LEARNING AUDIO FEATURES FOR SINGER IDENTIFICATION AND EMBEDDING

Anonymous authors
Paper under double-blind review

ABSTRACT

There has been increasing use of neural networks for music information retrieval tasks. In this paper, we empirically investigate different ways of improving the performance of convolutional neural networks (CNNs) on spectral audio features. More specifically, we explore three aspects of CNN design: depth of the network, the use of residual blocks along with the use of grouped convolution, and global aggregation over time. The application context is singer classification and singing performance embedding and we believe the conclusions extend to other types of music analysis using convolutional neural networks. The results show that global time aggregation helps to improve the performance of CNNs the most. Another contribution of this paper is the release of a singing recording dataset that can be used for training and evaluation.

1 INTRODUCTION

Deploying deep neural networks to solve music information retrieval problems has benefited from advancements in other areas such as computer vision and natural language processing. In this paper, experiments are designed to investigate whether a few of the recent signature advancements in the deep learning community can improve the learning capability of deep neural networks when applied on time-frequency representations to analyze music. Because time-frequency representations are frequently treated as 2-D images similarly to image input for computer vision models, convolution layers are popular choices as the first processing layers for time-frequency representations in audio and music analysis applications. One of the recent convolutional layer variants is the residual neural network with a bottleneck design (He et al., 2016), ResNet. Another variant built upon ResNet is to use grouped convolution inside the bottleneck as a generalization of the Inception Net (Krizhevsky et al., 2012; Xie et al., 2016), ResNeXt. These two variants have enabled more deepening of the convolutional layers of deep neural networks. Most existing music information retrieval research using convolutional neural networks (CNNs), utilizes vanilla convolutional layers with no more than 5 layers. In this paper, the two convolution layer variants mentioned and a deeper architecture with more than 5 convolution layers is proposed and shown to be effective on audio time-frequency representations.

Conceptually, convolution layers take care of learning local patterns (neighboring pixels in images or time frame/frequency bins in time-frequency representations) presented in the input matrices. After learning feature maps from the convolution layers, one of the reoccurring issues, when the input is a time-frequency representation, is how to model or capture temporal relations. Recurrent neural networks has been used to solve this problem (Eyben et al., 2010; Boulanger-Lewandowski et al., 2013; Chan et al., 2015; Choi et al., 2017). Recent developments from natural language processing in attention mechanisms(Bahdanau et al., 2014; Chan et al., 2015; Raffel & Ellis, 2015; 2016) provide a different approach to model temporal dependencies and relations. In this paper, the attention mechanism is viewed as a special case of a global aggregation operation along the time-axis that has learnable parameters. Typical aggregation operations such as $\mathrm{average}$ or $\max$ have no learnable parameters. The effects of global aggregation along the time axis using either $\mathrm{average}$, $\max$ or the attention mechanism is investigated experimentally.

Two specific applications are investigated in this paper: 1) singer classification of monophonic recordings, and 2) singing performance embedding. The goal of singer classification is to predict the singer‘s identity given as input an audio recording. A finite set of possible singers is considered so

this is a classification task. In singer performance embedding the goal is to create a space in which singers with similar styles can be visualized as being closer to each other compared to singers with different styles. Ideally, it should be possible to identify "singing style" or "singing characteristics" by examining (and listening to) the clusters formed from the projections of audio recordings onto the embedding space. Many tasks in music and audio analysis can be formulated in a similar way, therefore we believe that the conclusions of this paper generalize to other audio and music tasks.

1.1 MOTIVATION

The main challenge and interesting point about this application context is how to isolate the "singer effect" from the "song effect". Classic hand-crafted audio features capture general aspects of similarity. When the same song is performed by different singers audio-based similarity tends to be higher than when different songs are performed - i.e the "song effect" (Mesaros et al., 2007). In order to effectively model singing we need to learn a representation that emphasizes singer similarity while at the same time reduces the effect of song similarity.

As an analogy, consider the computer vision problem of face identification. When learning representations for this task we want the information about the identity of the face to be minimally affected by the effect of the environment and pose. The interfering "song effect " is even more dominant in the singing voice case than that of the environment/pose effect in face recognition. Extending this analogy with computer vision, singing performance embedding is analogous to the use of learning an embedded space for face verification (Chopra et al., 2005; Schroff et al., 2015). In this approach, an embedded space of faces is learned with the goal of having pictures of the same person close to each other, and having pictures of different persons away from each other. This is accomplished by utilizing a siamese neural network architecture instead of a classifier (Chopra et al., 2005; Hadsell et al., 2006; Raffel & Ellis, 2016). The large amount of identities make the use of a classifier impractical.

By learning an embedding space for singing voice audio recordings that places recordings of the same identity closer to each other, and pushes the ones with different identities away from each other, ideally "singing style" or "singing characteristics" can be identified by examining (and listening to) the clusters formed from the embeddings of audio recordings in the learned embedding space. For both the singer identity classification and the singing performance embedding, we employ an architecture that uses CNNs to extract features followed by a global aggregation layer after which fully connected dense layers are used.

The difference between the architectures used for these two tasks is that, for singer identity classification, the output layer is the standard $\mathrm{softmax}$ layer that outputs classification probabilities for each singer included in the dataset, but for the singing performance embedding, the output layer is a fully connected linear layer that will embed each input sample into a fixed length vector after which a copy of the network is used to construct a siamese architecture to learn the embedding space. Practically, having a model that embeds singing recordings into short fixed length vectors enables the possibility of fastening the similarity comparison of two long spectrogram sequences (differ in lengths) by calculating the Euclidean distance between their fixed length embedding vectors (Raffel & Ellis, 2016). This allows a large database of singing recordings to be queried by input query singing recordings more efficiently. In order to evaluate the singing performance embedding model in an unbiased way (not biasing towards the collection of songs sang by a singer), a new set of "balanced" singing recordings are gathered and released. The newly released dataset is an addition to the existing DAMP (Smith, 2013) data set of monophonic vocal music performances.

The paper is structured as follows. In section 2, the details of the neural network constructing blocks used in the experiments are described. The dataset used and the experiment details are disclosed in section 3. Discussions and conclusions are in Sec.4.

2 NEURAL NETWORKS DESIGN

The neural network architectures used in the experiments follow a general design pattern depicted in Figure 1. The general design pattern is to feed the input time-frequency features as 2-D images into convolutional layers/blocks, then feed the extracted feature maps to a global time-wise aggregation

layer. The output from the global time-wise aggregation layer is fed into dense layers, followed by the output layer. The details of each construction block are described below.

2.1 CONVOLUTION VARIANTS

The basic convolution layer that is used is the vanilla convolution layer that has shared weights and tied biases across channels without any modification. The other variant that is used in our experiments is the residual network design with the bottleneck block introduced in ResNet proposed in (He et al., 2016). This variant is extended by using the grouped convolutional block, introduced in ResNeXt (Xie et al., 2016), on top of the ResNet. Depictions of the vanilla convolution building block, the ResNet, and ResNeXt are shown in Figure 2. Let the outlets in Figure 2 be y, inlets be x, and f, g, h be convolution operations. The vanilla convolutional block (a) would be $y = g(f(x))$, while the ResNet bottleneck block (b) is $y = x + f(g(h(x)))$ and the ResNeXt bottleneck block is $y = x + \Gamma(x)$ with $\Gamma(\cdot)$ being the grouped convolution consisting of series of concatenated $f(g(h(\cdot)))$ operations over the channel axis of the input. Under the ResNeXt configuration, the ResNet configuration is a special case where the cardinality parameter equals 1. A max pooling layer with pool size $(2, 2)$, and stride of $(2, 2)$ is placed between convolutional blocks in the following way: The first convolutional layer is followed immediately by a max pooling layer, while for all the remaining layers the max pooling layers are inserted between every two consecutive convolutional layers/blocks. A distinction between the terms convolution layer and block needs to be made. A convolutional layer refers to a single convolution layer, while a convolutional block refers to any of the three architecture patterns show in Figure 2. In Table 1, in the column for the number of CNN filters, each number represents the number of output channels for each convolutional layer or block, with normal text for layer and bold text for block. Batch normalizations are applied after each non-linearity activation throughout the convolutional layer/blocks.

Figure 1: An overview of the neural network architecture used in this paper. Before feeding the output of the convolutional layers to the global time-wise aggregation, the $3-D$ feature map having the shape (# of channels, # of time frames, # of frequency bins) as their dimensions will be reshaped as 2-D matrices having the shape (# of time frames, # of channels$\times$ # of frequency bins)

2.2 GLOBAL AGGREGATION

Originally, the attention mechanism was introduced for sequence-to-sequence learning (Bahdanau et al., 2014) in an RNN architecture, that allows the prediction at each time-step to access information from every step in the input hidden sequence in a weighted way. Since the experiments done in this paper do not need sequence-to-sequence prediction, the feed-forward version of attention proposed in (Raffel & Ellis, 2015; 2016) is used instead of the original one. The feed-forward attention is formulated as follows: Given the input matrix $X \in \mathbb{R}^{N \times D}$ representing N frames of D dimensional feature vectors, a weight vector $\sigma \in \mathbb{R}^N$ over the time-steps is calculated by

$$\sigma = \mathrm{softmax}(f(Xw + b)) \tag{1}$$

where

$$\mathrm{softmax}(x)_m = \frac{e^{x_m}}{\sum_{n=1}^{N} e^{x_n}} \tag{2}$$

and f is a non-linear function ($tanh$ for the experiments done in this paper), and $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$ are the learnable parameters, which can be learned by back-propagation. The output $\hat{X}$ of the feed-forward attention layer is then calculated via

$$\hat{X} = \sum_{n=1}^{N} \sigma_n X_n \tag{3}$$

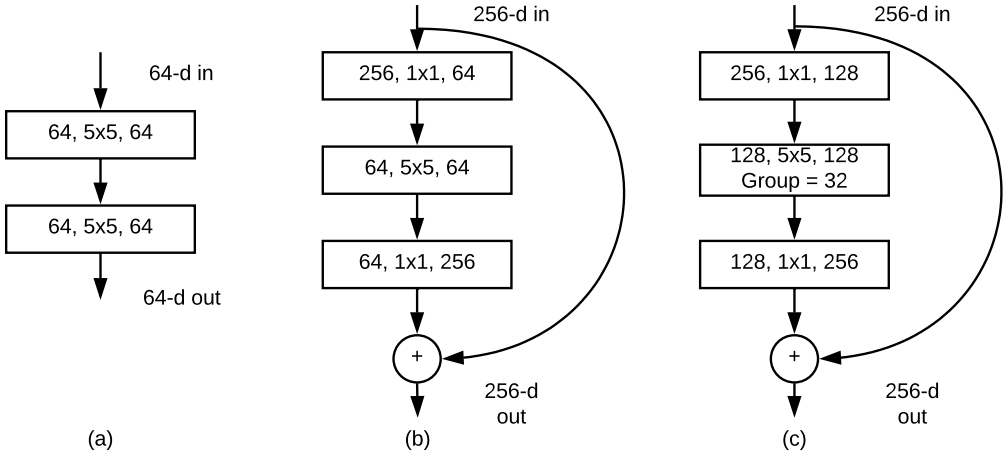

Figure 2: (a) is vanilla 2-layer CNN block, (b) is a ResNet 3-layer bottleneck block and can be seen as a ResNeXt block with cardinality of 1, and (c) is a 3-layer ResNeXt block with cardinality of 4. All three blocks in this example have similar time complexity. The numbers in boxes represent (# of input channels, kernel size, # of output channels.)

and where $\hat{X}$ can be considered a weighted average of X with weights σ, determined by the learnable parameters w and b. This attention operation can also be viewed as an aggregation operation over the time-axis similar to $\max$ or average. The idea of aggregation over a specific axis could then be generalized by having the feed-forward attention, $\max$ and average all in the same family, except that the later two have no learnable parameters. This family of operations is different from the standard max/average pooling in convolution layers, in that the aggregation is global to the scope of the the input sample i.e the aggregation will reduce the dimension of the aggregation axis to 1. A specific realization of the network architecture including both convolutional and the global aggregation parts can be found in Appendix B.

3 EXPERIMENTS

The two tasks explored in this paper are singer identity classification and singing performance embedding. In terms of experimentation with different hyper parameters and network architectures, the singer classification problem provides clear evaluation criteria in terms of model performance. That way different hyper parameters and architectural choices can be compared to each other. On the other hand, the embedding task allows a more exploratory way to understand the input in the sense that it is the spatial relationships between the embedded samples that are interesting to us. For both tasks, numerical evaluation metrics, as well as plots of the embedded samples from the singing performance embedding are provided in order for readers to examine the results both quantitatively and qualitatively.

3.1 DATASETS

The dataset being used for the singer identity classification is the DAMP dataset[1]. The DAMP dataset has a total of 34620 solo singing recordings by 3462 singers with each singer having 10 recordings. The collections of songs sang by each singer are different, and some singers sing the same song multiples times. Therefore the DAMP dataset is "unbalanced", and making it difficult for the learning algorithm not to be biased to the singer-specific collection of songs when learning to predict the singer identity. Therefore the original DAMP dataset has been extended by adding more solo singing performances so that a "balanced" dataset is created with each singer singing the same collection of songs available for training and evaluation. The set of added collections of solo singing recordings is named the DAMP-balanced dataset. DAMP-balanced has a total of 24874

[1] https://ccrma.stanford.edu/damp/

singing recordings sang by 5429 singers. The song collection of the DAMP-balanced has 14 songs. The structure of the DAMP-balanced is that the last 4 songs are designed to be the test set, and the first 10 songs could be partitioned into any $6/4$ train/validation split (permutation) that the singers in train and validation set sang the same $6/4$ songs collections according to the $6/4$ split (the number of total recordings for train and validation set are different from split to split, since there are different number of singers that all sang the same $6/4$ split for different split). The DAMP-balanced dataset is suitable for the singing performance embedding task while the original DAMP dataset can be used to train singer identity classification algorithms. The song list of DAMP-balanced is provided in Appendix A.

3.2 DATA PREPARATION

The input to the neural networks are time-frequency representations extracted from raw audio signals. These time-frequency representations are obtained by applying the short-time Fourier transform that extracts frequency information for each short time window analyzed. As a result, most time-frequency representations take the form of 2-D matrices with one axis representing time while the other axis represents frequencies. The entries $[j, i]$ of the matrix represent the intensity of a particular frequency i corresponding to a particular time frame j. In this paper, the Mel-scaled magnitude spectrogram (Mel-spectrogram)(Hinton et al., 2012) is used as the input feature to the neural network. Mel-spectrogram are used as input to neural network tasks in (Chan et al., 2015; Grill & Schlüter, 2015; Choi et al., 2017). The other common choice of audio time-frequency input, the constant-Q transformed spectrogram (CQT), which is used extensively in music information retrieval tasks (Su et al., 2014) due to its capability of preserving the constant octave relationships between frequency bins (log-scaled frequency bins). Since all neural network configurations using CQT perform worse than their Mel-spectrogram versions, only a few representative results of using CQT are shown in Table 1. The reason why CQT works worse is that although the CQT preserves a linear relationships of the intervals of different pitches, the linear relationships do not apply to the distances between different harmonics of one pitch. Since the audio recording being analyzed here only has one single singing voice at each time frame, the constant octave relationship does not help the neural networks learn the time-frequency patterns for singing voices.

The audio recordings are all re-sampled to have 22050Hz sample rate, then the Mel-scaled magnitude spectrograms are obtained using a Fast Fourier Transform (FFT) with a length of 2048 samples, hop size of 512 samples, using a Hanning window and 96 Mel-scaled frequency bins. The extracted Mel-spectrogram is squared to obtain the power spectrogram which is then transformed into decibels (dB). The values below -60dB are clipped to be zero and an offset is added to the whole power spectrogram in order to have values between 0 and 60.

For both tasks, each singing performance audio recording is transformed to a Mel-spectrogram as described above. The Mel-spectrogram of each recording is then chopped into overlapping matrices each of which has a duration of 6 seconds (256 time steps) and 20% hop size.

3.3 NEURAL NETWORK PARAMETERS

For both tasks the gradient decent is optimized by ADAM (Kingma & Ba, 2014) with a learning rate 0.0001 and a batch size of 32. A drop out of 10% is applied at the last fully connected dense layers. L_2 weight regularizations with a weight $1e-6$ are applied on all the learnable weights in the neural network. The above hyper parameters are chosen by the Bayesian optimization package SPEARMINT (Snoek et al., 2012). For both tasks, an early stopping test on the validation set is applied every 50 epochs. For the singer identity classification, the patience is 300 epochs with at least 99.5% improvement, and the patience for the singing performance embedding task is 1000. The non-linear activation function used in all convolution layers and fully connected layers is the rectified linear unit activation function. The convolutional filter sizes are $(10, 10)$ for the first convolutional layer and $(5, 5)$ for all subsequent convolutional layers. For the fully connected dense layer, 3 layers with each having 1024 hidden units are used before the last output layer.

Table 1: Singer Identity Classification: In the column for the number of CNN filters, each number represents the number of output channels for each convolutional layer or block, with normal text for layer and bold text for block. The actual number of convolution layers for the shallower and deeper architectures for vanilla CNNs are 3 and 13 respectively, and the corresponding numbers for ResNeXt configurations are 4 and 19.

Feature	# of CNN filters	Global aggregation	# of CNN params	# of total params	Train accuracy	Test accuracy
baseline					0.5164	0.2729
CNN	64, **128**	none	621k	200000k	0.7915	0.6276
CNN	64, **128**	max	621k	5867k	**0.9302**	0.7426
CNN(CQT)	64, **128**	max	621k	5867k	0.8864	0.6373
CNN	64, **128**	average	621k	5867k	0.9271	0.7291
CNN	64, **128**	attention	621k	5867k	0.9284	**0.7484**
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	none	708k	28000k	0.8534	0.6534
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	max	708k	3560k	0.8989	0.6943
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	average	708k	3560k	**0.9066**	**0.7083**
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	attention	708k	3560k	0.8945	0.6991
ResNeXt	64, **128**	none	54k	200000k	0.8216	0.6308
ResNeXt	64, **128**	max	54k	5300k	0.8833	0.702
ResNeXt	64, **128**	average	54k	5300k	0.9311	0.7385
ResNeXt	64, **128**	attention	54k	5300k	**0.9313**	**0.7415**
ResNeXt(CQT)	128, **128**	attention	54k	5300k	0.8497	0.5936
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	none	172k	52600k	0.8835	0.6889
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	max	172k	3844k	0.91	0.7051
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	average	172k	3844k	**0.9284**	**0.7315**
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	attention	172k	3844k	**0.9284**	0.7288

3.4 SINGER CLASSIFICATION

A subset of 46 singers (23 males and 23 females) corresponding to 460 solo singing recordings from the DAMP dataset is selected for the singer classification problem. A 10-fold cross validation is used to obtain the test accuracies for different models with each fold using 1 recording from each singer as the test set, while the training is performed on the rest 9 recordings with 1 of them selected randomly as the validation set for early stopping. For the classification task we explore different combinations of neural network configurations in terms of using either the vanilla CNN blocks or ResNeXt building blocks. Also different number of layers and different types of aggregation such as $\max$, average, feed-forward attention or no global aggregation are also investigated. A baseline SVM classifier is also included by having the mean and standard deviation of chroma, MFCC, spectral centroid, spectral roll-off, and spectral flux (Casey et al., 2008) extracted from each ~ 6 second clip as the input. The experimental results and associated measures of different models are displayed in Table 1. The number of convolution filters is chosen so that the total number of parameters are on the same scale between different configurations.

From Table 1, it can be seen that the baseline method achieved 27% accuracy which is clearly above the random prediction of 2.2% ($\frac{1}{46}$), while all the neural network models far exceeded it by at least 35%. For all the neural network models, use of any of the global aggregation methods improved the performance by $5\% \sim 10\%$. Among the neural network models, global aggregation with average or feed-forward attention has slightly better performance than $\max$ except for the shallower CNN.

3.5 SINGING PERFORMANCE EMBEDDING

For the singing performance embedding experiment, a subset of $6/4/4$ train/validation/test split from the DAMP-balanced is used[2]. The total number of recordings and singers for this specific split are $276/88/224$ and $46/22/56$ respectively. We would like an embedding space that places recordings by the same singer closer to each other and pushes recordings by different singers away

[2]The split is according to the index order provided in Appendix A.

Table 2: Singing Performance Embedding

Feature	# of CNN filters	Global aggregation	# of CNN params	# of total params	Test loss
CNN	64, **128**	none	621k	200000k	0.3369
CNN	64, **128**	max	621k	5884k	0.3207
CNN	64, **128**	average	621k	5884k	0.3305
CNN	64, **128**	attention	621k	5884k	0.3279
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	none	708k	28000k	0.3753
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	max	708k	3611k	0.3461
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	average	708k	3611k	0.3544
CNN	16, **32**, **32**, **32**, **64**, **64**, **64**	attention	708k	3611k	0.3697
ResNeXt	128, **128**	none	56k	200000k	0.3351
ResNeXt	128, **128**	max	56k	5318k	0.3216
ResNeXt	128, **128**	average	56k	5318k	0.3207
ResNeXt	128, **128**	attention	56k	5318k	0.3315
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	none	172k	52600k	0.3605
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	max	172k	3861k	0.3423
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	average	172k	3861k	0.3378
ResNeXt	32, **64**, **64**, **64**, **128**, **128**, **128**	attention	172k	3862k	0.3273

from each other, a siamese neural network architecture (Chopra et al., 2005; Hadsell et al., 2006; Raffel & Ellis, 2016) is used. The inner twin neural network is constructed following the same principle described earlier in section 2. The embedding dimension for the linear fully connected output layer is chosen to be 16 by SPEARMINT. Since a siamese network learns the embedding by shortening or lengthening the distance between pairs of embedded vectors based on their label, pairs of samples from the dataset are arranged and labeled. Denote a pair of samples by x_1, $x_2 \in \mathbb{R}^D$, and y a binary label that equals 1 when x_1, x_2 have the same label or identity and equals 0 when their identities are different. The distance metric optimized over the siamese network in this experiment is the squared euclidean distance

$$D(x_1, x_2) = \|G(x_1) - G(x_2)\|_2^2 \tag{4}$$

then the contrastive loss (Chopra et al., 2005; Hadsell et al., 2006; Raffel & Ellis, 2016) is used as the optimization goal and is defined as

$$\mathcal{L}(y, x_1, x_2) = \frac{1}{2} yD + \frac{1}{2}(1-y)max\{0, m - D\} \tag{5}$$

where G is the non-linear function that represents the neural network and m is the target margin between embedded vectors having different identities, and $m = 1$ throughout the experiments. To train the siamese networks, pairs of chopped samples from the same singer or different ones are randomly sampled in a $1:1$ ratio and fed into the siamese networks. The contrastive losses on the test set for different network configurations are shown in Table 2. The cardinalities for the ResNeXt configurations in Table 2 are 4.

The training and validation error over epochs are plotted in Figure 3. The observation from the training/validation plots are that, 1) feed-forward attention and $\mathrm{average}$ aggregation tend to overfit the data more than $\max$ and no aggregation by looking at training errors, 2) feed-forward attention and $\mathrm{average}$ aggregations reach early stopping earlier than $\max$ and no aggregation by looking at the best validation epoch, 3) Shallow architectures work slightly better than deeper ones if their number of parameters are controlled so that they are on the same scale. Results showing qualitative characteristic of the embedding are shown in Figure 4. In Figure 4, the embeddings of 40 performances sang by 10 singers with each singer sang the same 4 songs from the test split are plotted. The embedding of a performance is obtained by taking the mean of the embeddings from all the chopped input samples of that performance. A comparison is made between the embeddings from the shallow ResNeXt architecture with/without feed-forward attention and the handcrafted features used in the baseline case for singer classification. Both the embeddings and the extracted handcrafted audio features are projected down to a 2-D space by t-SNE (Maaten & Hinton, 2008). It is obvious that the baseline handcrafted audio feature captured the "song" effect while the learned embeddings from

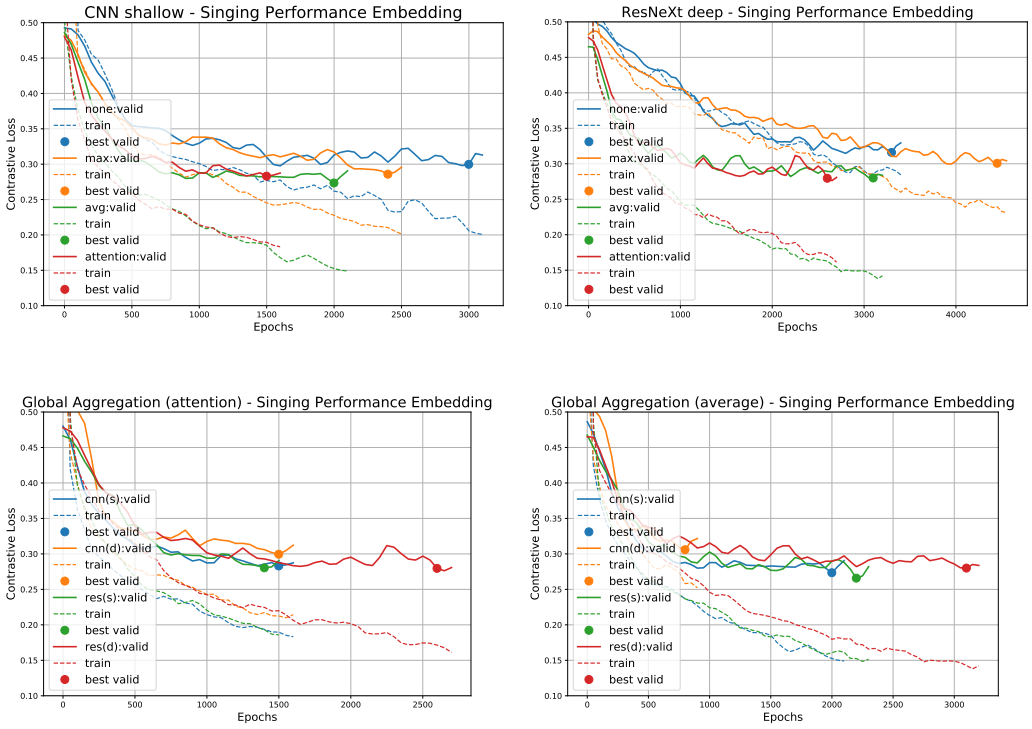

Figure 3: Train/validation losses over epochs on selected subsets of experimented networks configurations. The top shows results from two neural networks configurations with different aggregation methods used. The bottom row depicts two sets of results that have either feed-forward attention or average aggregation with varying CNN blocks choices and depths.

our singing performance embedding experiment were able to group together the performances by the same singers while invariant to the "song" effect. Also the feed-forward global aggregation helped the enhancement of "singer style" while reducing "song effect" slightly. The t-SNE projections of performed clips before summarized into songs are shown in Figure 7 in Appendix B. To have another quantitative assessment of the embeddings, leave-one-out k-nearest neighbor classifications using the embedded 16-dimensional performance vectors are used as training points. For each k and each network configuration, every sample is used as test sample once and the classification accuracies are obtained by averaging over the outcomes of every test sample for all k and network configurations. For the k-nearest neighbor singer classification, all the 224 performances from 56 singers are used. The classification results with multiple ks among the shallow ResNeXt configurations with/without feed-forward attention and the handcrafted features are shown in Figure 5. In addition, k-nearest neighbor classifications on performed songs are also conducted to demonstrate the "song effect". From the k-nearest neighbor classification results on singers and songs, it is evidence that the "song" effect exists and singing performance embedding learning is able to dilute the "song" effect while extracting features that are more relevant in terms of characterizing singers. Also it is worth mentioning that the k-nearest neighbor classification on performed songs is only possible due to the "balanced" nature of the dataset.

4 Discussion and Conclusions

In this paper, empirical investigations into how recent developments in the deep learning community could help solving singer identification and embedding problems were conducted. From the experiment results, the obvious take away is that global aggregation over time improves performance by a considerable margin in general. The performances among the three aggregation strategies; max, average and feed-forward attention, are very close. The advantage of using feed-

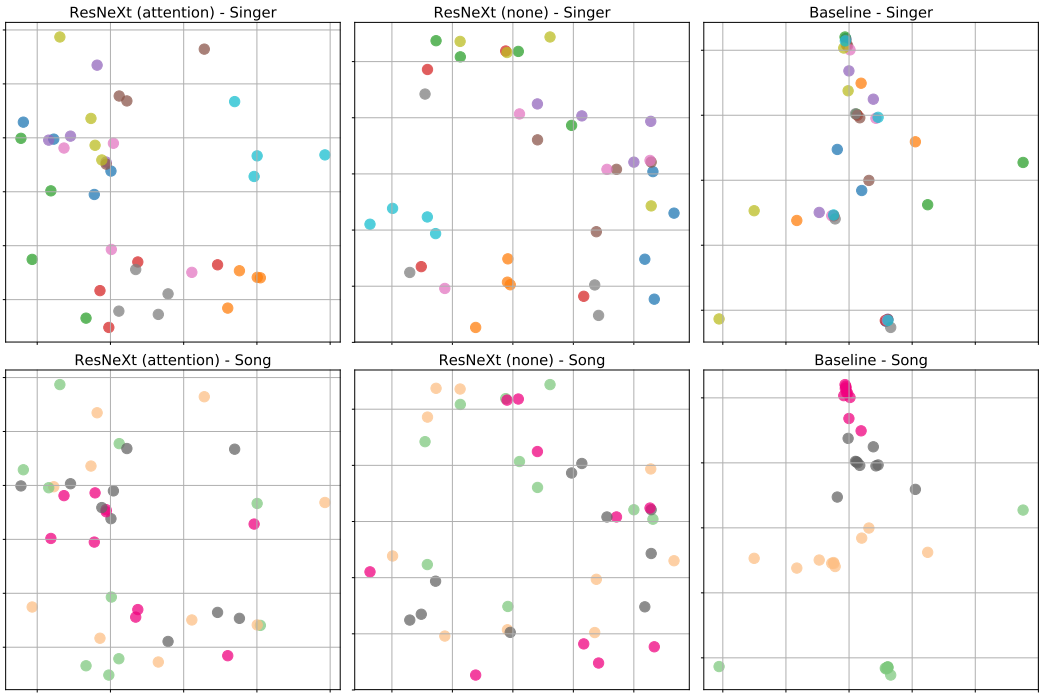

Figure 4: t-SNE projections of the embedded performances compared to baseline handcrafted audio features. The top row is colored by singer identities while the bottom is colored by song identities.

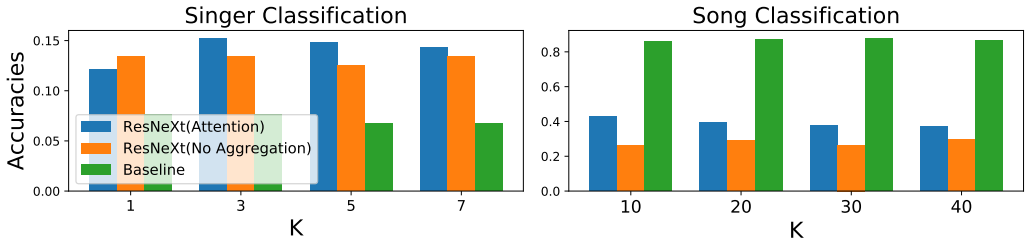

Figure 5: Bar plots of classification accuracies using k-nearest neighbor classification, on the embeddings learned from the singing performance embedding experiment. ResNeXt configurations with/without feed-forward attention and the handcrafted features (baseline) are and four k values are experimented. Left bar plot is for singer classification, while the right plot is for performed song classification.

forward attention from observing the experiment results is that it accelerates the learning process compared to other non-learnable global aggregations. One way to explain such observation is that the feed-forward attention layer learns a "frequency template" for each convolutional channel fed into it. These "frequency templates" are encoded in w and enable each convolutional channel fed into it to focus on different parts along the frequency axis (Since $w \in \mathcal{R}^D$ with $D =$ num of channels $\times$ num of frequency bins). In this paper we also have shown that training a deep neural networks having more than 15 convolutional layers on time-frequency input is definitely feasible with the help of global time aggregation. To the authors' knowledge, there is no previous music information retrieval research utilizing neural networks having more than 10 convolutional layers. A dataset consisting of over 20000 single singing voice recordings is also released and described in this paper. The released dataset, DAMP-balanced, could be partitioned in a way that for singer classification, the performed songs for each singer are the same.

For future works, replacing max-pooling with striding in convolutional layers which recent works in CNN suggest will be experimented. To improve global-aggregation, taking temporal order into consideration during the global-aggregation operation as suggested in (Vaswani et al., 2017) will also be experimented. Also the proposed neural network configurations will be experimented in other music information retrieval tasks such as music structure segmentation.

ACKNOWLEDGMENTS

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Audio chord recognition with recurrent neural networks. In *ISMIR*, pp. 335–340, 2013.

Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 2392–2396. IEEE, 2017.

Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 539–546. IEEE, 2005.

Florian Eyben, Sebastian Böck, Björn W Schuller, and Alex Graves. Universal onset detection with bidirectional long short-term memory neural networks. In *Ismir*, pp. 589–594, 2010.

Thomas Grill and Jan Schlüter. Music boundary detection using neural networks on combined features and two-level annotations. In *ISMIR*, pp. 531–537, 2015.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pp. 1735–1742. IEEE, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Annamaria Mesaros, Tuomas Virtanen, and Anssi Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *ISMIR*, pp. 375–378, 2007.

Colin Raffel and Daniel PW Ellis. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*, 2015.

Colin Raffel and Daniel PW Ellis. Pruning subsequence search with attention-based embedding. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 554–558. IEEE, 2016.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

Jeffrey C Smith. Correlation analyses of encoded music performance. 2013.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Li Su, Chin-Chia Michael Yeh, Jen-Yu Liu, Ju-Chiang Wang, and Yi-Hsuan Yang. A systematic evaluation of the bag-of-frames representation for music information retrieval. *IEEE Transactions on Multimedia*, 16(5):1188–1200, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. URL `https://arxiv.org/pdf/1706.03762.pdf`.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.

A THE DAMP-BALANCED DATASET SONG LIST

The DAMP-balanced dataset is a separate dataset from the original DAMP, but comes with the same format of metadata as the original DAMP dataset does. The audio recordings and metadata of the DAMP-balanced dataset are collected by querying the internal database of the Sing! Karaoke app hosted by Smule, Inc., which is the same as the original DAMP dataset.

The difference between the DAMP and the DAMP-balanced datasets lies at how the querying is done to collect the audio recordings and the metadata. For the original DAMP, 10 singing performances from 3462 Sing! Karaoke app users are randomly selected. There are no specific constraints on the collections of songs performed by each user. As a result, each user sang different collections of songs from each other and one song could be sang multiple times by one user. On the contrary, the queries to retrieve audio recordings and metadata for the DAMP-balanced dataset specifically ask for a group of users that all sang one specific collections of songs at least once, with only one performance returned for each song and each user, per query. 14 popular songs (defined as the more times being sang the more popular) over the past year and are listed in Table 3. For the first 10 songs, 210×2 queries were created to retrieve audio recordings and metadata that cover all different combinations of splitting the 10 songs into $6/4$ song collections. Each query returns a set of users, along with their singing performances and metadata, such that all users in that returned set has only one performance of each of the songs in the specific 6 or 4 song collection. For example, the train/validation sets used in this paper was the first 6 songs in Table 3 as training set and the following 4 songs as validation set. This specific split has 276 performances for training and 88 performances for validation, and that lead to 46 and 22 singers respectively. Different $6/4$ split results in different number of singers in each set thus making the total number performances of different songs differ from each other. For example, if instead the first 4 songs and the following 6 songs are taken as the $6/4$ split of the first 10 songs, the first 4-song collection will have 459 users and 1836 performances while the following 6-song collection having 3 users and 18 performances. The "balanced" structure of the DAMP-balanced allows possible train/validation rotation within the first 10 songs while leaving the last 4 songs as test set, or provides more possible "balanced" test sets for models training on other datasets.

B SUPPLEMENT PLOTS

Table 3: DAMP-balanced song list

Index	Song Name	Artist	# of Performances
1	one call away	Charlie Puth	3912
2	say you won't let go	James Arthur	3255
3	all of me	John Legend	2856
4	closer	The Chainsmokers	2873
5	seven years	Lukas Graham	2942
6	despacito	Luis Fonsi	1287
7	more than words	Extreme	586
8	lost boy	Ruth B.	1183
9	love yourself	Justin Bieber	3019
10	rockabye	Clean Bandit	2737
11	part of your world	Jodi Benson	56
12	when I was your man	Bruno Mars	56
13	chandelier	Sia	56
14	cups	Anna Kendrick	56

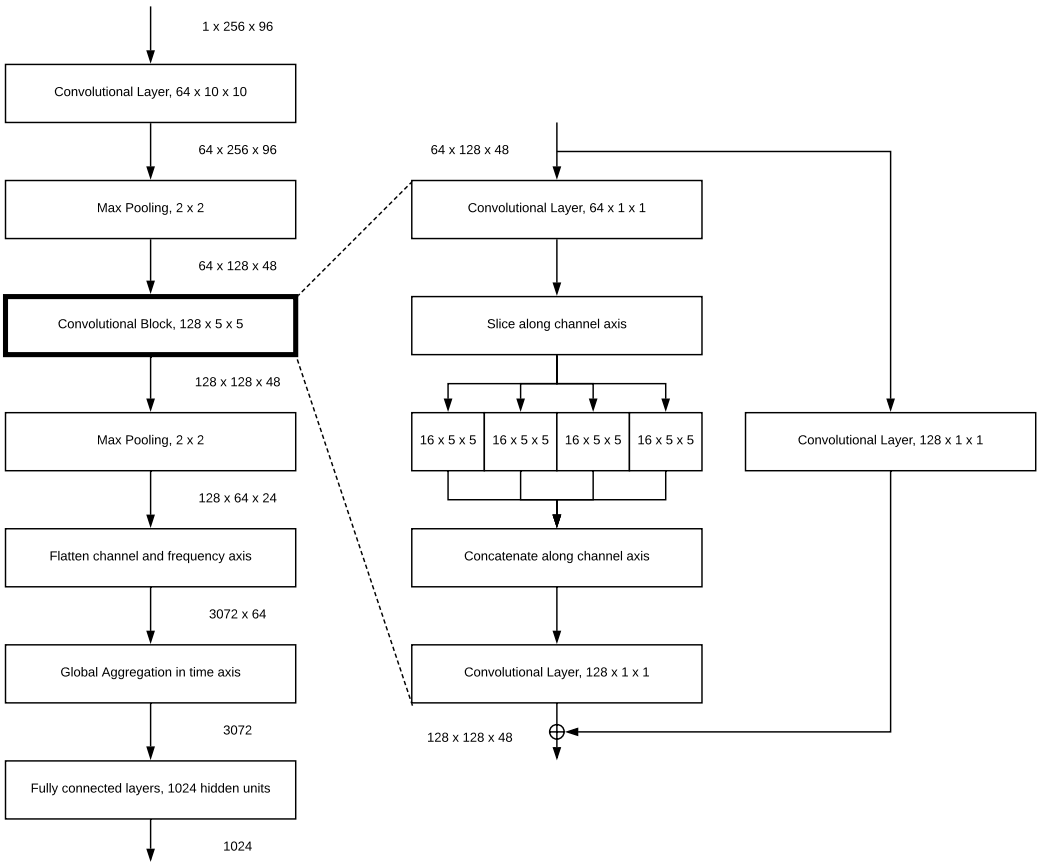

Figure 6: A detailed specification of one of the network configuration used (ResNeXt shallow for singer classification). Numbers next to arrows are the dimensions of the feature maps, while the descriptions in the boxes are operation specifics. Left plot is a flowchart from input to just before output layer, while the right plot is the detailed specifics for the RexNeXt configuration.

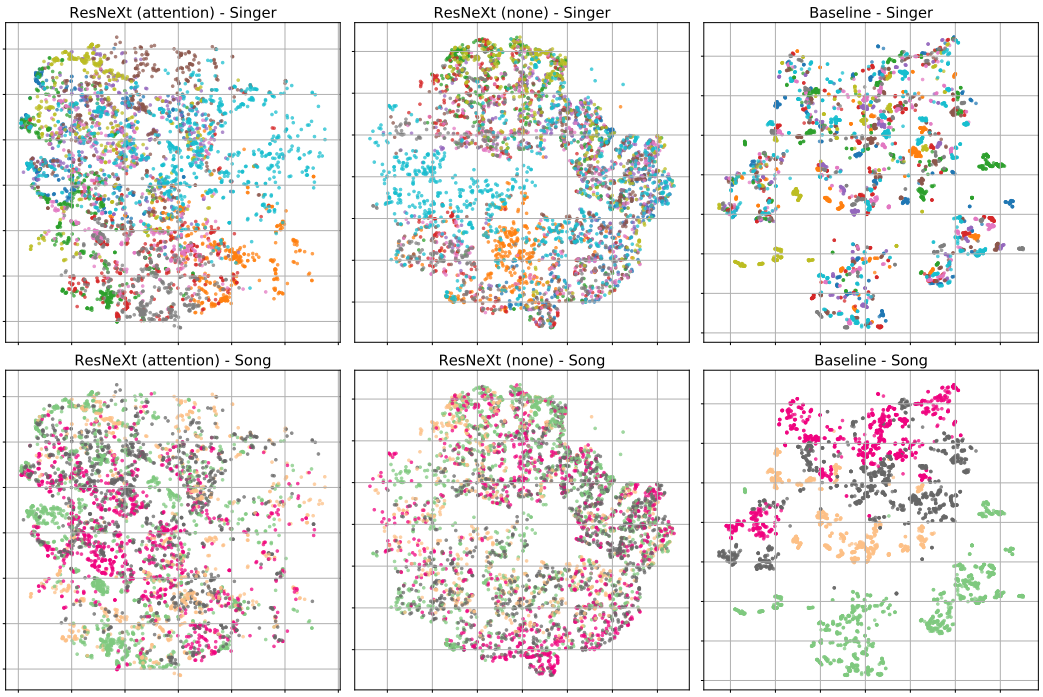

Figure 7: t-SNE projections of the embedded performance clips compared to baseline handcrafted audio features. The top row is colored by singer identities while the bottom is colored by song identities.