Acceptance Letter

2nd Annual International Conference on Artificial Intelligence and Applications
[ICAIA2022]

December 16-18, 2022, Wuhan, China
http://www.icaia2022.com

Review Result: Accept

Paper ID: ICAIA2022-N1114

Paper title: Singing Voice Detection Based on the Feature Attention Deep Neural Network

Dear Wenming Gui, Zeyu Xia, Rubin Gong and Gui Wang,

Thank you for submitting the above paper to 2nd Annual International Conference on Artificial Intelligence and Applications [ICAIA2022]. It will be hosted in Wuhan, Hubei, China on December 16-18, 2022.

The review process was extremely selective and many good papers could not be included in the final program. The review process has not finished yet, but your paper has been included in the first batch of accepted papers in the ICAIA2022 conference proceedings! Congratulations! We appreciate if you could send the final version of the manuscript at your earliest convenience, to ensure a timely publication of the paper. When submitting the final version, please highlight any changes or amendments made to the manuscript.

PUBLICATION INFORMATION:

The ICAIA2022 conference proceedings will be published by Springer's Lecture Notes in Electrical Engineering (LNEE, E-ISSN: 1876-1119; P-ISSN: 1876-1100). Springer's Lecture Notes in Electrical Engineering Series will submit the proceedings to **EI, CPCI, Scopus, Inspec** for indexing.

Thank you for your contribution to the ICAIA2022 and we are looking forward to your future participation in Wuhan, Hubei, China on December 16-18, 2022.

Yours sincerely,

ICAIA2022 Organizer

September 12, 2022

HONGKONG VICO INFORMATION AND INDUSTRY RESEARCH CENTER LIMITED
VIIRC

Singing Voice Detection Based on the Feature Attention Deep Neural Network

Wenming Gui[1][0000-0002-1230-1631], Zeyu Xia[2][0000-0003-0234-5857], Rubin Gong[1], Gui Wang[1]

[1] Jinling Institute of Technology, Nanjing 211169, Jiangsu, China
guiwenming@126.com, {gongrubin, wanggui}@jit.edu.cn
[2] Queensland University of Technology, Brisbane City QLD 4000, Australia
zeyu.xia@connect.qut.edu.au

Abstract. Singing voice detection is an important and challenging task in the field of music information retrieval. In this paper, we propose new algorithms based on the feature attention deep neural network. The attention mechanism aims at learning the importance of features and then adjusting the feature weights. We present two attention mechanisms combined with a deep neural network, which is fed by only simple features, so that the task can be achieved without any complicated feature engineering. One of the attention mechanisms is made up of the squeeze-and-excitation block, and the other mechanism is made up of the scaled dot-product attention. To make the convolutional neural network deeper, we designed residual block. The experiments are based on the public datasets: Jamendo, Mir1k, RWC pop, and their combined dataset. The results show that our networks produce better performance than the baseline. Moreover, we demonstrate that both the attention mechanism and making the convolutional network deeper are effective measures for the task. Compared with the shallower convolutional network, our feature attention deep neural network provides flexibility to choose proper depth to balance between training cost and performance.

Keywords: Singing voice detection, convolutional neural network, squeeze-and-excitation network, scaled dot-product, residual convolutional network

1 Introduction

Singing Voice Detection (SVD) is the task to discriminate whether an audio segment contains a singer's voice. Fig. 1 illustrates that the singing voices of a music clip are annotated through SVD, where the yellow segments indicate singing voices. Although it is very easy to identify vocal from songs for humans, it is still difficult for a machine to do that from their waveforms. As we all know, there are a huge number of variations of singer's voice as well as instruments. Especially, some instruments mimic the mechanism of human pronunciation, and they share the same frequency bands and the timber characteristics with the human voice, which makes the SVD task much tougher. In the field of music information retrieval (MIR), SVD is fundamental and crucial for various tasks, such as singing voice separation [1], singer identification [2] and lyrics alignment [3].

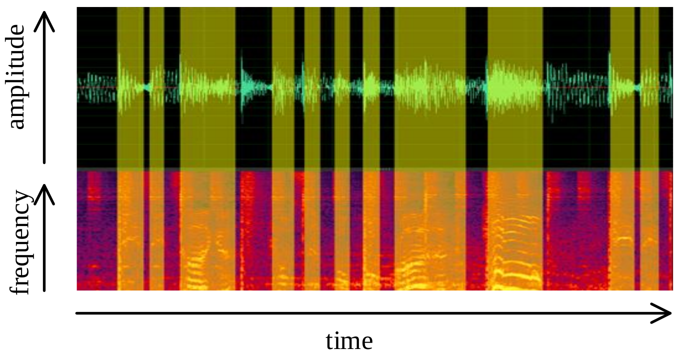


Fig. 1. An illustration of singing voice detection.
The upper image is the waveform of a music clip, and the lower is the spectrogram

In general, the task is addressed via two key steps: feature extraction and classification. Earlier features were directly inspired by Voice Detection (VD), using methods such as Linear Prediction Coefficients (LPC) from [4], Perceptual LPC (PLPC) from [5], and Mel-Frequency Cepstral Coefficients (MFCCs) from [6]. Recent features have been designed to capture specific musical characteristics, such as Fluctogram, Spectral Flatness, and Spectral Contraction (SC) from [7, 8]. For classification, several classifiers have been employed. The earlier ones are Hidden Markov Models (HMM) from [4, 9], Gaussian Mixture Models (GMM) from [6], Support Vector Machines from [10], and Random Forests from [8], to the modern Deep Neural Networks (DNN), such as Convolution Neural Networks (CNN) from [7, 11-13], and Recurrent Neural Networks (RNN) from [14, 15].

Obviously, if we can find a suitable feature to discriminate voice segments from audio, it would be very easy to solve the problem. However, it has been very difficult to design such a feature so far. Many instruments share frequency and timbre characteristics with the human voice (such as pitch range, harmonic spectra, and vibrato), making it very difficult to distinguish them from voice. As we cannot get ideal results using only single feature, it is natural to try to concatenate two or more features [7, 8]. As another way, we can try to overcome the feature problem by using more sophisticated classifiers such as Random Forests in [8], and DNN based algorithms mentioned above.

Besides acting as the classifiers, DNN based algorithms can learn different levels of features in the different layers of neurons, i.e., DNNs can be utilized as a feature extractor at the same time [16]. This not only can avoid the need to deliberately design features, but also can simplify the detection framework. In this paper, we use only one of the simple features as the DNN input, the Logarithmic and Mel-scaled Spectrum (LMS), which has been generally used as the basis for further feature engineering [7, 8, 11-13]. We focus on learning more complex features from this simple feature through DNNs.

Since we concentrate on learning features rather than designing features, we must face two questions. The first question is which features are important for the classification, because DNN learns various features at different layers and those features naturally would play different roles in achieving the result. We need to measure the importance of the learned features. For this question, we propose a feature attention mechanism to learn the importance, and then imposed this importance, i.e., the weights, on those features, and finally fed the weighted features into the next layer. Regarding the attention, a range of mechanisms can be considered, such as additive attention in [17], channel attention in [18], dot-product attention in [19], and scaled dot-product attention in [20]. As far as we know, there is no paper in the literature to address the feature importance problem in DNN based architecture in the SVD field.

The second question concerns how many features we need, which is mainly related to the depth of DNN, because we usually extend the network by depth rather than width. As we believe that a deeper network could learn more useful features than a shallower one, we propose the deeper CNN (DCNN) in this paper, whose depth could be much bigger than that of shallower CNNs (SCNNs) used in recent work [7, 11-13]. To implement the DCNN, we employed the Resnet [21]. It is worthwhile to note that a DCNN is not built by simply increasing the depth of a SCNN due to the well-known problems of vanishing and exploding gradients, which limit the CNN's learning capabilities, and can even reduce it with the network depth increasing. As for the RNN, for example LSTM, the recurrent architecture can capture long-term dependencies in the time dimension. If needed, we can still increase its depth by adding layers [15]. However, to train a deeper RNN with fully connected layers, we would pay higher computation cost and make training more difficult. Moreover, the deeper RNN has little advantage for SVD task because the adjacent frames have little effect on the current frame to predict whether it contains singing voice.

2 Methods

In this paper, we investigate the SVD problem solely using the simple LMS feature, instead of a complicated feature with an engineering procedure. We employ DCNN to generate different features at the different layers, and measure the importance of the features with an attention mechanism. For the network structure, we propose two variations, named squeeze-and-excitation residual neural network (SE-Resnet) and scale dot-product residual neural network (SDP-Resnet).

2.1 Attention Mechanism

Attention Background. The first work associated the attention mechanism with the Neural Network (NN) is done by Graves [22]. He proposed the Neural Turing Machine (NTM) based on the RNN, in which the content-based addressing mechanism was employed to impose “blurry” read and write operations on the “working memory”. Let’s take the read operation as an example. The operation at time t can be simply formulated as the following equation:

$$v = softmax\big(g(k, M)\big)M \tag{1}$$

In Eq. (1), $M = (m_1, m_2, \dots m_n)$ is the row vector of the memory contents, and k is the key vector produced by the read head, and g is a cosine function measuring similarity between the key vector and the content, i.e.:

$$g(u, v) = \frac{u \cdot v}{||u|| \cdot ||v||} \tag{2}$$

Other than the traditional read operation, in which the address and the returned value are fixed at time t, the address is blurry, depending on the distribution on all the memories, and the returned value depends on all the memory contents and their corresponding weights. This addressing mechanism essentially was an attentional process in which the weight refers to the degree of attention and the cosine similarity was used for measuring the degree.

Lots of attention mechanisms have been emerging since Graves' NTM turned up, among which we need to firstly mention the inspiring attention network related to Neural Machine Translation (NMT). Based on the RNN Encoder–Decoder framework, Bahdanau et al. proposed an additive attention model to score how well the inputs around position j and the output at position i match, and then the annotations were computed by these scores [17]. Finally, the context vector for each target word was obtained by the weighed sum of these annotations. Luong et al. further proposed a serious of attention mechanisms to address the NMT problem, categorized as the global attention and the local attention [19]. The global one put the attention to all the source words as Bahdanau et al. did, but with some different aspects, while the local one solely focusing on the small window of inputs and being differentiable. Vaswani et al. proposed the other famous attention mechanism named self-attention which we will utilize in this paper [20]. Just as the word "self" implies, the attention creates a mutual relationship among the source words, different from the relationship of the other mechanism described as between the target words and the source words. In this mechanism, the scaled dot-product (SDP) was utilized to measure the degree of the relationship. To learn more relative information, they further offered a multi head model featured different representation subspaces.

The attention mechanisms mentioned above were designed for the RNN. As another important type of NN, CNN plays a great role in audio and video processing. Can we put the attention to CNN and improve its performance? Hu et al. proposed a squeeze-and-excitation block to recalibrate the relations among the channels, essentially a self-attention mechanism, and greatly succeeded on the recent image classification task [18]. Some other works tried to integrate channel and spatial attention to solve the image processing problems [23, 24]. Unlike RNN, CNN has distinct feature maps hierarchy, and there are different feature maps at different layers which are generated through different convolutional kernels. In this paper, we will study the channel relationship of CNN and will try to improve the SVD performance.

Feature Attention. If we go through all the attention mechanisms above, we can draw a conclusion that these attention mechanisms are to obtain the distribution on which we will put our attention, while the distribution can be measured by the weight. For CNNs, feature maps represent the features extracted through the neurons. To get the attention distribution on the feature maps, we need to obtain the channel-wised weights. Meanwhile, the importance of the features in CNNs can be equal to the channel-wised weight.

In Fig. 2, we illustrate our feature attention mechanism for CNN. It is supposed that there are two adjacent layers $n-1$ and layer n in a CNN, and the layer $n-1$ is upward to the input, and $F=(F_1, F_2 \dots F_n)$ are the output feature maps at layer $n-1$. In this paper, we try to learn the weight w_i as the importance of the feature F_i by the attention block, and then feed the scaled features $F' = WF$, where $W = (w_1, w_2, \dots w_m)^{\mathrm{T}}$, to the next layer n. After imposed the weights of importance, the feature maps at layer $n-1$ are transformed more reasonably for the SVD task. We hope that this mechanism can improve the overall performance.

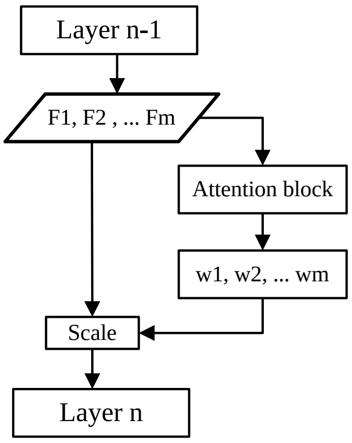


Fig. 2. Feature attention mechanism

Since the attention block needs to measure the importance among features at the same layer, it is a self-attention mechanism. In this paper, to implement this kind of attention block, we studied two methods, one was based on the squeeze-and-excitation from [18] and the other was based on the scaled dot-product from [20].

Squeeze-and-Excitation Block. The squeeze-and-excitation block (SE-Block) originates from the squeeze-and-excitation network (SENet) in the image classification field [18]. Other than studying the spatial structure of the CNN, the SENet strengthened its learning capabilities by making use of the relationship among channels. The SE-Block is the core structure of the SENet, shown in the Fig. 3.

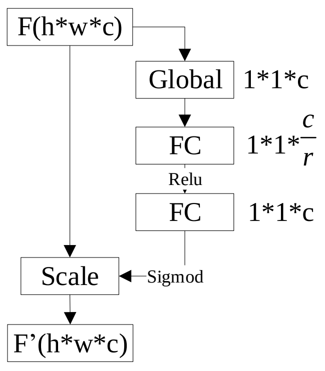


Fig. 3. Squeeze-and-excitation block

For the input F with c channels and each channel with a size h*w of feature map, where h is height and w is width, at the first step, the squeeze-and-excitation block squeezes each channel, i.e., feature map, into a channel descriptor using global pooling. At the second step, a gating mechanism is carried out on the descriptor aiming at limitation and generalization of the model, which comprises three units, a fully connected (FC) layer for reducing dimensionality, a Rectified Linear Unit (ReLU), and a fully connected layer for increasing dimension. At the third step, a sigmoid function is employed to extract the channel-wised weights, i.e., dependency. Finally, F is rescaled through the weights.

SE-Block is essentially a self-attention block, in which the feature importance is obtained by the squeeze-and-excitation operation. The rescaled F' is adjusted in line with the importance of the feature map. Therefore, we can use the SE-Block as our attention mechanism.

Scaled Dot-Product Block. The scaled dot-product (SDP) originated from the field of NTM from [20], which is another self-attention mechanism. It maps a query and a key-value pair to an attention value. We can formulate it in matrix form:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d}_k})V \tag{3}$$

In Eq. (3), Q, K, and V represent the matrix form of query, key, and value respectively and d_k is the key dimension. Like the content-based addressing mechanism in [22], SDP calculates the similarity measure first and then transforms this measure to the weight matrix by softmax function. Finally, it produces an attentional value by the weighted V. Compared with the content-based addressing, the differences of SDP lie in the facts that (shown in the Fig.4, where the two mechanisms have the same structure.), on the one hand, the measure function is the scaled dot-product, not cosine, on the other hand, the variables of the measure function are not the "memory content" and the "key" , but the query Q and the key K, in which K is similar to "memory content" if we treat K as the abstract of V, and Q is corresponding to the "key" produced by the read head.

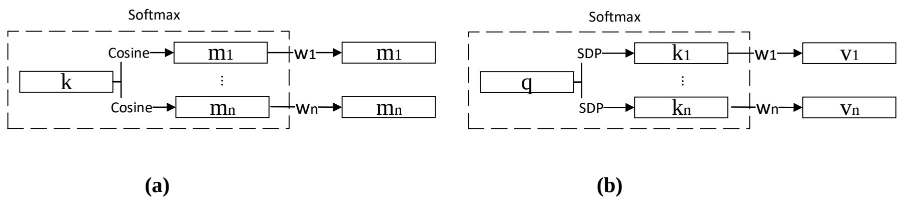


Fig. 4. The relation of attention mechanism between (a) content-based addressing, where $M = (m_1, m_2, \dots m_n)$ are row vectors of memory content, k is a vector produced by the read head; and (b) SDP, where q is one of the query vectors of Q, and <$K = (k_1, k_2, \dots\ k_n)$, $V = (v_1, v_2, \dots\ v_n)$> is a key-value pair

In addition, the value to be weighted is not the same as "memory content", but the value in the key-value pair, in which we think it brings much more flexibility for SDP mechanism as we could create different key-value pairs representing "memory content", not limited to itself, though we could set $K = V$ so that it degenerates to the content-based addressing.

To obtain the attentional value, we designed three linear units (Fig. 5) to map the feature maps $F(h * w * c)$ to Q, K, and V, and they have the same size $d * c$, where each image with the size $h * w$ is flattened and transformed to a vector with the length d. After the SDP attention function is imposed, another linear unit is employed to map the attentional value size to the original size. The linear units are essentially fully-connected layers, but without non-linear function such as ReLU. The first three linear units act as reducing dimensionality, while the fourth linear unit acts as increasing dimensionality, where the length d is usually bigger than that of SE-Block which is only 1, and we set $d = w$ in the experiments. This may lead to bigger number of parameters and more computation cost. The output value F' can be treated as the features embedded the importance, and then they can be fed into the next layer. Other than SE-Block, F' is already a scaled attention value, and it is not achieved by multiplying the weight with the original feature maps. All the weights of the linear units above can be learned through the network. In addition, we imposed multi head attention, in which the length d was multiplied by the head number N to increase the number of representation subspaces.

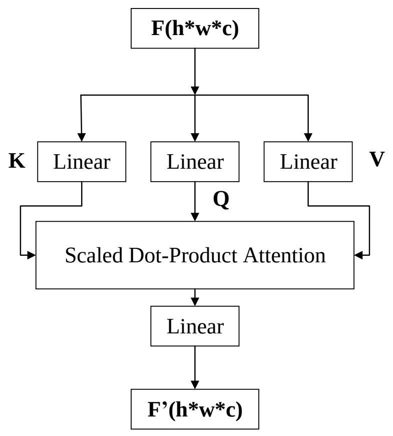


Fig. 5. SDP block

2.2 Deeper Convolution Neural Network

In general, how complex a DNN model suits a problem depends on the problem's own complexity. However, in most cases, we have no way out to measure the problem's complexity. Therefore, we must try to create various experimental models to suit the problem and finally choose the best one. In the literature, works from [7, 11-13] have tried to explore SCNNs to solve the SVD problem, but we have not seen any works related to DCNN so far. In this paper, we investigated DCNN and jointly embedded the attention mechanism for this problem. For DNN models the more complex they are, the better learning capabilities they have. While the complexity of the CNN model is contingent on its depth and width, increasing the depth could enhance its learning capability as well as expanding the width. But usually, we would like to increase the depth rather than to expand the width because increasing the depth essentially is to make the non-linear operation nest, while expanding the width is to add the non-linear operation at the same layer. Regarding increasing the depth of CNN, some works has been successfully done, such as VGG from [25] and Inception series from [26]. Especially Residual Neural Networks (Resnets) from [21], with identity-based skip connections, solved the problems of vanishing and exploding gradients to a significant extent. Resnet leads to easier optimization of the network without more parameters and computation complexity. Moreover, Resnet can be integrated with the other DNNs such as VGG and Inception mentioned above to improve their performances. In this paper, we did not explore the rich varieties of DNNs to integrate, just employed the Resnets proposed in [21] for DCNN as we solely focused on the "deep" as well as the "attention", where the numbers of layers in the Resnets included 18, 34, 50, 101, and 152. The residual functions employed were based on the basic building block and the bottleneck building block, shown in the Fig. 6. The networks with the depths 18 and 34 were piled up with the basic block, while the bottleneck block was for the depths 50, 101, and 152. We will describe the details further in the implementation section.

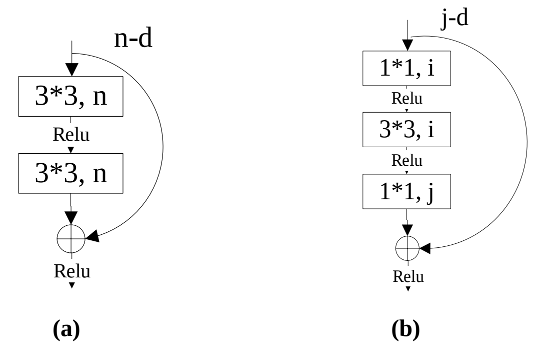


Fig. 6. The building block of the Resnets: (a) The basic block with a stack of two $3 * 3$ convolutional layers; (b) The bottleneck block with a stack of three convolutional layers, where the two $1 * 1$ convolutional layers are responsible for reducing and increasing dimension respectively

2.3 Squeeze-and-Excitation Residual Neural Network

If we use SE-Block as the attention mechanism and treat Resnet as the DCNN for our task, these two components can be integrated into the SE-Resnet, which we dub as SER-Block. There are two categories, one of which we dub as SERB-Block with the Resnet basic building block (Fig. 7a), one of which we dub as SERBT-Block with the Resnet bottleneck building block (Fig. 7b). Both inputs of the SE-Block are the outputs of the Resnet building block, and their dimension are the same as the dimension of the original feature maps F. The dimension of the final output feature maps F' are still the same as F though intermediate changes happen in the SE-Blocks and the bottleneck block of Resnet.

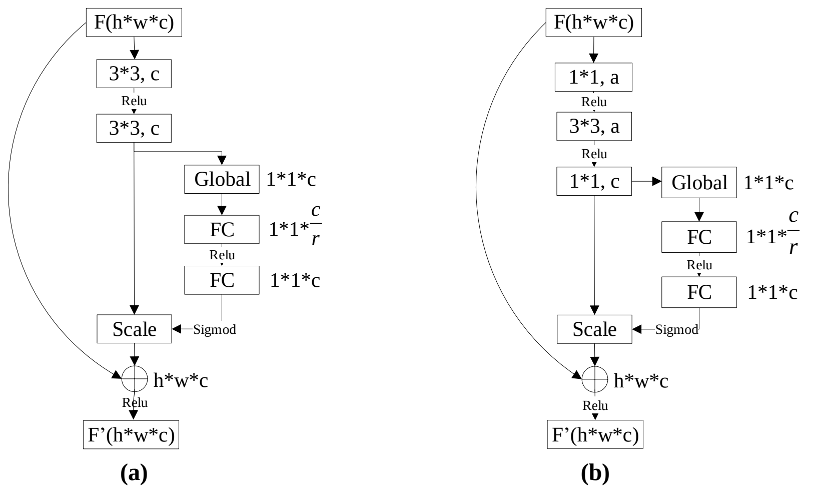


Fig. 7. SER-Block structure: **(a)** SERB-Block, integrated with the Resnet basic building block and SE-Block;
(b) SERBT-Block, integrated with the Resnet bottleneck block and the SE-Block

2.4 Scaled Dot-Product Residual Neural Network

If we use the scaled dot-product (SDP) as the attention mechanism and integrate it into Resnet, we can create SDP-Resnet structure for the SVD task, which we dub as SDPR-Block. Fig. 8a and Fig. 8b illustrate the two categories, one of which we dub as SDPRB-Block with the Resnet basic building block, one of which we dub as SDPRBT-Block with the Resnet bottleneck building block. Like the SER-Block, the dimensions of *F* and *F'* are the same.

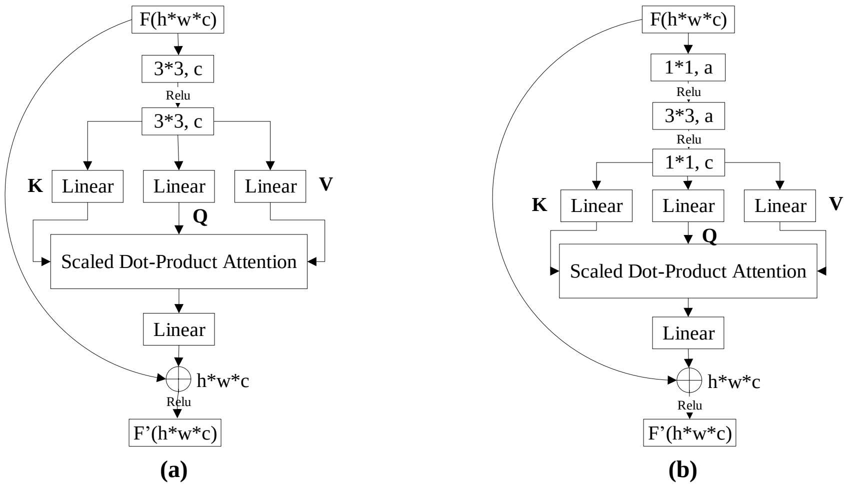


Fig. 8. SDPR-Block structure: (a) SDPRB-Block, integrated with the Resnet basic building block and SDP-Block;
(b) SDPRBT-Block, integrated with the Resnet bottleneck block and the SDP-Block

3 Implementation

In this section, we will describe the details about the implementation, including how we dealt with the input feature, the attention mechanism, the Resnets setup, and the training configuration.

3.1 Input Feature

As discussed, we computed a simple feature, LMS, as the input. Following a standard way, the original music signal was resampled at 22050Hz, and then the spectrogram was calculated. After applying the Mel-scale to the spectrogram with 80 frequency bands, we made the magnitude logarithmic. We cut off frequency bands outside the range 27.5 Hz to 8kHz. The frame length was 1024, and the hop size was 315, i.e., 14.3ms. We segmented the LMS feature into $80 * 115$ images, and fed them into our networks, with the hop size 5. So, the total hop size was 71.5ms, and every image length was 1.6s. It is worth noting that the label paired with the image was defined as the voice presence at the image center, neither in the whole image nor in a segment of the image, which meant the detection precision would be 71.5ms though the duration of the image was 1.6s. In fact, our precision was contingent on the hop size of LMS segmentation, therefore, if we limited the hop size to 1, which is equal to the frame length of STFT in terms of duration, the precision would be 14.3ms. We did not do that because a huge amount of the training time would be added. The detection precision we provided did not include in the first 1.6s of every song because we did not impose padding strategy on the LMS segmentation stage. However, it did not affect the overall performances evaluation. Finally, we made a classification decision for every image, i.e., for the image center.

For every dataset including training dataset, validation dataset, and test dataset, we constructed an image library and selected a batch of images to feed to the networks. The batch size was 64. The strategy of selecting images from the datasets was different. We imposed the random strategy for the training dataset, while kept the ordered strategy for the validation dataset and the test dataset.

3.2 The Resnets and the Attention Mechanism

To investigate the impact of the different depths on the results, we designed the Resnets with depths of 18, 34, 50, 101, 152, which are typical depths used for the Resnet.[21] In addition, we added the depth 200 to test whether a deeper network could increase performance. The Resnets with the depths 18 and 34 were built with the basic blocks, while the bottleneck blocks were used for the depths 50, 101, 152 and 200. We show the building blocks and structures of the Resnets in the Fig. 9. We kept the original structure for the typical depths unchanged, in which the scale parameters for the depths 18, 34, 50, 101, and 152 are $[2, 2, 2, 2]$, $[3, 4, 6, 3]$, $[3, 4, 6, 3]$, $[3, 4, 23, 3]$, and $[3, 8, 36, 3]$ respectively, while the scale parameters for the depth 200 is $[3, 12, 48, 3]$. In fact, the scale parameters for the Resnet would affect the performance, for example, we found that the scale parameter $[3, 8, 16, 23]$ for the 152-layer Resnet produced a better result for the SVD problem. However, we would not like to further investigate this structure, instead we employed the typical structures and did not change the structures throughout the experiments for a fair comparison. For the Resnets built on the basic block, the channel dimensions of the layers conv1, conv2_X, conv3_X, conv4_X, and conv5_X, i.e., the numbers of the feature maps, also the value of n in Fig. 6a, are 64, 64, 128, 256, 512 respectively. While for the Resnets on the bottleneck block in Fig. 6b, the values $[j, i]$ of the layers conv2_X, conv3_X, conv4_X, and conv5_X are $[64, 256]$, $[128, 512]$, $[256, 1024]$, $[512, 2048]$ respectively. The number of feature maps between the adjacent layer (except conv1) of the Resnets built on the basic block, i.e., depths 18 and 34, increases by half, while the number decreases by half for those build on the bottleneck block, i.e., depths 50, 101, 152, and 200.

layer name	18-layer	34-layer	50-layer	101-layer	152-layer	200-layer
conv1	7×7, stride 2					
conv2_X	3×3 max pool, stride 2					
	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_X	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 12$
conv4_X	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 48$
conv5_X	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	average pool, 2-d fc					

Fig. 9. The Resnets structures. The basic building block makes up the 18-layer and 34-layer Resnets, while the bottleneck building block makes up the 50-layer, 101-layer, 152-layer, and 200-layer Resnets. The scale parameter near the bracket indicates the number of the building blocks for stacking the Resnet

Since the input image size is $80 * 115$, the output image sizes of the layers conv1, conv2_X, conv3_X, conv4_X, and conv5_X are $40 * 58$, $20 * 29$, $10 * 15$, $5 * 8$, and $3 * 4$ respectively. In addition, it is worth noting that the output of the Resnets is 2-dimension vector with the length equal to the batch size, in which if the value of the first dimension is bigger than the value of the second dimension, we classify the input image as the voice type, otherwise the non-voice type.

Regarding the SE-Resnet, we made them up according to Fig. 7 with the SERB-Blocks and the SERBT-Blocks, in which the reduction parameter r was set to 16. Regarding the SDP-Resnets we constructed them in line with Fig. 8 with the SDPRB-Block and the SDPRBT-Block. In addition, we used multi-head structure in SDP-Block and set the head number $N = 8$. For the three linear units on top of the SDP module in Fig. 8, we set the output length $d = N * w$, where w was the width of the input image at the current SDP layer.

3.3 Training Configuration

We implemented the network on the PyTorch platform, with the help of the Homura package [27]. The weighted binary cross entropy was utilized as the loss function. The reason why we used the weighted version is because our datasets usually contain more voice samples, being unbalanced to some extent. Adam was utilized as the optimizer with the weight decay $1e-4$. We used the early stopping mechanism as well, and the patience was set to 10. From the experiments, we usually could reach early stopping criteria in 30 epochs, although we set the maximum epoch as 50. We employed the step-wise learning rate scheduler with step size of 10. In addition, we will publish the train, validation, and test source codes at GitHub after organizing.

4 Experiments and Results

In this section, we present experimental results and compare them with the baseline.

4.1 Datasets

We chose three publicly available datasets for experiments. The first dataset is the Jamendo corpus (JMD), which contains 93 songs with 371 minutes of total length. The second is the RWC pop, which contains 100 songs with 407 minutes of total length. The third is the Mir1k which is relatively smaller and contains 133 minutes of total length and 1000 song clips with durations ranging from 4 to 13 seconds.

We kept the original division of train, validation, and test datasets for JMD unchanged, i.e., 61 songs for train, 16 for validation, and 16 for test [7, 14, 15, 28]. We divided the RWC dataset by which the songs ending with the digit 0–4 were selected for the train dataset, 5 and 6 as the validation dataset, and 7–9 as the test dataset. We separated the Mir1k datasets in the way that songs starting with a–g were chosen as the train dataset, h–k (including K) was acting as the validation dataset, and l–z was acting as the test dataset. The divisions for RWC and Mir1k were random to some extent.

We also combined all the three datasets to a whole dataset by integrating the corresponding train, validation, and test datasets respectively. Hence, we got a new dataset, which we call the dataset JRM. The ratios of voice to non-voice samples of JMD, RWC, Mir1k and JRM are 1.12, 1.55, 4.37 and 1.48 respectively, which were used for the weighted binary cross entropy loss function. Table 1 shows the summary of these datasets.

Table 1. Summary of the datasets used for experiments

Dataset	Number of songs	Total length(minutes)	Voice to non-voice ratio
JMD	93	371	1.12
RWC	100	407	1.55
Mir1k	1000	133	4.37
JRM	1193	911	1.48

4.2 Baseline

In this paper we compare the performances based on the pure models, without data augmentation and without feature engineering. We used the implementation from [28] as the baseline, which can be treated as a third-party evaluation, where a SCNN from [12] with 4 CNN layers and 3 dense layers and a Bi-LSTM from [15] with 3 layers of

(30, 20, 40) LSTM units and 1 dense layer were developed for comparison. The SCNN was only fed by the LMS feature in this implementation, just like our approach. Therefore, we directly used their results on JMD. Because they did not provide the experimental results for the other datasets, we conducted additional experiments. To obtain the results of SCNN on RWC, Mir1k, and JRM, we ran the train and test programs on the respective datasets. For Bi-LSTM, we also referred to these results on JMD as the base line despite the HPSS preprocessing, and ran the programs on RWC, Mir1k, and JRM as we did for SCNN. It is worth noting that we intentionally keep the most of the parameters in our attention-based DCNN implementation same as the baseline implementation for fair comparison, such as the frame length, the image size, and the detection precision, which we described in the implementation section.

In addition, it is worthwhile to note that SCNN from [7] and Bi-LSTM from [15] were reported to have higher accuracies (93.2 and 91.5) on JMD. We did not use these results as the baseline because we did not have the codes or the implementation details, also because it imposed data augmentation or extra preprocessing. LSTM-RNN from [14] was also reported to have an accuracy of 92.9 on RWC, but we did not treat it as the baseline due to lack of codes or implementation details, also its feature engineering solution. Moreover, they utilized 5-fold cross-validation which was different from us.

4.3 Comparison of Results

All additional parameters except depth keep unchanged throughout all experiments. We extended the network depth aiming at studying impact on the result. Other than our networks, SCNN cannot extend the depth so much and Bi-LSTM have a different type of "depth" as we mentioned before. As we proposed DCNN in this paper, we conducted experiments on the relatively deeper SE-Resnet and SDP-Resnet at the depths 50, 101,152, and 200 for comparison. Because the results are usually not the optimal, we provide the performance statistics on the datasets JMD, RWC, Mir1k, and JRM in Table 2, 3, 4, and 5 respectively, where $\mu \pm \sigma$ refers to mean and standard deviation performance, FP is false positive and FN is false negative. Especially, for convenient comparison, we made the average performances (Avg) based on Table 2, 3 and 4 in Table 6.

Table 2. Comparison of the Performances on the dataset JMD

%	Accuracy	F-measure	Precision	Recall	FP	FN
SCNN	86.80	86.30	83.70	89.10	15.10	10.90
Bi-LSTM	87.50	86.60	86.10	87.20	12.20	12.80
SE μ±σ	88.25±0.3	87.85±0.41	84.73±1.18	91.25±1.95	14.35±1.61	8.75±1.95
SDP μ±σ	87.63±1.2	86.45±1.58	87.96±2.41	85.15±4.31	10.23±2.78	14.85±4.31

Table 3. Comparison of the Performances on the dataset RWC

%	Accuracy	F-measure	Precision	Recall	FP	FN
SCNN	87.94	89.73	91.46	88.07	12.25	11.93
Bi-LSTM	87.59	89.72	89.55	89.90	15.92	10.10
SE μ±σ	91.1±0.3	92.79±0.41	93.55±1.18	92.07±1.95	10.84±1.61	7.93±1.95
SDP μ±σ	89.58±0.73	91.11±0.67	92.78±0.56	89.51±1.32	10.32±0.92	10.49±1.32

Table 4. Comparison of the Performances on the dataset Mir1k

%	Accuracy	F-measure	Precision	Recall	FP	FN
SCNN	87.61	92.87	90.64	95.22	54.92	4.78
Bi-LSTM	86.54	92.04	90.89	93.22	47.12	6.78
SE μ±σ	88.66±0.2	93.52±0.1	90.73±0.66	96.49±0.71	54.85±0.66	3.51±0.71
SDP μ±σ	87.54±0.24	92.81±0.19	89.64±0.45	96.22±0.91	56.73±3.25	3.78±0.91

Table 5. Comparison of the Performances on the dataset JRM

%	Accuracy	F-measure	Precision	Recall	FP	FN
SCNN	87.39	89.03	90.83	87.30	12.49	12.70
Bi-LSTM	87.83	89.63	88.48	90.81	27.95	14.21
SE μ±σ	90.13±0.35	91.74±0.23	90.1±0.9	93.46±0.57	14.59±1.56	6.54±0.57
SDP μ±σ	89.05±0.37	90.8±0.32	89.5±0.31	92.14±0.47	15.34±0.48	7.86±0.47

Table 6. Average performances on the datasets JMD, RWC, and Mir1k

%	Accuracy	F-measure	Precision	Recall	FP	FN
SCNN	87.45	89.63	88.60	90.80	27.42	9.20
Bi-LSTM	87.21	89.45	88.85	90.11	25.08	9.89
SE	89.34	91.38	89.67	93.27	26.68	6.73
SDP	88.25	90.12	90.13	90.29	25.76	9.71

JRM is the combination of all the other three datasets, therefore, the results based on JRM deserved more value with more comprehensive data. For the accuracies on JRM, we achieved an average 2.74 ppt gain over SCNN and 2.3 ppt over Bi-LSTM by SE-Resnet, and an average 1.66 ppt gain over SCNN and 1.22 ppt over Bi-LSTM by SDP-Resnet, though SVD is a well-studied task with limited improvement scope. For the single dataset, the highest gain of 3.51 ppt over Bi-LSTM was made on RWC by SE-Resnet; A little bit lower accuracy of 0.07 ppt than the accuracy on SCNN was found on Mir1k by SDP-Resnet, which means that not all the attention mechanism can ensure effectiveness for all the dataset. We should notice that the accuracies of SCNN were on par with that of Bi-LSTM on all the four datasets. We guess that it lies in the fact that SCNN and Bi-LSTM cannot adapt to the different dataset due to their relatively poor learning capability compared to DCNN. On the contrary, the attention networks based on DCNN can learn more features from the dataset so that they can improve the performance according to the characteristics of the data. Compared with the average accuracies on Avg, the accuracies on JRM by SE-Resnet and SDP-Resnet are both higher than counterparts, which means the accuracy would be increased by simply combining the dataset. On the contrary, the accuracy on JRM by SCNN is lower than on Avg.

Regarding F-measure which is a combination of the Precision and Recall indexes, all the values are bigger than the maximum value of corresponding SCNN and Bi-LSTM on all the four datasets, except SDP-Resnet on JMD with 0.15 ppt lower values. The highest gain of 3.07 ppt lies in SE-Resnet over Bi-LSTM on RWC. The F-measures of SE-Resnet and SDP-Resnet on JRM is bigger than their corresponding values on Avg, which indicates the combination of datasets is positive for the attention-based DCNN again. Regarding Precision and Recall indexes, we can get the similar comparison results though there are a few differences.

Regarding FP and FN, both SE-Resnet and SDP-Resnet on JRM obtained better results than those on Avg. Also, we can find both have the better FN than SCNN and Bi-LSTM on JRM, meanwhile, both improve FPs greatly over SCNN but with a slight drop over Bi-LSTM. It is worth noting that FPs on Mir1k are significantly bigger than those on the other datasets. It is because the voice to non-voice ratio of the dataset Mir1k reaches 4.37 and the dataset Mir1k is relatively small, therefore even a small number of FP samples would lead to big FP.

All in all, the performances of SE-Resnet are better than all the other algorithms including SDP-Resnet. Although SDP-Resnet did not produce exciting results, we think it is promising because the key-value pair gives us more flexible space, also we can say subspaces, to represent the feature map. We might just not have found the way. In addition, because SDP-Resnet has more parameters to train than SE-Resnet, maybe we could improve the performance with more data.

4.4 Results with Different Depths

For the different datasets, due to the amount and the diversity of the data, the different depths lead to different performances. We show the results in terms of accuracy on the four mentioned datasets with the different depths of SE-Resnet in Fig. 10, where we also show the average accuracies of the corresponding depths on the dataset JMD, RWC, and Mir1k. For investigating the depth impacts we added the experimental results for the relatively smaller depth 18 and 34.

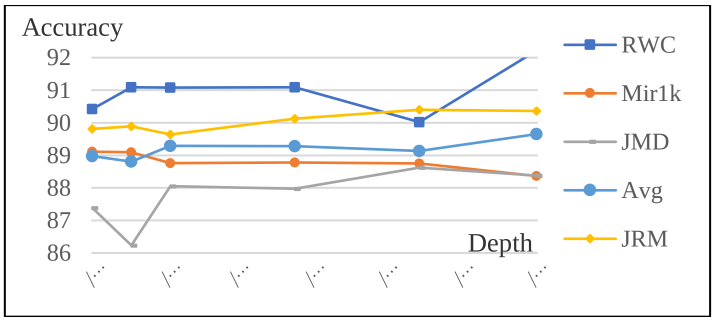


Fig. 10. The accuracies of SE-Resnets with the different depths

The accuracies of RWC and Avg reach the highest points at the depth 200, while those of JRM and JMD reach the top at the depth 152. Other than reaching the top on the relatively bigger depths, the top accuracy of Mir1k begins at the depth 18, which we think it is due to the small amount of data. If there is enough data, the deeper SE-Resnet would have more potential for better performance, however, on small dataset, the deeper network tends to overfit so that they do not obtain the best performance at the bigger depth. It is also worthwhile to note that the performances on JRM are always better than on Avg, which means the combination of the three basic datasets can develop the overall learning capability of the network. In addition, the accuracies on the more comprehensive dataset JRM have an increasing trend with the depths increasing, which means the SE-Resnet with more learned feature would be better, however, the performance also could be limited by the dataset characteristics and the model itself, therefore cannot be improved unlimitedly as we see it reaches the top at the depth 152.

Meanwhile, we should notice that there are two inharmonious points, RWC at the depth 152 and JMD at the depth 34, which we guess it is because the network possibly reached the saddle points or the patience was unsuitably set for the early stopping mechanism. We did not try to change the optimization method to resolve the problem because we would like to keep the parameters unchanged throughout the experiments for comparison. Instead, we used the mean and standard deviation statistics to leverage the inharmonious points.

Fig. 11 illustrates the accuracy curves of SDP-Resnets with the different depths. All the curves reach the top at the relatively bigger depths, with JRM and JMD at the depth 152, with RWC and Mir1k at the depth 50, and with Avg at the depth 200. It is noted that both SE-Resnet and SDP-Resnet on JRM reach the top at the depth 152. Additionally, there are two inharmonious points as well, one of which we can explain as the possible saddle point or as the consequence of a small patience for the early stopping mechanism, one of which is prominently higher and we suppose the optimizer found the possible optimal point. Just like the SE-Resnet, we made the leveraged statistics to mitigate these inharmonious impacts on the comparison as the above section described. Also, the JRM curve runs over Avg at all the depths, which is the same as that of the SE-Resnet.

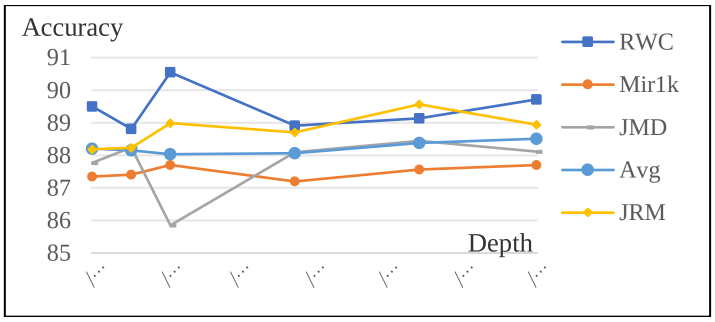


Fig. 11. The accuracies of SDP-Resnets with the different depths

From the Fig. 10 and Fig. 11, we may recommend depth 152 for the user to apply the proposed methods practically. However, if the amount of the datasets changing, we suggest that the user should conduct experiments firstly and then choose the best depth. In fact, our method can provide better performance as well as flexible depths for the user, other than the SCNN based method.

5 Discussion

In this section, we will discuss the impact of the attention mechanism and DCNN, and will disclose the computation cost in our experiments.

5.1 Attention Mechanism and DCNN Impacts

Based on the comparison between our proposed attention-based DCNN and SCNN in the section 4 (we neglect the comparison with Bi-LSTM here), we can draw a conclusion that with the integration architecture of attention mechanism and DCNN, SE-Resnet and SDP-Resnet can achieve better performance than SCNN. Although it is difficult for us to measure how much the attention mechanism and the DCNN architecture contribute to the result, we would like to try to investigate the effectiveness of attention mechanism and DCNN impacts. We conducted the experiments on dataset JRM by Resnet without the attention mechanism, in which the depths chosen are the same as Table 5. Then, we got the mean and standard deviation of accuracy. For explicit comparison, we coped and omitted the other indexes except accuracy and F-measure from Table 5, shown in the Table 7 below.

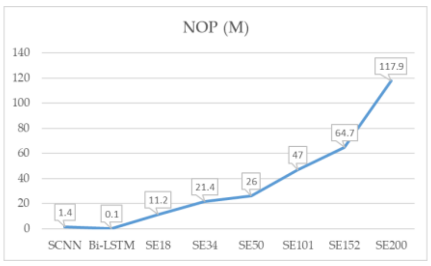
Table 7. The attention mechanism and DCNN impacts

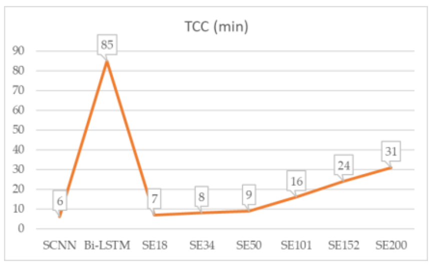
%	SE-Resnet	SDP-Resnet	Resnet	SCNN
accuracy μ±σ	90.13±0.35	89.05±0.37	88.7±0.71	87.39
F-measure μ±σ	91.74±0.23	90.8±0.32	90.59±0.59	89.03

We can see a significant relation among the accuracy means of SE-Resnet, SDP-Resnet, Resnet, and SCNN, SE-Resnet > SDP-Resnet > Resnet > SCNN, as well as the F-measure means. It proves that the attention mechanism over Resnet is effective, in which the SE-block is better than the SDP-Block, and the Resnet (DCNN) holds better performances than the SCNN. The reason why the SDP-Block is worse than SE-Block is that the SDP-Block needs more data for training due to multiple fully-connected layers with more parameters as we guess.

5.2 Computational Cost

The number of parameters (NOP) is determined by the network architecture, while the training computation cost (TCC) depends on many factors such as the network scale, the network convergence mechanism, the implementation method, the server load and so on, so that we cannot measure it precisely. We ran the different depths of the SE-Resnet on the NVIDIA Tesla V100 with 32 GB memory and evaluated TCC by minutes per training epoch on the JRM dataset. The NOPs and TCCs are shown in Fig. 12.



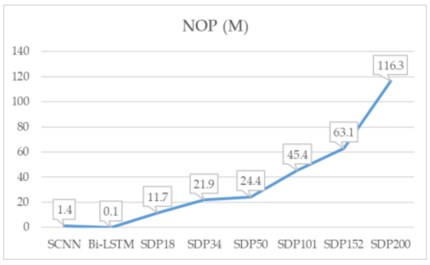


(a) **(b)**

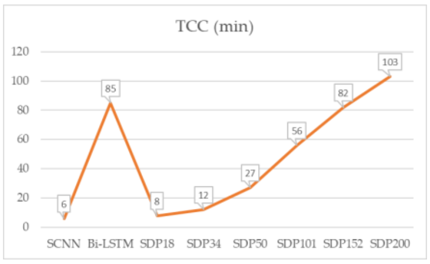
Fig. 12. The computation cost of SE-Resnets, where SE refers to SE-Resnet. **(a)** NOPs, where M refers to million; **(b)** TCC

Certainly, the bigger depth of the SE-Resnet is, the more parameters the SE-Resnet has. All the NOPs of the SE-Resnet are bigger than the SCNN and the Bi-LSTM. Regarding to the TCC, SCNN has the minimum value. Like the trend of the NOPs of SE-Resnets, the TCCs at the different depth increase slowly at the relatively smaller depths, but quickly at the bigger depths from 101. It is noticed that although the Bi-LSTM has the minimum NOP, it consumes the maximum TCC.

For the NOPs, the differences between the SE-Resnets and the SDP-Resnets lie in their corresponding building blocks. For the SE-Block, it has two fully connected layers, and it's NOP depends on the number of channels c, the reduction parameter r. If it is SE-Resnet18 and we neglect the bias parameters, the NOP of the SE-Block at the layer conv2_X would be $c * c/r * 2 = 64 * 64/16 * 2 = 512$, and the sum of NOPs of all layers conv2_X, conv3_X, conv4_X, and conv5_X would be 0.1M. This sum is the same for SE-Resnet18 and SE-Resnet34, and those sums of SE-Resnet50, SE-Resnet101, SE-Resnet152 and SE-Resnet200 are the same as well, but much bigger, i.e., 2.3M. For the SDP-Resnet, it has four fully connected layers, and it's NOP depends on the image size at the different layer, i.e., h and w, and on the head number N. If we neglect the bias parameter, the NOP of the SDP-Block at the layer conv2_X would be $h * w * w * N * 4 = 20 * 29 * 29 * 8 * 4 = 538240$ and the sum of NOPs of all layers conv2_X, conv3_X, conv4_X, and conv5_X would be 0.6M. These sums of the SDP-Resnets at different depths are the same. Furthermore, because the NOP of the Resnet is much bigger than those of the SE-Block and SDP-Block, those NOPs have little effect on the overall NOPs of the SE-Resnets and SDP-Resnets. Therefore, SE-Resnets and SDP-Resnet have small difference on the NOPs, shown in Fig. 12a and Fig. 13a.



(a)



(b)

Fig. 13. The computation cost of SDP-Resnets, where SDP refers to SDP-Resnet. **(a)** NOPs, where M refers to million; **(b)** TCC

Regarding TCCs, those of SDP-Resnets, especially at the relatively bigger depths, are much bigger than SE-Resnets. In Fig. 13b, we can see that the curve is steeper than Fig. 12b. That is because that SDP-Block consumes significantly more TCC to calculate SDP and train the four linear units than SE-Block with the depth increasing.

6 Conclusions

In this study, we have proposed the novel SVD algorithms based on the attention-based deep convolutional neural network. Through experiments, we demonstrated that both SE-Resnet and SDP-Resnet outperformed the baseline. The channel-wised attention can measure the importance of the feature maps of DCNN so that the attention-based DCNN can learn more useful features and then improve the SVD performance. Compared with SDP-Resnet, SE-Resnet presented better results and was more suitable for the SVD task, however, we think SDP-Resnet is also promising because SDP is flexible due to its key-value pair, which means we could develop different subspaces to represent the feature. In addition, we investigated the impacts of the different depths of SE-Resnet and SDP-Resnet. It showed that the best performance generally happened at the relatively bigger depth, and this trend was significant on the combined dataset JRM, with an average 2.74 ppt gain over SCNN and 2.3ppt over Bi-LSTM by SE-Resnet. We recommend to use depth 152 as it produces the best performance on JRM with more convincing and combined data.

7 Acknowledgement

This research was funded by the Major Program of Natural Science Foundation for Jiangsu Higher Education Institutions of China, grant number 22KJA520001, and by the Chinese National Natural Science Foundation, grant number 61872199, and by the Jiangsu Overseas Visiting Scholar Program for University Prominent Young & Middle-aged Teachers and Presidents, and by the Jiangsu Province Modern Education Technology Research Project, grant number 2022-R-105945, and by the High Level Project of the Educational Cooperation between Jinling Institute of Technology, Jiangsu, China and Queensland University of Technology, Brisbane, Australia.

References

1. Chan T-S, Yeh T-C, Fan Z-C, Chen H-W, Su L, Yang Y-H, et al. Vocal activity informed singing voice separation with the iKala dataset. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP): IEEE; 2015. p. 718-22.
2. Maddage NC, Xu C, Wang Y. Singer identification based on vocal and instrumental models. 17th International Conference on Pattern Recognition (ICPR): IEEE; 2004. p. 375-8.
3. Wang Y, Kan M-Y, Nwe TL, Shenoy A, Yin J. LyricAlly: automatic synchronization of acoustic musical signals and textual lyrics. 12th annual ACM international conference on Multimedia: ACM; 2004. p. 212-9.
4. Berenzweig AL, Ellis DP. Locating singing voice segments within music signals. IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics: IEEE; 2001. p. 119-22.
5. Berenzweig AL, Ellis DPW, Lawrence S. Using Voice Segments to Improve Artist Classification of Music. 22nd Audio Engineering Society International Conference2002.
6. Tsai W-H, Wang H-M. Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signals. IEEE Transactions on Audio, Speech, Language Processing. 2005;14(1):330-41.
7. Lehner B, Schlüter J, Widmer G. Online, Loudness-Invariant Vocal Detection in Mixed Music Signals. IEEE/ACM Transactions On Audio, Speech, And Language Processing. 2018;26(8):1369-80.
8. Lehner B, Widmer G, Sonnleitner R. On the reduction of false positives in singing voice detection. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Florence, Italy: IEEE; 2014. p. 7480-4.
9. Nwe TL, Shenoy A, Wang Y. Singing voice detection in popular music. 12th annual ACM international conference on Multimedia: ACM; 2004. p. 324-7.

10. Maddage NC, Xu C, Wang Y. A SVM-based Classification Approach to Musical Audio. International Society for Music Information Retrieval (ISMIR) 2003.
11. Schlüter J. Learning to Pinpoint Singing Voice from Weakly Labeled Examples. International Society for Music Information Retrieval (ISMIR). New York, USA2016. p. 44-50.
12. Schlüter J, Grill T. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. International Society for Music Information Retrieval (ISMIR). Malaga, Spain2015. p. 121-6.
13. Huang H-M, Chen W-K, Liu C-H, You SD. Singing voice detection based on convolutional neural networks. 7th International Symposium on Next Generation Electronics (ISNE). Taipei, Taiwan: IEEE; 2018. p. 1-4.
14. Lehner B, Widmer G, Böck S. A low-latency, real-time-capable singing voice detection method with LSTM recurrent neural networks. 23rd European Signal Processing Conference (EUSIPCO). Nice, France: IEEE; 2015. p. 21-5.
15. Leglaive S, Hennequin R, Badeau R. Singing voice detection with deep recurrent neural networks. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Brisbane, Australia: IEEE; 2015. p. 121-5.
16. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. European conference on computer vision. Zurich, Switzerland: Springer; 2014. p. 818-33.
17. Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014.
18. Hu J, Shen L, Sun G. Squeeze-and-excitation networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)2018. p. 7132-41.
19. Luong M-T, Pham H, Manning CD. Effective Approaches to Attention-based Neural Machine Translation. 2015.
20. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. Advances in neural information processing systems2017. p. 5998-6008.
21. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, USA2016. p. 770-8.
22. Graves A, Wayne G, Danihelka I. Neural Turing Machines. 2014.
23. Hu Y, Li J, Huang Y, Gao X. Channel-wise and Spatial Feature Modulation Network for Single Image Super-Resolution. IEEE Transactions on Circuits and Systems for Video Technology. 2019;30(11):3911-27.
24. Chen L, Zhang H, Xiao J, Nie L, Shao J, Liu W, et al. SCA-CNN: Spatial and Channel-Wise Attention in Convolutional Networks for Image Captioning. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)2017. p. 6298-306.
25. Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014.
26. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)2015. p. 1-9.
27. homura: homura package. https://github.com/moskomule/homura (2019). Accessed.

28. Lee K, Choi K, Nam J. Revisiting Singing Voice Detection: a Quantitative Review and the Future Outlook. 19th International Society of Music Information Retrieval Conference (ISMIR). Paris (France): International Society for Music Information Retrieval; 2018.