminative features from audio samples with little cost. Meanwhile, the pretrained models from the computer vision field (e.g., models pretrained with ImageNet) build a strong recognition prior to boost the establishment and training of the CNN-based audio models [8]. In literature, CNN-based CSI models can be generally classified into two categories. The first category (e.g., [1,2,9]) treats CSI as a multi-class classification problem where each cover group is considered as a class. The second category of methods (e.g., [3]), on the other hand, treats CSI as a metric learning problem and trains CNN models using triplet loss to minimize the distances between cover pairs and maximize the distances between different covers.

In [10], we presented a CNN-based CSI system, i.e., ByteCover, which has achieved new state-of-the-art (SOTA) performances of CSI on all the datasets used in the experiments. However, despite its high effectiveness, we found that there are still rooms to improve the performance of ByteCover, especially considering the use of ByteCover in industrial scenarios where the music database may contain millions of or even tens of millions of music tracks.

Similar to most CNN-based methods, ByteCover utilizes the output of the penultimate layer, which is a fixed-length latent embedding after training, to characterize each audio. During the use of ByteCover, the embeddings of gallery audio tracks are extracted, indexed and stored in advance. Given a music query, the query's embedding is extracted and matched against the references to find the nearest neighbors as cover versions. In this process, the dimension of embedding plays an important role in the trading off between effectiveness and efficiency of CSI. Specifically, shortening the embedding reduces the storage used to maintain the gallery set and lowers the computation cost used for feature indexing and retrieval. However, reducing the embedding length in a brute-force way, e.g., directly decreasing the number of units used in the network, will reduce the representative capacity of the embedding, thus making the CSI performance drop. In the literature of CSI, several works have also been proposed to find a better audio embedding with smaller size, such as REMOVE [11] via knowledge distillation.

In this paper, we also focus on the embedding dimension problem in CSI and take a step further from ByteCover to propose ByteCover2, a more efficient CSI model achieved by the dimensionality reduction of latent embedding. The contributions of ByteCover2 can be listed as follows. First, we introduce a new module termed PCA-FC into the architecture

† The first two authors have equal contribution.

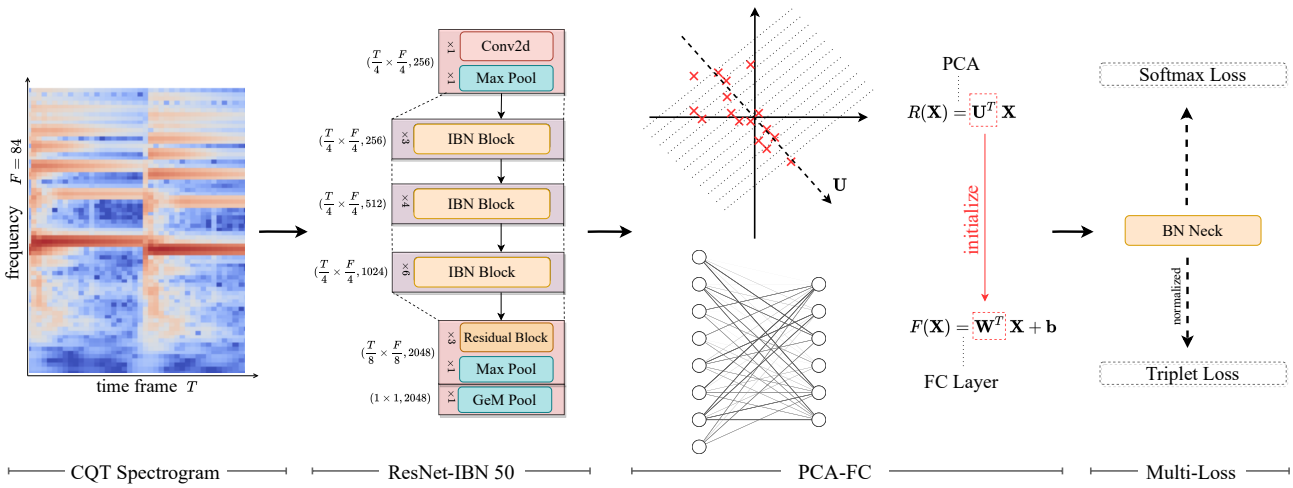


Fig. 1. The model architecture of ByteCover2, which takes CQT spectrogram as input, and comprises of a ResNet-IBN 50 model, a PCA-FC module, and a multi-target loss function.

of ByteCover. The PCA-FC module contains a single fully-connected (FC) layer, with weights initialized by principal component analysis (PCA), which is a classic dimensionality reduction method. PCA-FC not only performs dimensionality reduction on the original audio embedding, but also provides a trainable scenario for freeze-finetuning the model to achieve a better performance even with smaller embedding size. Second, we evaluated ByteCover2 on multiple CSI datasets, including SHS100K [9], Covers80 [5] and Da-TACOS [12], and experimental results show that ByteCover2 can beat all the competitors including the previous SOTA, i.e., ByteCover, even with significantly smaller embedding size. Especially, when using a dimension size of 1536, ByteCover2 achieves a new SOTA performance of CSI on all the datasets, where the mean average precision (mAP) measures are 2.8%, 2.2% and 7.7% higher than those of ByteCover respectively.

2. BYTECOVER2 APPROACH

Fig. 1 shows the overall model architecture of ByteCover2. Inherited from ByteCover, ByteCover2 follows the data-driven approach and adopts the multi-loss learning paradigm to perform CSI among audio queries and galleries. In this section, we first briefly introduce the existing components in the ByteCover: the ResNet50-IBN for embedding extraction and the multi-target loss function for model convergence. Then we introduce our main contribution: the PCA-FC module as an add-on between the above two components.

2.1. Embedding Extractor and Loss Design

As shown in the left of Figure 1, the constant-Q transform (CQT) spectrogram [13] of the audio recording is fed into the model as the input. To calculate CQT, we set the number of bins per octave as 12, the hop size as 512. and the Hann window as the window function. All audio tracks are resampled to 22050 Hz to ensure the consistent sampling rate before processed into CQTs. Subsequently, the CQT is downsampled with an averaging factor of 100 along with the time-axis to initially reduce the latency and improve training efficiency of the model's workflow. As a consequence, the input audio is processed to a compressed CQT spectrogram $\mathbf{S} \in \mathbb{R}^{84\times T}$, where T is related to the duration of the input music track.

The embedding extractor contains a ResNet-IBN backbone and a generalized mean (GeM) pooling layer [10]. The ResNet-IBN is constructed by replacing the residual connection blocks of ResNet50 [14] with the instance-batch normalization (IBN) blocks. In the implementation of ByteCover2, our ResNet-IBN follows the original ByteCover setting with three IBN groups containing 3, 4 and 6 IBN blocks respectively, and a final group with 3 residual blocks. The output of ResNet-IBN before the GeM pooling layer is a 3-D embedding $\mathcal{X} \in \mathbb{R}^{K\times H\times W}$, where K is the number of output channel, H and W are the spatial sizes along the frequency and time axes, respectively. In practical use, we set $K = 2048$, $H = 6$ and $W = T/8$, thus obtaining $\mathcal{X} \in \mathbb{R}^{2048\times 6\times \frac{T}{8}}$. After that, the GeM pooling layer compacts the output $\mathcal{X}$ to a fixed-length embedding vector $\mathbf{f}$, of which the dimension size is 2048 in practical use.

In the design of loss, the original ByteCover combines a softmax classification loss L_{cls} and a triplet loss L_{tri} using the BNNeck method [15] to perform multi-loss training to joint use the advantages of the two categories of CSI methods, that is

$$L = L_{cls}(\mathbf{f}') + L_{tri}(\mathbf{f}) \quad (1)$$

$$= \mathrm{CE}(\mathrm{Softmax}(\mathbf{W}\mathbf{f}'), y) + [d_p - d_n + \alpha]_+, \quad (2)$$

where L is the final loss, CE means cross entropy, d_p and d_n are feature distances of positive pair and negative pair in the triplet setting respectively, $\mathbf{f}'$ is the embedding vector which is normalized by BNNeck, $\alpha = 0.3$ is the margin of triplet loss, and $[z]_+$ equals to $max(z, 0)$.

2.2. PCA-FC Module

Principal component analysis (PCA) [16] is a linear algebra method to transform a group of latent vectors into another

group of vectors with a lower dimensionality, while maintaining the information as much as possible. Formally, let $\mathbf{X} \in \mathbb{R}^{c \times n}$ be a matrix that represents the c-dim embeddings of n audio samples by the CSI model. The covariance matrix $\mathbf{\Sigma} \in \mathbb{R}^{c \times c}$ of $\mathbf{X}$ is obtained by

$$\mathbf{\Sigma} = \frac{1}{n}(\mathbf{X} - \mu \cdot \mathbf{1}^T)(\mathbf{X} - \mu \cdot \mathbf{1}^T)^T + \epsilon \mathbf{I}, \quad (3)$$

where $\mu = \frac{1}{n}\mathbf{X} \cdot \mathbf{1}$ is the mean of $\mathbf{X}$, $\mathbf{1}$ is a column vector of all ones, and $\epsilon > 0$ is a small positive number for numerical stability [17]. After obtaining $\mathbf{\Sigma}$, we compute the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_c$ and their related eigenvectors $p_1, p_2, ..., p_c$ of $\mathbf{\Sigma}$. Then we select the top-k ($k < c$) eigenvalues and take their corresponding k eigenvectors to construct the matrix $\mathbf{U} \in \mathbb{R}^{c \times k}$. The dimensionality reduction is applied by

$$R(\mathbf{X}) = \mathbf{U}^T\mathbf{X}, \quad (4)$$

where $R(\mathbf{X}) \in \mathbb{R}^{k \times n}$ denotes the embedding matrix after dimensionality reduction by PCA, and k is the reduced dimension size from c.

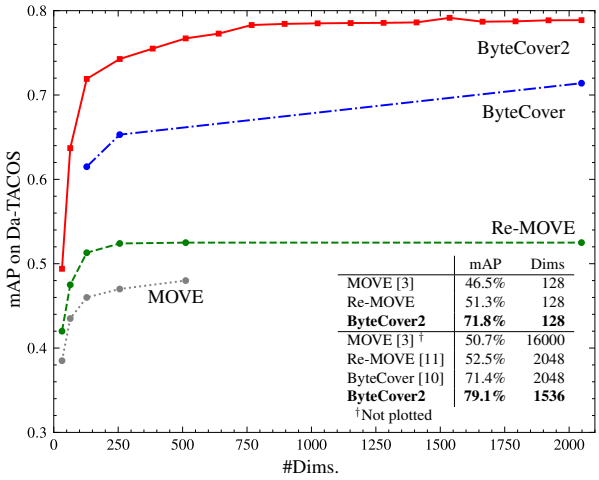
Although PCA can perform dimensionality reduction on audio embeddings, this reduction may lower the CSI performance by two reasons. One potential reason is that the removed redundant dimensions may still contain useful information for discriminating songs. The other reason is the decoupling between the optimization of PCA and CSI model, as PCA can only perform a fixed dimensionality reduction based on fixed calculation methods.

In the design of neural networks, the FC layer can also be used for adjusting the dimension size of embedding. Given an embedding matrix $\mathbf{X} \in \mathbb{R}^{c \times n}$, FC coverts $\mathbf{X}$ to a new embedding matrix $F(\mathbf{X}) \in \mathbb{R}^{k \times n}$ by

$$F(\mathbf{X}) = \mathbf{W}^T\mathbf{X} + \mathbf{b}, \quad (5)$$

where $\mathbf{W} \in \mathbb{R}^{c \times k}$ is a weight matrix, and $\mathbf{b} \in \mathbb{R}^{1 \times k}$ is a bias vector. On contrary to PCA, the neural network provides a high degree of adjustment and update efficiency. However, when initialized randomly or from zero, the FC layer usually performs poorly in maintaining the information from high to low dimensionality. The reason is that the layer lacks strong constraints to serve as a dimensionality reduction module, as there are many pathways to make the model converged.

In this paper, we seek to combine the capacity of PCA and FC for more powerful dimensionality reduction. Specifically, since the matrix $\mathbf{U} \in \mathbb{R}^{c \times k}$ shares the same format as the FC weight matrix $\mathbf{W} \in \mathbb{R}^{c \times k}$ in the neural network, we leverage the PCA-transformation matrix to initialize the weight of FC layer, where the PCA-transformation matrix serves as a strong dimensionality reduction prior for the neural network. We call the PCA-initilized FC layer *PCA-FC*. When we finetune this network, the parameters of the CSI model and the PCA-FC can be jointly optimized using stochastic gradient descent method. This helps find a more optimized weight of the FC layer to perform dimensionality reduction.



	mAP	Dims
MOVE [3]	46.5%	128
Re-MOVE	51.3%	128
ByteCover2	**71.8%**	**128**
MOVE [3] †	50.7%	16000
Re-MOVE [11]	52.5%	2048
ByteCover [10]	71.4%	2048
ByteCover2	**79.1%**	**1536**

†Not plotted

Fig. 2. **Dimensionality vs. Performance.** Our ByteCover2 model achieves better accuracy with a magnitude fewer embedding dimensionality than existing models.

3. EXPERIMENTS

3.1. Training Details and Evaluation Settings

To evaluate ByteCover2, we conducted several experiments on three public datasets: (1) *SHS100K* [9], collected from the *Second Hand Songs* website, consists of 8858 cover groups and 108523 recordings. In our use of SHS100K, we followed the settings of [2] and divided the dataset into the training, validation and test sets with a ratio of 8:1:1. (2) *Covers80* [5], which is a dataset of 160 recordings, consists of 80 songs, with 2 covers per song. We used Covers80 by matching each of the 160 recordings against the full dataset to find the corresponding cover. (3) *Da-TACOS* [12] consists of 15000 music performances, of which 13000 performances belong to 1000 cliques, with each clique containing 13 samples. The remaining 2000 pieces do not belong to any clique.

Our ByteCover2 model was trained on the training set of SHS100K, validated on the validation set of SHS100K, and tested using SHS100K's test set, as well as Covers80 and Da-TACOS. Please note that there are 8373 samples in Da-TACOS overlap with the samples in the SHS100K training subset. We removed these samples from the SHS100K training set and used the retrained model for evaluation. Similar to ByteCover, we implemented ByteCover2 in Pytorch framework and trained it using the Adam Optimizer [18] with the default setting. The learning rate was 0.0001 and the batch size was 64 in NVIDIA Tesla V100 GPUs.

During the retrieval phase, the cosine distance metric was used to estimate the similarity between two musical performances. Following the evaluation protocol of the MIREX Audio Cover Song Identification Contest[1], the mean average precision (mAP) and the mean rank of the first correctly identified cover (MR1) were used as evaluation metrics.

[1] www.music-ir.org/mirex/wiki/2020:Audio_Cover_Song_Identification

Model	#Dims. ↓	mAP ↑	MR1 ↓
	SHS100K-TEST [9]		
TPPNet [1]	300	0.465	72.2
CQT-Net [2]	300	0.655	54.9
ByteCover [10]	2048	0.836	47.3
ByteCover2 - 128	**128**	0.839	45.5
ByteCover2 - 1536	1536	**0.864**	**39.0**
	Covers80 [5]		
Qmax [7]	5500	0.544	-
CQT-Net [2]	300	0.840	3.85
ByteCover [10]	2048	0.906	3.54
ByteCover2 - 128	**128**	0.912	3.40
ByteCover2 - 1536	1536	**0.928**	**3.23**
	Da-TACOS [12]		
Qmax [7]	5500	0.365	113
MOVE [3]	16000	0.507	40
Re-MOVE - 256 [11]	256	0.524	38
Re-MOVE - 2048 [11]	2048	0.525	-
ByteCover [10]	2048	0.714	23.0
ByteCoverFC - 128	**128**	0.615	34.6
ByteCoverPCA - 128	**128**	0.682	36.4
ByteCover2 - 128	**128**	0.718	22.7
ByteCover2 - 1536	1536	**0.791**	**19.2**

Table 1. Performance of different models on three datasets (- means that results are not shown in original papers).

3.2. Comparison on Performance and Efficiency

In the following analysis, we denote a method with different embedding sizes using the form {method}-{embedding size}.

Fig. 2 shows the effect of embedding dimension on the performance of CSI, using MOVE [3], Re-MOVE [11], ByteCover [10] and ByteCover2 as compared methods and Da-TACOS as test set. As illustrated in the figure, for all methods, the mAP results on Da-TACOS mostly increase with the increase of embedding dimension, as a larger embedding can generally hold more information. As for the comparison among different methods, our ByteCover2 model achieves the best mAPs for all the dimension sizes, which clearly proves the effectiveness of our proposed PCA-FC for dimensionality reduction. A counter-intuitive finding is that ByteCover2-128 outperforms ByteCover-2048. Some works in the field of document retrieval have provided an explanation. The space of neural network embeddings tends to be anisotropic, which lowers the performance of retrieval [19]. PCA, as part of the whitening transformation, helps alleviate this problem [20].

The CSI results of ByteCover2 with embedding size of 128 and 1536 on the three test sets are illustrated in Table 1. For comparison, the table also lists the identification results of a few existing models, including Qmax [7], CQT-Net [2], MOVE [3], Re-MOVE [11], TPPNet [1] and ByteCover [10], as well as two ablation versions of ByteCover2, i.e., ByteCoverFC and ByteCoverPCA, which perform dimensionality reduction using FC and PCA directly and separately on the embeddings of ByteCover. The dimension sizes of all these models' embeddings are also presented.

From Table 1, we can see that our ByteCover2 models with embedding sizes of 128 and 1536 both outperform all the compared methods on all three datasets. Especially, the mAP of ByteCover2-1536 significantly surpasses that of ByteCover by 7.7% on Da-TACOS. Even with the smallest embedding size as 128, ByteCover2-128 also achieves higher mAPs and lower MR1s than ByteCover. However, when only using the FC layer or PCA as ByteCoverFC and ByteCoverPCA, the performance is lower than the original ByteCover's. These observations again show the effectiveness of PCA-FC.

Time (ms) / Model	Embedding Extraction		Retrival	Total
	Preprocess	Inference		
Re-MOVE [11]	5352 ± 123	$\mathbf{60 \pm 8.7}$	360 ± 73	5772
ByteCover2-1536	$\mathbf{285 \pm 31}$	108 ± 15.2	2601 ± 384	2994
ByteCover2-128	292 ± 39	105 ± 13.5	$\mathbf{141 \pm 13}$	**538**

Table 2. Time consumption of different models in data preprocessing, model inference and retrieval phases respectively.

An important motivation to propose PCA-FC is to leverage the power of dimensionality reduction to improve the efficiency of CSI. To prove that, we conducted a set of experiments by building a Faiss-based [21] CSI system for ByteCover2 and Re-MOVE separately and comparing the data preprocessing time, model inference time and vector retrieval time. The database used in these experiments contained 1 million music tracks, and the query audios were 1000 45-secs music clips sampled at 44.1 kHz. For each query audio, the top-5 nearest neighbors in the dataset were returned. All experiments were done using a single logical core of the Intel(R) Xeon(R) Platinum 8260 CPU.

Table 2 illustrates the results of the above experiments, where we can see that, compared with Re-MOVE, two ByteCover2 models consume significantly less time for data preprocessing and slightly more time for inference. The reason is that Re-MOVE uses cremaPCP [22] as model input, which is computationally more intensive than CQT, but the model used in Re-MOVE is shallower. As for retrieval time, embedding dimension clearly has a large impact on the speed of nearest neighbor search, as ByteCover2-128 with the smallest embedding size performs the fast among the three methods. Finally, considering the total time used for the whole CSI process, ByteCover2-128 shows the best efficiency, with time consumption being about $1/6$ that of ByteCover2-1536 and $1/10$ that of Re-MOVE.

4. CONCLUSION

In this paper, we propose a simple yet efficient dimensionality reduction method, i.e., PCA-FC, for CSI. The results show that our ByteCover2 based on PCA-FC outperforms all CSI benchmark models on three public datasets, while being highly efficient in embedding extraction and retrieval. As for future work, we are currently studying to apply ByteCover2 to real-world industry applications, where the query audio may suffer from various distortions and pollution. Also, it is of interest to us to analyze the interpretability of our model.

5. REFERENCES

[1] Zhesong Yu, Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang, "Temporal pyramid pooling convolutional neural network for cover song identification," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 4846–4852.

[2] Z. Yu, X. Xu, X. Chen, and D. Yang, "Learning a representation for cover song identification using convolutional neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 541–545.

[3] 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.

[4] 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.

[5] 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.

[6] Joan Serrà, Emilia Gómez, Perfecto Herrera, and Xavier Serrà, "Chroma binary similarity and local alignment applied to cover song identification," *IEEE Transactions 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, pp. 093017, 2009.

[8] Ke Chen, Shuai Yu, Cheng i Wang, Wei Li, Taylor Berg-Kirkpatrick, and Shlomo Dubnov, "Tonet: Tone-octave network for singing melody extraction from polyphonic music," in *ICASSP 2022*.

[9] Xiaoshuo Xu, Xiaoou Chen, and Deshun Yang, "Key-invariant convolutional neural network toward efficient cover song identification," in *IEEE International Conference on Multimedia and Expo*, 2018, pp. 1–6.

[10] Xingjian Du, Zhesong Yu, Bilei Zhu, Xiaoou Chen, and Zejun Ma, "Bytecover: Cover song identification via multi-loss training," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 551–555.

[11] Furkan Yesiler, Joan Serrà, and Emilia Gómez, "Less is more: Faster and better music version identification with embedding distillation," in *Proc. of the Int. Soc. for Music Information Retrieval Conf. (ISMIR)*, 2020.

[12] 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)*, 2019.

[13] Judith C Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2016, pp. 770–778, IEEE Computer Society.

[15] H. Luo, W. Jiang, Y. Gu, F. Liu, X. Liao, S. Lai, and J. Gu, "A strong baseline and batch normalization neck for deep person re-identification," *IEEE Transactions on Multimedia*, pp. 1–1, 2019.

[16] Svante Wold, Kim Esbensen, and Paul Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[17] Lei Huang, Dawei Yang, Bo Lang, and Jia Deng, "Decorrelated batch normalization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 791–800.

[18] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR 2015*.

[19] Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li, "On the sentence embeddings from pre-trained language models," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pp. 9119–9130.

[20] Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou, "Whitening sentence representations for better semantics and faster retrieval," *arXiv preprint arXiv:2103.15316*, 2021.

[21] Jeff Johnson, Matthijs Douze, and Hervé Jégou, "Billion-scale similarity search with gpus," *arXiv preprint arXiv:1702.08734*, 2017.

[22] Brian McFee, "crema: convolutional and recurrent estimators for music analysis," Oct. 2017.