2021 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, OCT. 25–28, 2021, GOLD COAST, AUSTRALIA

DETECTING COVER SONGS WITH PITCH CLASS KEY-INVARIANT NETWORKS

Ken O'Hanlon, Emmanouil Benetos, Simon Dixon

Centre for Digital Music, Queen Mary University of London, UK

ABSTRACT

Deep Learning (DL) has recently been applied successfully to the task of Cover Song Identification (CSI). Meanwhile, neural networks that consider music signal data structure in their design have been developed. In this paper, we propose a Pitch Class Key-Invariant Network, PiCKINet, for CSI. Like some other CSI networks, PiCKINet inputs a Constant-Q Transform (CQT) pitch feature. Unlike other such networks, large multi-octave kernels produce a latent representation with pitch class dimensions that are maintained throughout PiCKINet by key-invariant convolutions. PiCKINet is seen to be more effective, and efficient, than other CQT-based networks. We also propose an extended variant, PiCKINet+, that employs a centre loss penalty, squeeze and excite units, and octave swapping data augmentation. PiCKINet+ shows an improvement of $\sim$17% MAP relative to the well-known CQTNet when tested on a set of $\sim$16K tracks.

1. INTRODUCTION

Given a query track, Cover Song Identification (CSI) seeks to retrieve either a reference, or an alternative, version of the same musical work [1]. CSI is important for attribution of royalties to composers and detection of copyright infringement. The content of a query track and other versions of the same work may differ in many ways, such as instrumentation, tempo, key and lyrics [1], and CSI is therefore considered a challenging task. Early CSI methods compared tracks by taking a measure of a Cross-Correlation Matrix (CCM) as likelihood of tracks being versions of the same work. CCMs were derived from chromagrams [1] or other features [2, 3], with measures calculated using dynamic programming [1, 3, 4], or quicker approaches [5]. An alternative CSI approach was to compare trackwise embeddings that were formed from 2D Fourier transform magnitudes of chromagrams, either directly [6], or in a data-driven fashion [7]. While efficient, performance of embedding methods lagged behind CCM methods.

Deep Learning (DL) methods have recently become the focus of CSI research, with improved performance recorded. While some DL-CSI methods have employed CCMs [8, 9], embedding based approaches are more popular [10–16]. Proposed embedding extraction networks have employed triplet loss functions [12, 13, 15], or classification based learning with the embedding extracted from the penultimate network layer at test time [10, 11, 14, 16].

Tonal input features are typically used in embedding based DL-CSI. Hence, it is desirable that DL-CSI systems are invariant to key shifts between versions of a work. To this end, $f0$ features were aligned to an estimated centre pitch in [12, 15]. Recently, musically structured networks have been proposed [17–19], including key-invariant networks [20–22]. Key-invariant DL-CSI networks [10, 13] exploit the circular key-shift-invariance of chroma features representing the activity of 12 pitch classes. This is effected by circular padding of the chroma before applying a pitch class dimension filter, followed by octave-based pooling that removes the pitch class dimension [10, 13]. Hence, in order to capture the tonal evolution over time, the filter used large temporal dimensions which may be less robust to tempo variation between versions. Semitone Constant-Q Transform (CQT) features [23], from which chroma is often extracted, are also used in DL-CSI. The earlier such networks [11, 14] consider some pitch invariance by using filters of octave-based dimensions in early layers, leading to receptive fields and latent representations of several octaves range. Alternatively, the state-of-the-art ByteCover [16] uses a structure-agnostic ResNet-50, resulting in large frequency range receptive fields. Improvements in ByteCover are sourced mostly from extensions, including mixed cost functions and Instance Batch Normalisation, rather than the network architecture.

In this paper we propose a Pitch Class Key-Invariant Network (PiCKINet) for CSI. PiCKINet affords use of the better-performing CQT input features [14, 16], which are transformed to a pitch class dimension activation map by a key-invariant multi-octave filter. As PiCKINet does not employ octave-based pooling as in [10, 13], there is no need to employ filters with large temporal dimensions; key-invariant convolutions are applied to pitch class dimension activation maps throughout the network. We describe PiCKINet in Section 2 before introducing network extensions that are employed in PiCKINet+ in Section 3. Section 4 describes experiments that show PiCKINet to be an effective network for CSI, while PiCKINET+ compares favourably with the state-of-the-art. We find an approximate harmonic subspace is learnt by the multi-octave key-invariant filter.

This work was supported by Innovate UK [grant number 30863].

978-1-7281-6338-3/21/$31.00 ©2021 IEEE

2. PICKINET

Our proposed network employs a feature extraction module to convert an input CQT spectrogram of a track to a fixed length embedding, similar to [14]. PiCKINet, outlined graphically in Fig. 1, first employs a CQT2PC module to transform the CQT to a pitch class dimension representation. The latent representation is then processed by sequential Pitch Class Blocks (PCBs) that maintain key-invariance throughout. Channel-wise pooling than extracts a feature, that is itself transformed to derive the embedding.

The CQT2PC unit, outlined in Table 1, inputs a magnitude CQT, $\mathbf{X} \in \mathbb{R}^{84\times T}$, spanning 7 octaves at 12 bins per octave with T temporal frames. This CQT can be considered a pitch feature, as each dimension is related to a fundamental frequency on the semitone scale. Large 2D kernels of dimension 73×1 reduce the CQT pitch dimension to a pitch class dimension in the first convolution layer. While such large kernels have previously been used in music processing [24, 25], we are not aware of them being used in this key-invariant fashion. This leads to a tensor, $\mathbf{Y} \in \mathbb{R}^{12\times T\times i}$, where i is the number of channels and 12 is the pitch class dimension. Convolution is then performed with a 3×3 filter, before which circular padding of $\mathbf{Y}$ is performed to effect key invariance:

$$\hat{\mathbf{Y}} \leftarrow [\mathbf{Y}_{12,:,:}\mathbf{Y}_{1:12,:,:}\mathbf{Y}_{1,:,:}] \tag{1}$$

where $[\ ]$ represents a tensor concatenation in one dimension that is indicated by numbered indices while other dimensions display underscored indices, and $\mathbf{Y}_{a:b,:,:}$ represents a slice of $\mathbf{Y}$ containing the elements indexed by the ordered set of integer indices $(a, \ldots, b)$. The resultant $\hat{\mathbf{Y}}$ is also zero padded in the time dimension, such that $\hat{\mathbf{Y}} \in \mathbb{R}^{14\times(T+2)\times i}$, which is reduced to the same dimensions as $\mathbf{Y}$ by convolution with the 3×3 filter. A further pointwise 1×1 convolution is performed in the module. All convolutions are followed by batch normalisation and a ReLU activation. In the work presented here, we set the parameter values $i = 32$ and $o = 64$.

Table 1. CQT2PC and PCB units with optional Squeeze-and-Excite module (SE). Input parameters i and o relate the number of kernels in the initial and further layers, respectively, with c denoting the number of channels in PCB input. T and $\mathcal{T}$ refer to the input, and downsampled, temporal dimension, respectively. Shaded rows are shared by both units.

	CQT2PC		PCB	
Layer	Filter	Outputs	Filter	Outputs
Input	-	$(84, T, 1)$	-	$(12, \mathcal{T}, c)$
Pad(2)	-	-	-	$(24, \mathcal{T}+2, c)$
Conv	73×1	$(12, T, i)$	13×3	$(12, \mathcal{T}, o)$
Pad(1)		$(14, T+2, i)$		$(14, \mathcal{T}+2, o)$
Conv	3×3	$(12, T, o)$	3×3	$(12, \mathcal{T}, o)$
SE	-	$(12, T, o)$	-	$(12, \mathcal{T}, o)$
Conv	1×1	$(12, T, o)$	1×1	$(12, \mathcal{T}, o)$

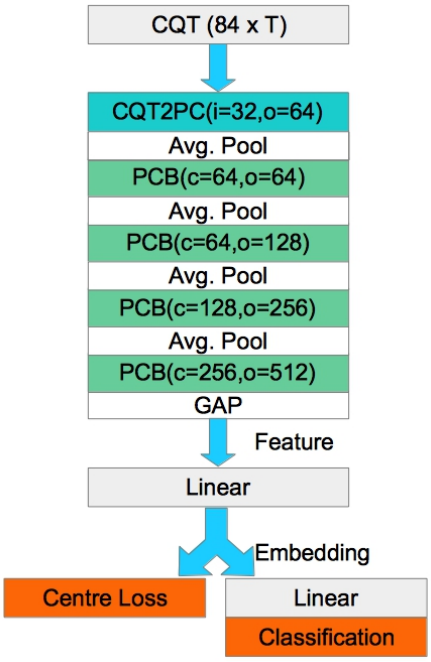

Fig. 1. Schematic of PICKiNet(+) with optional centre loss module.

The PCB unit, also outlined in Table 1, is similar to CQT2PC, apart from the first convolutional layer and prior circular padding for key-invariance. In particular, the convolutional layer employs a kernel of dimensions 13×3, that spans all pitch classes. Given the larger kernel dimensions, a larger padding is required to effect key-invariance:

$$\hat{\mathbf{Y}} \leftarrow [\mathbf{Y}_{1:12,:,:}\mathbf{Y}_{1:12,:,:}] \tag{2}$$

which is used alongside zero-padding in the time direction. Previous key-invariant convolutions across pitch class features [10, 13, 20, 22] used filters with pitch dimension 12 and circular padding such that $\hat{\mathbf{Y}} \in \mathbb{R}^{23\times\mathcal{T}\times i}$, where $\mathcal{T}$ is the size of the downsampled temporal dimension. Here, we employ a slightly different formulation, with filters of pitch dimension 13 applied to a representation padded to a similarly incremented size, $\hat{\mathbf{Y}} \in \mathbb{R}^{24\times\mathcal{T}\times i}$. This results in a similar output size as the former setup, while coefficients repeated in the padded representation are accessed twice by the 1st and 13th coefficients of the filter. Similar to CQT2PC, the module is completed by 2 convolutional layers; the first using a 3×3 filter, after circular (1) and temporal zero padding, and the second using a pointwise 1×1 filter. Likewise, batch normalisation and ReLU activations follow each convolution.

Pooling is performed after CQT2PC and PCB modules. In each case, apart from the final PCB module, average pooling is used. This pooling is effected only in the time direction, with a filter of dimension 1×3, and using a stride of 1×2,

thereby halving the temporal dimension. The final pooling layer extracts the feature vector, $\mathbf{f} \in \mathbb{R}^{\mathcal{C}}$, from the feature map $\mathbf{Y} \in \mathbb{R}^{12\times\mathcal{T}\times\mathcal{C}}$, using Global Average Pooling (GAP):

$$f_c = (12 \times \mathcal{T})^{-1} \sum_{p,t} Y_{p,t,c}. \quad (3)$$

Here, the feature dimension, $\mathcal{C} = 512$, and a learnable linear layer transforms this to an embedding, $\mathbf{e} \in \mathbb{R}^{300}$, similar to [14]. At training time, $\mathbf{e}$ is output to a classification layer, with one neural unit per work, that uses a softmax activation. At test time, $\mathbf{e}$ is emitted by the network and is compared to other stored embeddings using the cosine distance.

An extra advantage of using the pitch class representations throughout is the reduced size of the representations. The kernels employed in the convolutions at the start of each PCB are large, and lead to a increased number of parameters being used in the network. However the smaller representations result in smaller numbers of multiplications being employed in the network, which may be more efficient.

3. PICKINET+

3.1. Centre Loss Penalty

Categorical Cross Entropy (CCE) and triplet losses are popular cost functions for classification and embedding losses respectively, and have been used almost exclusively in DL-CSI for their respective loss type [10–15]. These two losses have also been successfully employed in tandem for CSI [16]. Embedding losses, such as triplet loss, seek to cluster embeddings of similar class. Triplet loss, in particular, seeks to move a given embedding closer to an embedding of a similar class relative to an embedding of a different class. Such approaches require extra sophistication in training, such as mining strategies, and hence are considered slower to train.

Centre loss [26] provides an alternative embedding loss that does not require mining strategies. Unlike contrastive losses, centre loss does not explicitly consider embeddings from other classes. Rather, a centre embedding is stored for each class and the centre loss is given by

$$\mathcal{L}_{centre} = \|\mathbf{e} - \mathbf{m}_{\mathbf{k(e)}}\|_2^2 \quad (4)$$

where $\mathbf{m}_{\mathbf{k(e)}}$ is a centre embedding related to the kth class, to which $\mathbf{e}$ belongs. As often, centre loss is not employed as a lone loss function [26]. Instead, centre loss is used in tandem with CCE [26], as seen in Fig. 1, and is considered as a penalty term added to the classification loss:

$$\mathcal{L}_{total} = \mathcal{L}_{CCE} + \lambda\mathcal{L}_{centre} \quad (5)$$

where λ is a weighting term used to balance the two loss functions. At training time, the loss (4) is estimated and backpropagated through the network. The centre embedding is updated accordingly. After training, the centre embeddings are discarded and the output embeddings for tracks are compared directly at test time using cosine loss, as before.

Layer	Outputs
Input : $\mathbf{Y}$	$(12, \mathcal{T}, \mathcal{C})$
GAP (3)	$\mathcal{C}$
FC / ReLU	$\mathcal{C}/s$
FC / sigmoid	$\mathcal{C}$
Output : $\mathbf{w}$	$\mathcal{C}$

Table 2. Extracting the weighting vector, $\mathbf{w}$, from a Squeeze & Excite module with scale parameter, s.

3.2. Squeeze and Excite Units

Squeeze and Excite (SE) modules [27] are a channelwise attention mechanism which emphasise the focus of the network on channels deemed important through a data-adaptive weighting system. Squeeze refers to the initial operation reducing each channel to a single number, with the particular case of global mean pooling (3) employed [27]. This squeezed vector is passed through an encoder-decoder of 2 fully connected (FC) layers, outlined in Table 2. A scaling factor, s, determines the downsampling in the encoding FC layer which is followed by a ReLU activation. The decoding layer is followed by a sigmoid activation outputting a channel weighting vector, $\mathbf{w} \in \mathbb{R}^{\mathcal{C}}$, which is applied to the input, $\mathbf{Y}$:

$$Y_{p,t,c} \leftarrow Y_{p,t,c} \times w_c. \quad (6)$$

In PiCKINet(+), a SE unit is applied to the outputs of the 3×3 convolutions in CQT2PC and PCB, as seen in Table 1.

3.3. Data Representation and Augmentation

As we consider the CQTNet [14] as our baseline, we employ a similar data representation. Inputs are CQT spectrograms with 12 frequency bins per octave, calculated using librosa [28] with 93 ms frames, prior to downsampling by a factor of 5 through averaging. Likewise, two augmentations employed in [14] are also used here. The first simulates a tempo change through interpolation in the temporal direction of the downsampled CQT. The second considers samples of varying temporal sizes during training, with $T \in \{200, 300, 400\}$ samples. As the lengths of tracks differ, samples are selected starting at any point in a given track.

We also propose a new octave switching augmentation as an extension. Two indices, $i \in (1, ..., 60)$ and $a \in (1, ..., 12)$ are selected from uniform distributions, to relate a point in the first 5 CQT octaves, and how many frequency bins to shift. The augmentation is then effected by swapping CQT chunks

$$\mathbf{X_{i:i+a,\cdot}} \leftrightarrows \mathbf{X_{i+12:i+12+a,\cdot}} \quad (7)$$

where a frequency bins, starting from i, are shifted up one octave, while the corresponding pitch coefficients one octave higher are shifted downwards one octave.

4. EXPERIMENTS

In order to evaluate the proposed approaches, several experiments were run. We first compared the plain networks; PiCKINet & CQTNet, alongside a PiCKINet variant, PNet12, in which the large convolutions in PCBs employ 12×3 filters, rather than 13×3 filters as in PiCKINet, with padding adjusted accordingly. We also tried to train ResNet50, the basic network behind ByteCover [16]. However, the network did not train successfully, perhaps due to implementation details not provided in the paper [16]. We then considered extensions to PiCKINet, including the octave switching (OS) augmentation, SE units, and the centre loss penalty (CL). The extensions were evaluated individually, and compared to PiCKINet+, which employs all extensions. For CL, we found $\lambda = 0.001$ to be a good weighting parameter (5), after initial experiments. We also compared CQTNet+, an extended variant of CQTNet with the same extensions as PiCKINet+.

A private dataset of $\sim$95K tracks, representing $\sim$9K different musical works, was employed. The distribution of tracks per work was not uniform, with several works accounting for more than 50 tracks, while a median of 8 tracks per work was seen. A $5/1$ training / test split was used, resulting in a test set of around 16.2K tracks, and a training set of $\sim$79K tracks, with a similar normalised distribution of numbers of covers per track. The validation dataset comprised Covers80 [29], YouTubeCovers (YTC) [30], and Covers1000 [3] datasets. For all tracks an 84D CQT was extracted, covering 7 octaves from pitch A0 with fundamental frequency 27.5 Hz.

After training, an embedding was extracted for each track in the test set. All pairs of embeddings were compared using cosine distance. For each given embedding, all other embeddings were ranked according to proximity. From these collective rankings several metrics were calculated. Mean Average Precision (MAP), for each track, records the precision at every other version of the same work and takes the average of these precisions, before finally taking the mean of these average precisions over all tracks. MAP is expressed as a percentage, and is a measure that considers all versions of a track. Other metrics here only use the highest ranked correct hits, denoted r_k for the kth track. Mean Reciprocal Rank (MRR), given by $\frac{1}{K}\sum_{k=1}^{K}\frac{1}{r_k}$, where K is the number of tracks, is a general purpose metric that is robust to outliers. Top1 denotes the percentage of queries for which the top ranked track, or nearest neighbour, from the test set is a correct hit. Similarly, Top10 measures the percentage of times that the top ranked correct hit is in the top 10 nearest neighbours of the query.

CQTNet was run using code by its authors [14], available online. All other algorithms were implemented by ourselves, using PyTorch. All networks were trained using the Adam optimiser with default parameters, and set to run for 300 iterations, with the learning rate reduced when the training loss did not decrease for 3 epochs. The final network was selected according to the best MAP metric on the validation set.

Network	Top10	Top1	MRR	MAP
CQTNet	89.3	78.4	0.823	53.7
PNet12	89.5	79.5	0.829	57.5
PiCKINet	**90.5**	**81.0**	**0.840**	**59.1**

Table 3. CSI results for various plain networks in terms of Top10 and Top1 metrics, mean reciprocal rank (MRR) and mean average precision (MAP).

Results for the plain networks are given in Table 3, with PiCKINet seen to improve on CQTNet for all metrics. The difference is largest for MAP, with an improvement of 5.4%, while Top1 increases by $\sim 2.6\%$. It is worth noting the similarity between these two networks which share the same input temporal resolution, number of poolings, and dimensions of extracted features and embeddings. The baseline ResNet50 employed in ByteCover [16] produced a smaller improvement 2.8% MAP relative to CQTNet. This may suggest PiCKINet is a superior base network than ResNet50, although results were on different datasets. A smaller difference was seen between PiCKINet and PNet12. This difference was consistently observed, suggesting that the larger filter is useful.

In Table 4, results are given for PiCKINet using individual extensions, OS, SE & CL and for PiCKINet+ and CQTNet+. It is seen that OS improves MAP by almost 3%, with other metrics improving slightly. The SE units bring slightly smaller improvements than OS. Using CL a larger jump in the metrics is seen, with over 6% MAP and 2.5% Top1. For PiCKINet+, improvements of 11.8% MAP and 5.5% Top1 are seen relative to the baseline PiCKINet, and 17.2% MAP and 8.1% Top1 relative to CQTNet. Interestingly, a complementarity is observed, whereby improvements for the collaborative extensions are similar to, and even slightly larger than, the sum of improvements for the individual extensions. The difference between PiCKINet+ and CQTNet+ is very similar to that between PiCKINet and CQTNet, as the extensions afford similar improvements for both networks, again demonstrating the importance of apt network design.

We then compare PiCKINet+ to the best published results for the Covers80, Covers1000 and YTC datasets. In this case, we use the same training set and employ the previous test set as the validation set. For YTC, results were calculated using 2 versions of each work as references, and 5 versions as

Extensions	Top10	Top1	MRR	MAP
OS	91.0	82.6	0.857	61.9
SE	90.9	82.3	0.853	61.4
CL	91.3	83.5	0.863	65.3
PiCKINet+	**93.5**	**86.5**	**0.888**	**70.9**
CQTNet+	92.0	84.3	0.871	65.6

Table 4. CSI results for PiCKINet with individual extensions, and for PiCKInet+ and CQTNet+ that both use all extensions.

Network	MAP	MR	P@10	MRR	Top1
Covers80					
PiCKINet+	**95.3**	2.0	**0.106**	-	-
ByteCover [16]	90.6	**1.6**	0.093	-	-
YTC					
PiCKINet+	94.3	**2.2**	0.194	-	-
ByteCover [16]	**95.5**	3.5	**0.196**	-	-
Covers1000					
PiCKINet+	–	**5.3**	–	**0.925**	**90.7**
Tralie [3]	–	14	–	0.904	88.4

Table 5. Table comparing PiCKINet+, with all extensions OS, SE, CL, with state of the art on different datasets.

queries [5] [16]. We tabulate results from ByteCover [16] for Covers80 and YTC, and from Tralie's multi-CCM method [3] for Covers1000. Extra metrics are used to afford comparability. Mean Rank (MR) takes the mean of the ranks of the top hits for all tracks $\frac{1}{K}\sum_{k=1}^{K} r_k$, while Precision @10 (P@10) describes the ratio of times a correct hit is found in the 10 nearest neighbours. The comparison is displayed in Table 5, which shows that PiCKINet+ improves over the state-of-the-art on both Covers80 and Covers1000 by reasonable margins. On YTC, PiCKINet+ approaches the results of ByteCover, while we note that ByteCover employed a larger training set. Compared to other metrics, relative performance is inverted for PiCKINet and ByteCove for MR. However, we consider MR a poor metric for DL-CSI methods, as it is less stable from epoch to epoch than other metrics.

Table 6 compares PiCKINet and CQTNet in terms of the numbers of network parameters, and Multiply-ACCumulate operations (MACCs) used for a 3 minute track. PiCKINet requires more network parameters than CQTNet due to using larger pitch class kernels, but uses substantially fewer MACCs due to smaller pitch class latent representations.

The top of Fig. 2 shows the filter learnt at the first convolutional layer of PiCKINet+. It is seen that basis kernels that contain much information across the spectrum are generally not learnt. Rather, many kernels contain either a single, or two proximal harmonics that possess local structure similar to a mexican hat wavelet. Other kernels are relatively flat, and such a mixture of high and low information kernels might go some way to explain why the SE modules are useful in this context. Although full spectrum kernels have not been learnt, any point of the feature map emitted from such a filter does contain full spectrum information through the channels.

Network	#Params	MACCs
PiCKINet	$12.74M$	$4.25G$
CQTNet	$7.29M$	$9.15G$

Table 6. Table showing computational expense metrics for CQTNet and PiCKINet.

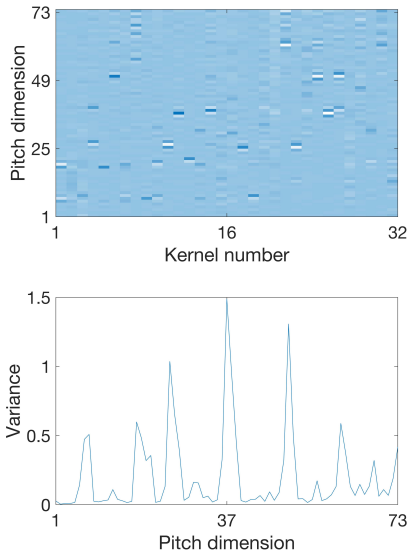

Fig. 2. Top: kernels learnt by best PiCKINet. Bottom: variance in each pitch dimension of the kernels.

The bottom of Fig. 2 shows the variance in the individual pitch dimensions across these kernels. Harmonicity can be more clearly observed here, although there appears to be evidence of two harmonic patterns. In the mid-pitch ranges, four strong harmonics are seen that correspond to a fundamental frequency in the first pitch dimension of the kernel. In the bass range two harmonics, of lower variance, corresponding to a different fundamental frequency are seen. Such emergence of harmonicity is useful as responses to harmonics are localised in feature maps emitted by the filter.

5. CONCLUSION

We have proposed a novel network for DL-CSI using CQT inputs that outperforms other plain networks for the task, while being more efficient. PiCKINet employs key-invariant convolutions throughout the network, and we proposed a variation of these key-invariant convolutions using extended kernels and padding. We think that this may have helped to anchor kernels in the large multi-octave filter towards a similar root, thereby encouraging the emergence of harmonicity across the filter. This harmonic aspect may be worth further investigation in future work, while we think that elements of PiCKINet may be applicable to other tasks. We then introduced PiCKINet+, extending the proposed network using a centre loss penalty, octave swapping data augmentation and SE modules. This resulted in a large performance jump on a large test set, and results comparable to the state-of-the-art on other datasets, while we noticed a complementarity amongst the proposed extensions. Further investigations will seek other such compatible extensions.

6. REFERENCES

[1] J. Serra, E. Gomez, P. Herrera, and X. Serra, "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.

[2] Z. Rafii, B. Coover, and J. Han, "An audio fingerprinting system for live version identification using image processing techniques," in *Proceedings of the International Conference on Audio, Speech and Signal Processing (ICASSP)*, 2014.

[3] C. J. Tralie, "Early MFCC and HPCP fusion for robust cover song identification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[4] J. Serra, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, no. 9, 2009.

[5] D. F. Silva, C.-C. M. Yeh, Y. Zhu, G. E. Batista, and E. Keogh, "Fast similarity matrix profile for music analysis and exploration," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 29–38, 2019.

[6] T. Bertin-Mahieux and D. Ellis, "Large-scale cover song recognition using the 2D Fourier transform magnitude," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[7] E. J. Humphrey, O. Nieto, and J. P. Bello, "Data-driven and discriminative projections for large-scale cover song identification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[8] J. Lee, S. Chang, S. K. Choe, and K. Lee, "Cover song identification using song-to-song cross-similarity matrix with convolutional neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[9] C. Jiang, D. Yang, and X. Chen, "Similarity learning for cover song identification using cross-similarity matrices of multi-level deep sequences," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[10] X. Xu, X. Chen, and D. Yang, "Key-invariant convolutional neural network toward efficient cover song identification," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2018.

[11] Z. Yu, X.Xu, X. Chen, and D. Yang, "Temporal pyramid poolling convolutional neural network for cover song identification," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[12] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[13] F. Yesiler, J. Serra, and E. Gomez, "Accurate and scalable version identification using musically-motivated embeddings," in *Proceedings of the International Conference on Audio, Speech and Signal Processing (ICASSP)*, 2020.

[14] Z. Yu, X.Xu, X. Chen, and D. Yang, "Learning a representation for cover song identification using convolutional neural network," in *Proceedings of the International Conference on Audio, Speech and Signal Processing (ICASSP)*, 2020.

[15] G. Doras and G. Peeters, "A prototypical triplet loss for cover detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[16] X. Du, Z. Yu, B. Zhu, X. Chen, and Z. Ma, "Bytecover: Cover song identification via multi-loss training," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 551–555.

[17] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for f0 estimation in polyphonic music," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[18] V. Lostanlen and C.-E. Cella, "Deep convolutional networks on the pitch spiral for musical instrument recognition," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[19] K. O' Hanlon and M. B. Sandler, "The fifthnet chroma extractor," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[20] A. Elowsson and A. Friberg, "Modeling music modality with a key-class invariant pitch chroma CNN," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[21] J. F. Ducher and P.Esling, "Folded CQT RCNN for real-time recognition of instrument playing techniques," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[22] K. O'Hanlon and M. B. Sandler, "Fifthnet: Structured compact neural networks for automatic chord recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, pp. 2671–2682, 2021.

[23] J. C. Brown, "Calculation of a constant q spectral transform," *Journal of the Acoustic Society of America*, vol. 89, pp. 425–434, 1991.

[24] B. McFee and J. P. Bello, "Structured training for large-vocabulary chord recognition," in *Proceedings of the International Society for Music Information Retrieval*, 2017.

[25] S. Venkataramani, C. Subakan, and P. Smaragdis, "Neural network alternatives toconvolutive audio models for source separation," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017.

[26] Y. Wen, K. Zhang, and Z. Li amd Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Computer Vision – ECCV 2016*, 2016, pp. 499–515.

[27] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and music signal analysis in Python," in *Proceedings of the Python Science Conference*, 2015, pp. 18–25.

[29] D. P. W. Ellis and G. E. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *Proceedings of the International Conference on Audio, Speech and Signal Processing (ICASSP)*, 2007.

[30] D. F. Silva, V. de Souza, and G. E. Batista, "Music shapelets for fast cover song recognition," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2015.