Improvement of Vocal Detection Accuracy Using Convolutional Neural Networks

Shingchern D. You*, Chien-Hung Liu, and Jia-Wei Lin
Department of Computer Science and Information Engineering,
National Taipei University of Technology
Taipei, Taiwan
[e-mail: {you,cliu}@csie.ntut.edu.tw, linitachi@gmail.com]
*Corresponding author: Shingchern D. You

Received October 3, 2020; revised November 5, 2020; accepted November 12, 2020; published February 28, 2021

Abstract

Vocal detection is one of the fundamental steps in musical information retrieval. Typically, the detection process consists of feature extraction and classification steps. Recently, neural networks are shown to outperform traditional classifiers. In this paper, we report our study on how to improve detection accuracy further by carefully choosing the parameters of the deep network model. Through experiments, we conclude that a feature-classifier model is still better than an end-to-end model. The recommended model uses a spectrogram as the input plane and the classifier is an 18-layer convolutional neural network (CNN). With this arrangement, when compared with existing literature, the proposed model improves the accuracy from 91.8% to 94.1% in Jamendo dataset. As the dataset has an accuracy of more than 90%, the improvement of 2.3% is difficult and valuable. If even higher accuracy is required, the ensemble learning may be used. The recommend setting is a majority vote with seven proposed models. Doing so, the accuracy increases by about 1.1% in Jamendo dataset.

Keywords: Vocal Detection, CNN, Ensemble Learning

A preliminary version of this paper was presented at APIC-IST 2020, and was selected as an outstanding paper. This version extends the preliminary one with more experiments and discussions. This work was supported by research grants MOST 108-2221-E-027-089 and MOST 109-2221-E-027-083 from Ministry of Science and Technology, Taiwan.

http://doi.org/10.3837/tiis.2021.02.019 **ISSN : 1976-7277**

1. Introduction

Vocal detection technique is to detect the presence of vocal sound (singing voice) in a piece of audio. It is one of the fundamental steps toward advanced applications [1]. For example, in the karaoke application, the vocal sound in the album soundtrack is intentionally removed so that the causal singers can sing only with instrumental accompaniment [2]. To accomplish this work, certainly, we need to determine the locations of singing segments in the soundtrack. Another application is the music summarization [3], where a short audio segment is used to "summarize" the entire soundtrack. For a popular song, the audio segment for summarization almost always contains vocal sound. Other applications of vocal detection techniques include melody extraction [4] and singer recognition [5], to name a few.

According to [6], there are two types of vocal detection problems. The first type is to locate the starting and ending points of vocal segments in a soundtrack. As typically there are multiple vocal segments to detect, therefore, the performance of an approach is measured based on the detection (or miss) of the points. The second type is to detect the presence of vocal sound in a short piece of audio clip. As pointed out in [6], it is not easy to say which type of problems is more difficult. In this paper, we only study the second problem, but still use the name of vocal detection.

A conventional vocal detection approach contains two steps, namely, the feature extraction step and the pattern classification step. The extracted features are usually time-frequency representations, such as MFCC (Mel-scale Frequency Cepstral Coefficients) [7], and a widely used classifier is the HMM (hidden Markov model) [8]. Recently, neural networks become the preferable classifier because they have higher detection accuracy for various audio-related applications [9-11]. Previously, we have also studied this problem and reported our results [6].

Although our previous work shows that CNN (convolutional neural network) has good detection accuracy, there are still several problems that need to be addressed. The problems include the followings:

- Preferable type of network structure. In terms of network structure, we may use a CNN, a ResNet [12], a modified capsule net [13], or even an end-to-end network without using the feature extraction step [14, 15]. It is important to know which one is better.
- Suitable number of convolutional layers in CNN structure. It is known that performance degradation may occur if the CNN is very deep [12]. Therefore, it is useful to find a reasonable number of convolutional layers with good performance.
- Preferable time-frequency representation used in conjunction with CNN. Although MFCC is widely used, other types of spectral-temporal features are also used in audio applications, such as spectrogram [16], constant-Q [17], cepstrum [18], and filter bank [19]. Thus, it is important to know, among the available representations, which one is better than others.
- Effectively conducting ensemble learning. The ensemble learning for the present problem can be achieved by using fusion or voting [6]. We need to know which one is better. In addition, it is unclear whether using heterogeneous classifiers is better than using homogeneous ones, and how many classifiers used in the ensemble learning is more cost-effective.

In this paper, we report our recent findings and show that we can further improve the detection accuracy by answering the above questions.

This paper is arranged as follows. Section 2 describes the related work. Section 3 discusses the features and classifiers used in the experiments. Section 4 presents the experiments and

results, and finally section 5 is the conclusion.

2. Related Work

Traditionally, detecting vocal segments includes a feature extraction step and a classification step. The feature extraction step mainly is a time-to-frequency conversion to obtain spectral-temporal features. One of the well-known features is the spectrogram (detailed in section 3.1). However, the spectrogram usually contains many coefficients, and thus it is not efficient for traditional classifiers, such as HMM.

To reduce the dimensionality of the spectrogram, alternative features such as MFCC and MPEG-7 ASP (Audio Spectrum Projection) have been studied in the literature. It is shown that the MFCC features are better [7, 20], but the accuracy is only around 78% for vocal detection problems. Another alternative to spectral-temporal features is to use statistical features. However, according to Berenzweig and Ellis [8], the accuracy of this type of features only reaches 80%. Therefore, it seems that statistical features do not outperform the MFCC features. When augmenting MFCC features with vocal variation and Flutogram variation, Dittmar et al. obtain the F-measure of 87% [21].

If the entire soundtrack is available, it is possible to perform a post-processing step to improve accuracy. There are two types of post-processing approaches reported in the literature. The first one uses an ARMA (autoregressive moving average) smoothing filter to "correct" mis-labeled segments based on neighboring segments [22]. For example, a very short vocal segment between two long nonvocal segments is likely mis-labeled, and thus can be corrected. When a smoothing filter is in use, the accuracy can reach to 84% [22]. The second post-processing approach is the bootstrapping technique [23]. This technique augments the original training set with a portion of classified test set to form a new training set. When trained with the new training set, the new classifier is, in a sense, adapted to the distribution of the test data set. The accuracy increases to around 87% with the bootstrapping technique [23].

When considering the relative accuracy due to classifiers, Leglaive et al. [9] compare the BLSTM (Bidirectional Long Short-Term Memory) classifier with traditional classifiers. With MFCC-like features, BLSTM outperforms traditional classifiers. Their F-measure is up to 91%. Other than BLSTM, Schluter and Grill use a CNN model as the classifier [10]. The chosen features are similar to MFCC. With the data augmentation technique, their model reaches 91% accuracy.

When CNN is used in object detection, it is possible to use a unified, end-to-end neural-network approach [24]. In this approach, the input to the network is a raw image (pixels) without pre-processing steps, such as segmentation. Therefore, Humphrey et al. argue that it is preferred to use an end-to-end network for music informatics [25], because the learning algorithm can learn how to extract audio features (sub-)optimally. Unfortunately, according to Dieleman and Schrauwen [14], such an end-to-end network does not outperform conventional feature-CNN models in music auto-tagging. Actually, based on our previous study [6], the end-to-end network is unable to compete with a feature-CNN model.

Recently, Lee et al. propose a new end-to-end model for music auto-tagging with many layers of small filters [15]. The authors claim that their model is better than the end-to-end model proposed by Dieleman and Schrauwen [14]. As their experiments are mainly for music auto-tagging, it is uncertain whether this model is better in the vocal detection problems. Therefore, we will compare this model with other feature-CNN models to find out the answer.

Although a deeper network typically performs better, unfortunately degradation problem may occur if the network is ultra-deep. Therefore, He et al. show that by using a bypass path,

extremely deep network actual performs better [12]. Therefore, it is important to know a suitable depth of network, including the consideration of the ResNet [12] architecture.

3. Features and Classifiers in the Experiments

3.1 Selection of Features

Among the various spectral-temporal features, we compare the following types: spectrogram [16], MFCC [7], cepstrum [17], constant Q [18], and filter bank (IIRT) [19]. The features are computed with the aid of the Librosa [26] except the spectrogram, which is implemented in a network structure [6]. In the experiments, the input audio clip has duration of 2 seconds, consisting of 32,000 PCM samples. We now briefly explain the computational steps of these features.

The spectrogram is computed using the following steps:

- Audio sequence $x[n]$ is multiplied with a series of Hamming windows. The window has 2048 coefficients. The hop length of the window is 512.
- The spectral coefficients of each segment of windowed samples are computed by FFT (fast Fourier transformation).
- The modulus of each spectral coefficient is one feature value. Totally, there are 63 sets of coefficients, with each set having 1024 real-valued coefficients. Therefore, the feature plane has a size of 63 × 1024.

The computational steps of the MFCC are as follows:

- The audio clip is converted to spectral coefficients using the steps similar to those of computing the spectrogram.
- The default window in the Librosa is a 2048-coefficient Hann window with hop length of 512. Therefore, there are, again, 63 windowed segments.
- The power of frequency coefficients is multiplying with a series of triangular overlapping windows. The window widths are constructed based on Mel-scale. Summing all coefficients within each triangular-window forms the per-band power coefficients.
- Compute the logarithm of the per-band power coefficients, denoted as log power coefficients.
- The log power coefficients are converted to MFCC by using the discrete cosine transform (DCT). The first 80 coefficients are used in the experiments. By using the Librosa library, we have a feature plane of 80 × 63.

The constant-Q transform (CQT) is defined as

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n] x[n] e^{\frac{-j2\pi Qn}{N[k]}} \tag{1}$$

where $N[k]$ is the window length for k-th bin, $W[k,n]$ is a window function, such as Hamming window, and Q is a quality factor related to filter width. In terms of implementation, there is a fast CQT algorithm based on FFT. In our experiments, we use Librosa to save coding time. After the CQT conversion, we have a feature plane of 84 × 63.

Cepstrum is also widely used in speech processing, especially for removing noise or for source separation. For a segment of samples $x[n]$, the cepstrum is computed as

$$x_p[n] = |IFFT\{\log[|FFT(x[n])|^2]\}|^2 \quad (2)$$

In our experiments, as the sample is very long, one set of cepstrum coefficients is computed from one Hamming-windowed segment. Overall, we have an input plane of 62 × 1024.

According to Librosa, the IIRT function returns "a time-frequency representation using a multi-rate filter bank consisting of IIR filters." The filters are designed to have a bandwidth of one semitone, i.e., (1/12)-th octave, and there are 85 filters corresponding to MIDI pitches from 24 to 108 [19]. One banded output value is computed by summing mean-square power of 2,048 samples in a band. Thus, the filter outputs could be an indication of MIDI notes. For this feature, we have a feature plane of 85 × 61.

3.2 Network Models Used in Experiments

In the experiments, we plan to compare the relative accuracy between the end-to-end network and a traditional feature-classifier network. The comparison counterpart is a CNN-based network. In this category, we use three types of network models, to be discussed later.

The end-to-end network is a modified version of SampleCNN [15], as shown in **Fig. 1**. The input to the network is a segment of raw PCM with 31,104 samples. The first convolutional layer has 128 filters with kernel size of 3 and stride of 3. From layer 2 to layer 9, a layer actually is a combination of convolutional layer and max pooling layer (given on the right of the figure). In these layers, the numbers of filters are 128, 256, or 512, the kernel and max pooling sizes are 3, 3, 3, 2, 4, 4, 2, 2, and stride is 1. The next layer is 1×1 layer, and the output layer is a fully connected layer. The activation function for all nodes is relu (rectified linear unit), except the output nodes, which use the softmax function. Also note that the batch normalization [27] is included in the training process.

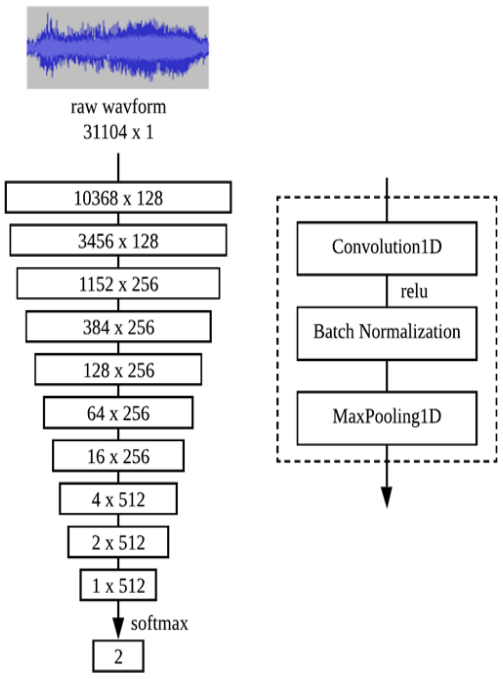

Fig. 1. End-to-end (SampleCNN) model

The feature-classifier network compared in this paper consists of three types. The first type is a feature-CNN model. The feature plane can be a spectrogram, an MFCC plane, or others (see section 3.1 for details). The second type is a modified Capsule net. The third type is a 50-layer ResNet [28], called ResNet50.

To illustrate the first type of network, we use the spectrogram plus CNN (called SCNN in the following) as an example. **Fig. 2** shows an 11-layer SCNN, denoted as SCNN-11. The incoming PCM, through the STSA (short-time spectral analysis) conversion, becomes a spectrogram with 63 time steps and 1024 bins. Although the spectrogram described in section 3.1 is computed with FFT, STSA is actually implemented in network architecture [6]. The spectrogram is the first layer ($63 \times 1024 \times 1$) in **Fig. 2**. The numbers of filters in subsequent layers are 64, 128, or 256, given in the figure. The kernel sizes and max pooling sizes are 3×2 in the first layer, 1×2 in the next seven layers, and then 2×3 in the ninth layer. The final two layers are 1×1 layer and the fully connected output layer with softmax activation function. Note that both network models in **Fig. 1** and **Fig. 2** have the same number of processing layers.

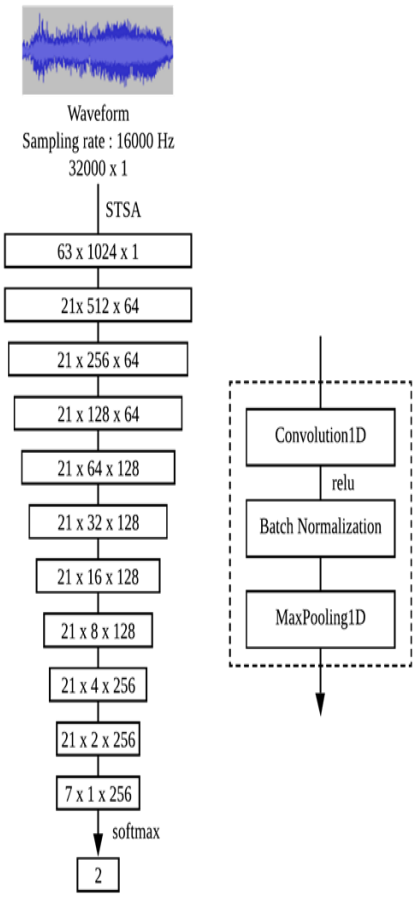

Fig. 2. The SCNN-11 structure

In the experiments, the number of layers is a variable from 11 to 36. To be complete, we also provide the structure of an 18-layer network model, as shown in **Fig. 3**. In the figure, a thick-line box indicates the use of max pooling layer, whereas a thin line box does not have the max pooling layer. Because this network has a similar notation as that in **Fig. 2**, a detailed explanation is omitted here for brevity.

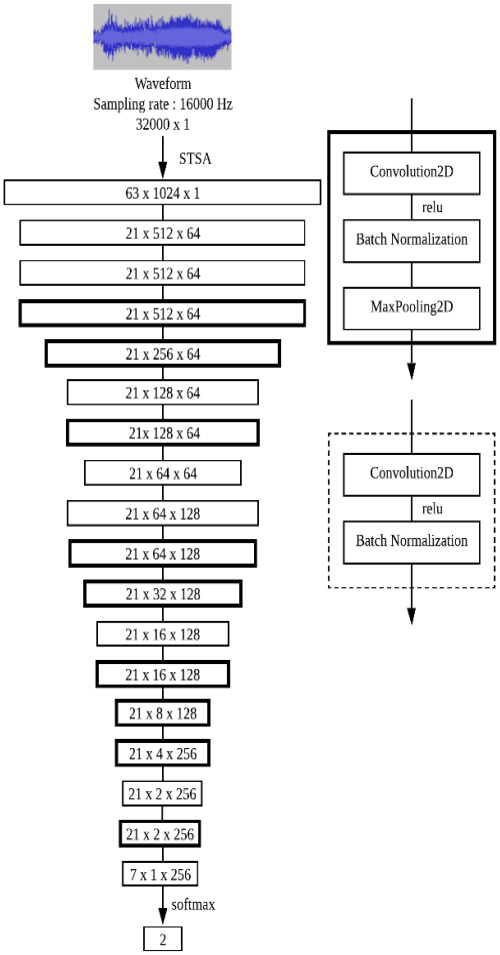

Fig. 3. The SCNN-18 structure

As described in [6], a conventional Capsule net does not have good performance in vocal detection problems. After some literature search, we think that the unsatisfactory performance is likely because the network is not deep enough. Since it is not obvious how to concatenate many primary or digital capsule layers, therefore the only reasonable choice is to use multiple convolutional layers. Consequently, we use 17 convolutional layers in conjunction with capsule layers, as shown in **Fig. 4**. In the network, the box of "CapsNet SCNN" is the multi-layer convolutional sub-network, as shown in **Fig. 5**. Please note that **Fig. 5** is identical to **Fig. 3** except the last two layers. In the following, this network is called CCaNN.

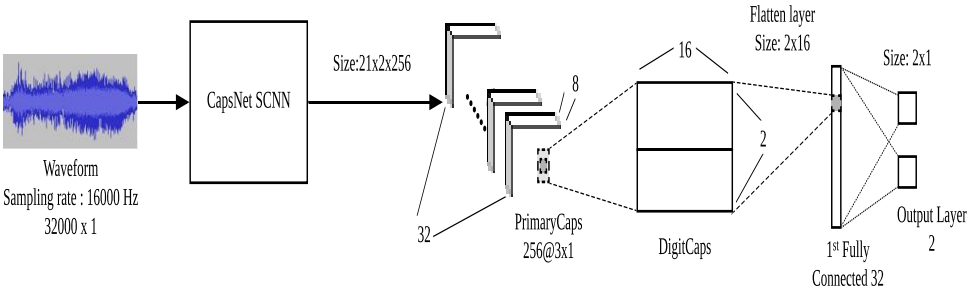

Fig. 4. Convolutional capsule net classifier

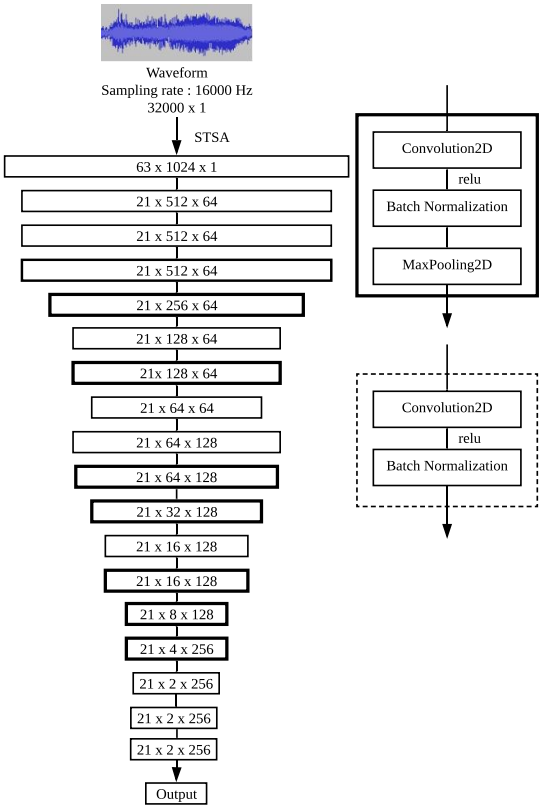

Fig. 5. Convolutional layers for Capsule net

To understand if an ultra-deep network has its advantage, we also adopt the ResNet50 to replace the SCNN for classification. The ResNet50 is a built-in software module in Keras [29], therefore we use it without modification. The structure of the used ResNet50 is shown in **Fig. 6**.

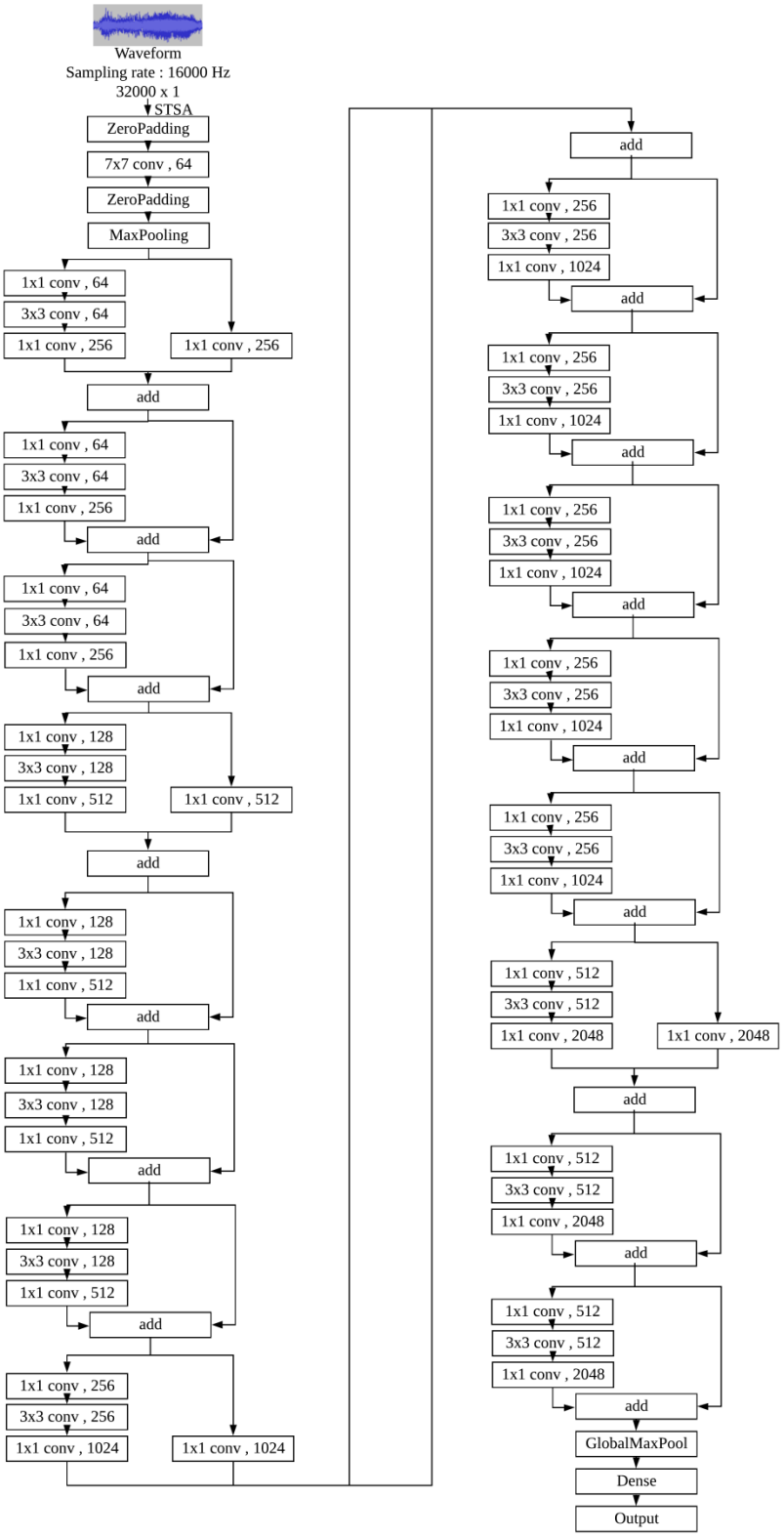

Fig. 6. ResNet50 structure

3.3 Network Models Used in Ensemble Learning

It is known that ensemble learning can effectively improve the accuracy over a single classifier. In the present case, there are three possible ensemble learning strategies: voting, post classifier, and fusion [6]. Previously we have shown that voting is better than fusion [6]. However, in our previous study we had some difficulties to train the network after fusion. Therefore, we plan to compare the performance between voting and fusion strategies again.

Voting is a simple but powerful approach. If we have multiple classifiers available, we may use a majority vote to obtain the final decision. For example, as shown in **Fig. 7**, there are three classifiers used in the detection problem. Because two out of three classifiers have a decision of "vocal," the voted result is "vocal."

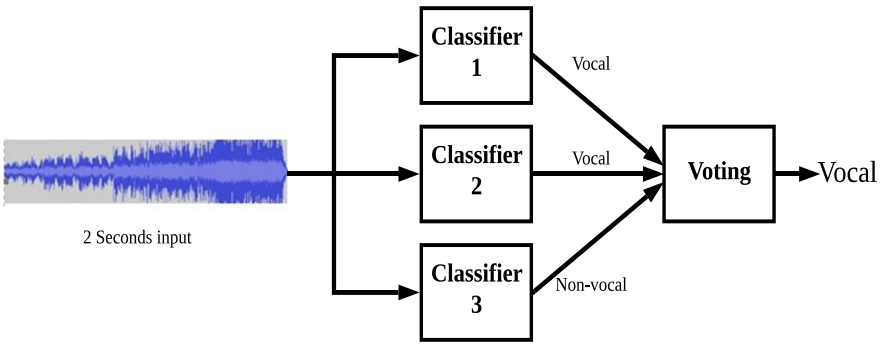

Fig. 7. Voting with multiple classifiers

Another type of ensemble learning is fusion. In this strategy, convolutional layers of multiple models are concatenated with a decision sub-network. **Fig. 8** shows an example of such a strategy. In the figure, Models 1 to 3 contain only the convolutional sub-network, such as the one in **Fig. 5**. The outputs of all three sub-networks, through the concatenation layer and fully-connected (dense) layers, together determine the final decision. Conceptually, this strategy should be better because there is higher flexibility in the fully-connected layers. However, the downside of this strategy is the huge amount of weights to be trained.

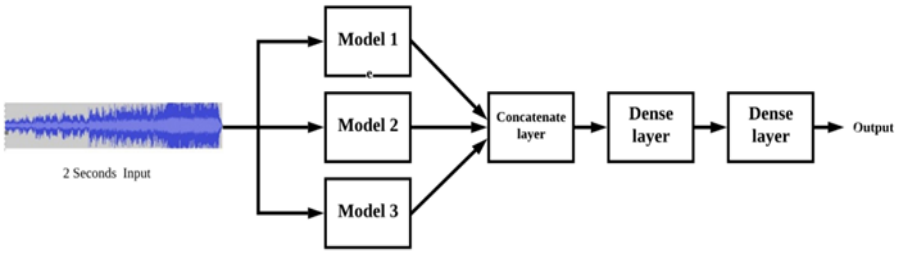

Fig. 8. Fusion of multiple classifiers

4. Experiments and Results

4.1 Experimental Settings and Conducted Experiments

In the experiments, we use two datasets to assess the accuracy. Each audio clip in either dataset has duration of 2 seconds. The first dataset is from Jamendo [30] that contains 93 soundtracks.

We then partition the soundtracks into audio clips. Overall, there are 8,480 vocal and 7,868 non-vocal clips in the training set, and 1,487 vocal and 1,499 non-vocal clips in the test set. The second dataset contains audio clips from the FMA (free music archive) website [31], with hand-made excerption and labeling. During the construction of the FMA dataset, we intentionally excerpt only one audio clip from one soundtrack. The FMA dataset contains 4,783 vocal and 7,451 non-vocal clips in the training set, and 1,660 vocal and 2,485 non-vocal clips in the test set [32].

The simulation programs are developed under the Keras library [29] and the Tensorflow [33] framework. To speed up the computation, GPUs (graphic processing unit) are used during training. To ensure good training results, we use ADADELTA [34] as the optimizer and dropout regularization [35]. The probability for dropout is set to 0.5. The training epoch for each trial is 200. Unless otherwise specified, the reported accuracy is an average of 10 trials.

When conducting the experiments, we use one training set (either Jamendo or FMA) to test both test sets. To save space, unless otherwise specified, we only report the accuracy of both test sets with the Jamendo training set. Note that in a typical situation, we divide the entire dataset into training, validation, and test sets. In our case, because we do not use a validation set, we use both test sets to validate the claim. If both agree, the claim is valid. For example, if SCNN is better than SampleCNN in both test sets, we claim that SCNN is better. In a sense, it is similar to that we use the first test set as the validation set and the second one as the true test set. However, our test approach is more difficult because the second test set is from a different data source. Therefore, our test approach can better assess the generalization capability.

The conducted experiments include the followings. Experiment one compares the performance between a conventional feature-classifier model and an end-to-end model (section 4.2). The second experiment studies the accuracy against the number of convolutional layers (section 4.3). The next experiment determines which time-frequency representation is better (section 4.4). It is followed by experiment four to compare the performance between SCNN, CCaNN, and ResNet50 (section 4.5). The final two experiments are related to ensemble learning. Experiment five is used to determine a cost-effective way to perform ensemble learning (section 4.6), and experiment six compares the relative accuracy between using homogeneous and heterogeneous classifiers (section 4.7). Finally, we compare the achieved performance with that of previous models (section 4.8).

4.2 Comparison between End-to-end and Feature-classifier Approaches

The first experiment is to compare the relative accuracy between an end-to-end model and a traditional feature-classifier model. In this experiment, the end-to-end network is the SampleCNN given in **Fig. 1** and the feature-classifier is the SCNN-11, given in **Fig. 2**. The experimental results are listed in **Table 1**. In the table, SCNN-4 is the network model we used in [6]. It is also an SCNN model, but with only four convolutional layers. It is clear that the feature-classifier model still outperforms the end-to-end model for the same depth. In addition, this experiment also shows that a deeper network SCNN-11 has higher accuracy than a less-deep one, SCNN-4.

Table 1. Comparison between end-to-end and feature-classifier models

Test set	SampleCNN	SCNN-11	SCNN-4 [6]
Jamendo	86.46%	**92.22%**	91.80%
FMA	77.83%	**82.32%**	81.69%

4.3 Accuracy with Different Layer Depth

The second experiment is to determine a suitable number of convolutional layers. Although a deeper network may perform better, unfortunately degradation problem may occur if the network is overly deep [12]. Therefore, we need to experimentally determine a suitable number of layers. In this experiment, the number of convolutional layers is varied from 11 to 36. The experimental results are given in **Fig. 9** and **Fig. 10** for both test sets. The results show that an 18-layer network is better.

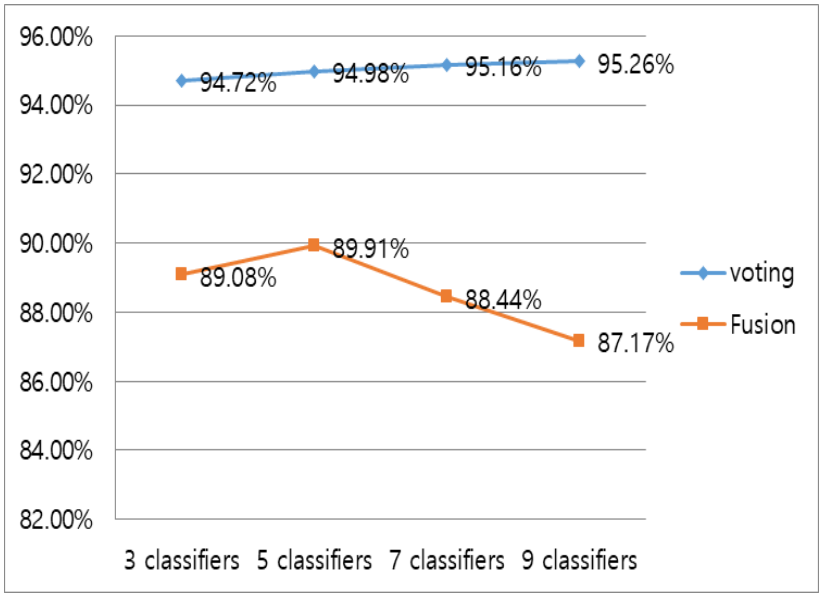

Fig. 9. Jamendo test accuracy against number of layers

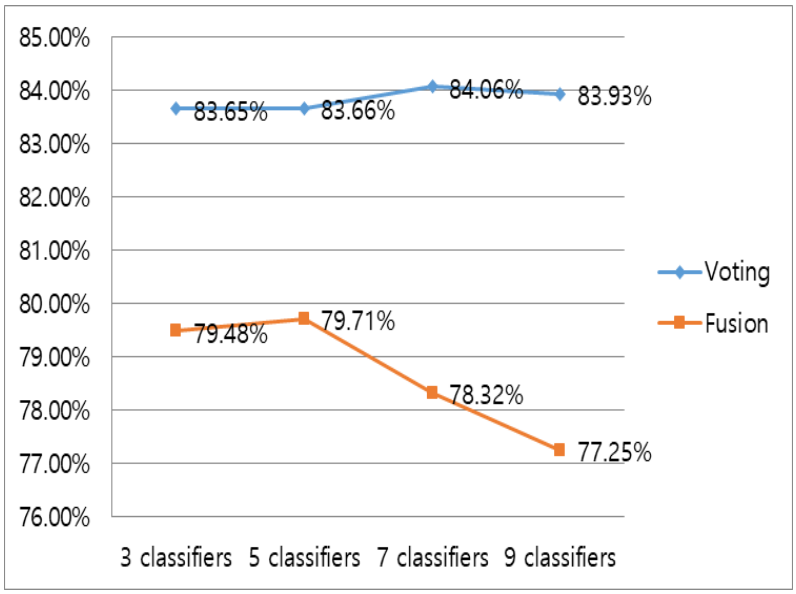

Fig. 10. FMA test accuracy against number of layers

4.4 Accuracy with Different Features

Given that a CNN with 18 layers is better, the next question to ask is which particular time-frequency representation is better than others. Other than spectrogram, the examined features include MFCC, constant-Q, cepstrum, or IIRT. We call these models as MCNN, QCNN, CCNN, and ICNN, respectively. This experiment is based on the network model of **Fig. 3** with necessary adjustment of kernel and pooling sizes to fit different sizes of feature planes. To save space, the detailed network parameters are omitted here. The experimental results are given in **Fig. 11**. The results show that a spectrogram is indeed a better choice. Also note that the MFCC feature is the second best, better than the CQT and IIRT features. This experiment, in a sense, also explains the reason that MFCC is widely used in speech applications. It is also noted that the cepstrum features have relative poor accuracy. When carefully examining the cepstrum features, we found that the logarithm operation produces some extremely large coefficients when a short pulse (silence) is present in the audio clip. This situation partially leads to low accuracy.

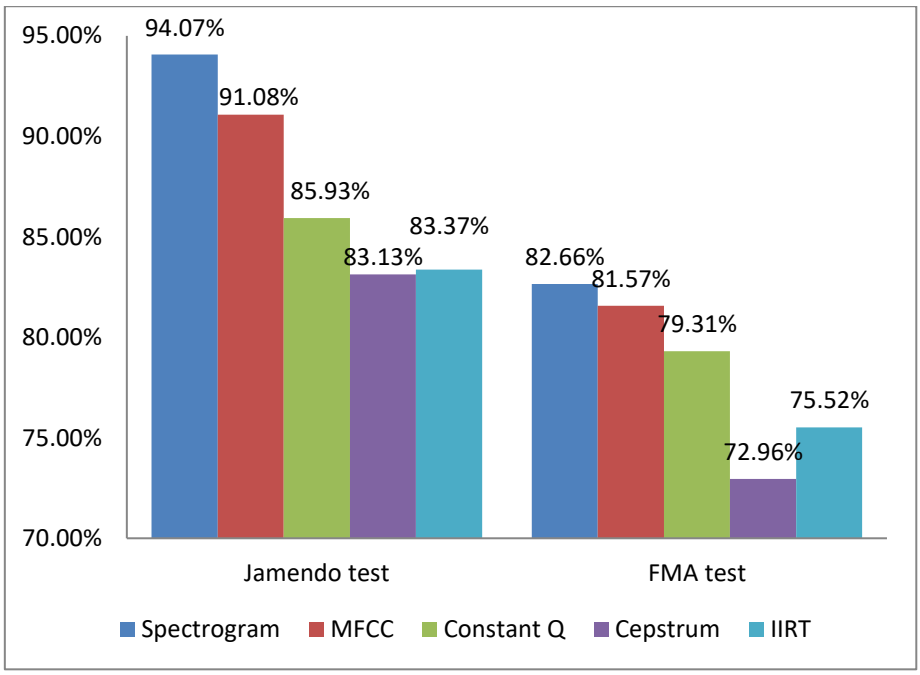

Fig. 11. Accuracy with different features

4.5 Accuracy with Different Classifier Structure

We compare the performance of various network structures in this section. The network structures under comparison are SCNN-18, CCaNN, and RestNet50. The SCNN-18 model is in **Fig. 3**, the CCaNN is in **Fig. 4**, and the ResNet50 is in **Fig. 6**. **Fig. 12** shows the experimental results, which indicate that a conventional CNN is still better than a modified capsule net. In addition, the performance of ResNet50 is not as good as expected. It is partly because the ResNet50 is optimized for a square-like input plane. Unfortunately, the incoming spectrogram is a thin and long rectangle. Judging from this point, we conclude that the design of network structure strongly affects the performance, even more important than the depth of the network.

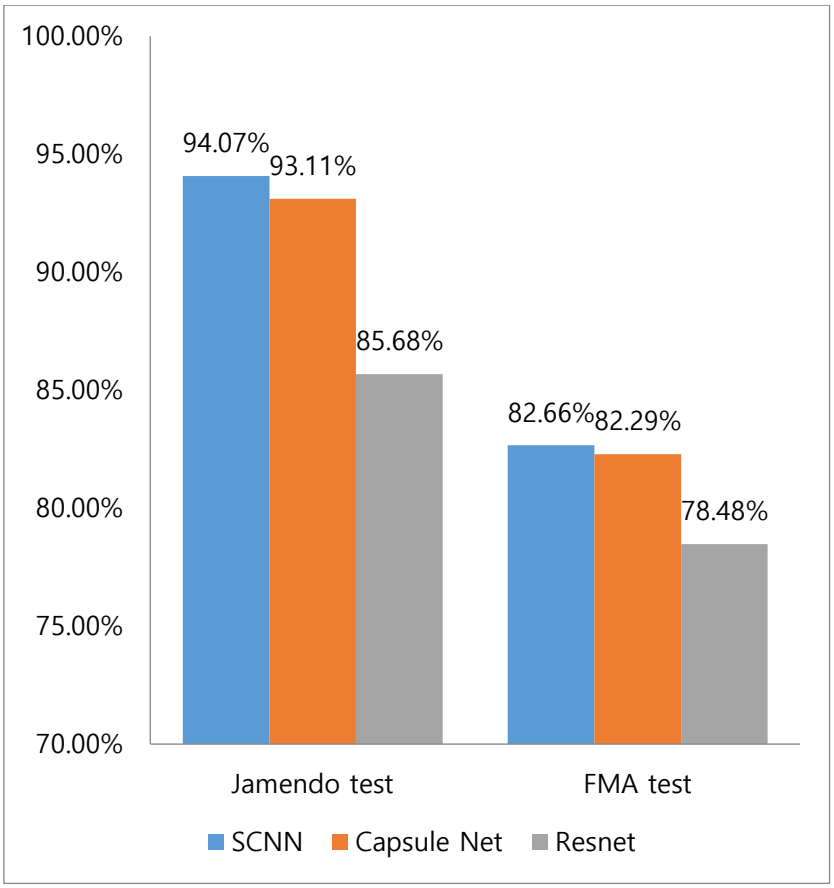

Fig. 12. Accuracy with different network structures

4.6 Accuracy vs Number of Homogeneous Classifiers in Ensemble Learning

Previously we reported that using ensemble learning can effectively improve the accuracy [6]. Conceptually, as the number of classifiers increases, the accuracy should increase. However, with the increase of classifiers, the computational burden to train and test multiple classifiers becomes a serious problem. Therefore, we intend to find a good trade-off between accuracy and required computation. To this end, we use the SCNN-18 structure with different initial weights for voting or fusion. The experimental results are given in **Fig. 13** and **Fig. 14**. From the figures, we have the following observations.

- Fusion is not as good as voting. It is likely due to the large amount of weights to be trained. Therefore, it is more difficult to train the fusion network.
- Although the accuracy improves along with the number of classifiers, the improvement from 7 to 9 is marginal in Jamendo test set, and the accuracy actually decreases in the FMA test set.

Therefore, it seems safe to conclude that using fusion is not an effective ensemble learning method. For voting, using 7 classifiers are sufficient for both test sets.

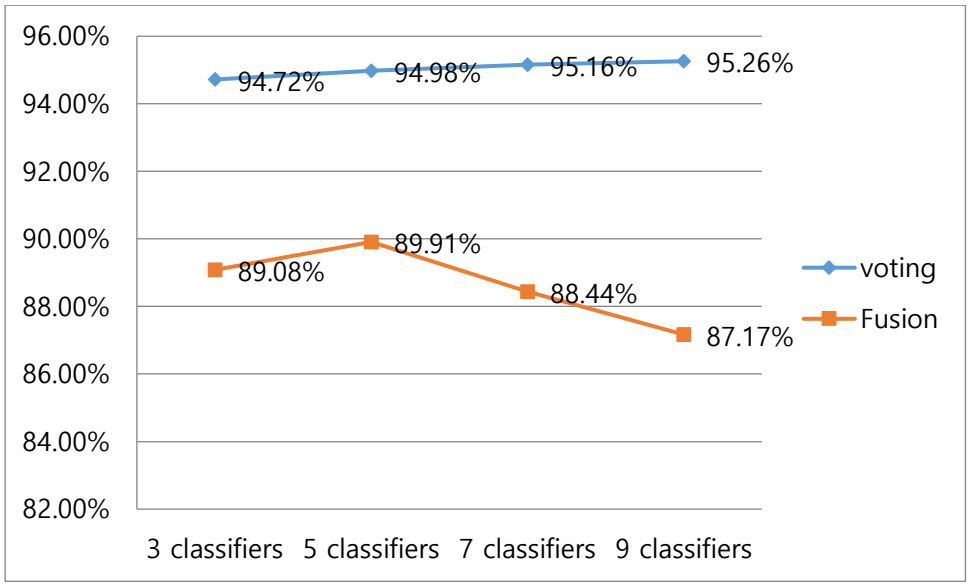

Fig. 13. Accuracy with Jamendo test

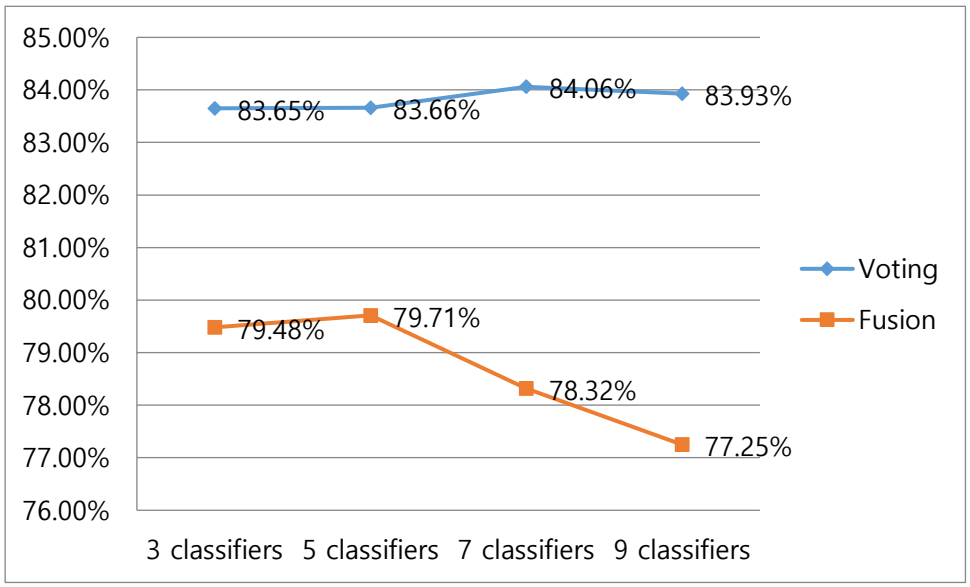

Fig. 14. Accuracy with FMA test

4.7 Homogeneous vs Heterogeneous Classifiers in Ensemble Learning

In this experiment, we compare the accuracy improvement with voting between using homogeneous and heterogeneous classifiers. Homogeneous classifiers are voting from five SCNN-18 classifiers with different initial weights, as given in section 4.6. Heterogeneous classifiers include SCNN, MCNN, QCNN, CCNN, ICNN, and CCaNN. In this experiment, the heterogeneous classifiers are divided into three groups.

- Group one: SCNN, MCNN, and QCNN.
- Group two: SCNN, MCNN, QCNN, ICNN, and CCNN.
- Group three: SCNN, MCNN, QCNN, ICNN, and CCaNN.

Note that CCNN(cepstrum-CNN) in group two is replaced with CCaNN in group three because CCNN has lower detection accuracy.

The experimental results are given in **Fig. 15**. Then, we have the following observations.

- Voting with five heterogeneous classifiers (group 3) is better than with three classifiers (group 1) if no voting member has much lower accuracy.
- Voting with heterogeneous classifiers may not outperform voting with homogeneous classifiers. Homogeneous classifiers perform better if the test samples are from the same source as the training samples, although heterogeneous classifiers seem to have better generalization capability.

With the above observations and results in section 4.6, we recommend using seven homogeneous classifiers with voting. With this arrangement, programmers can reuse their programs without additional coding efforts. Thus, it becomes an easy task to trade a longer training time for higher accuracy.

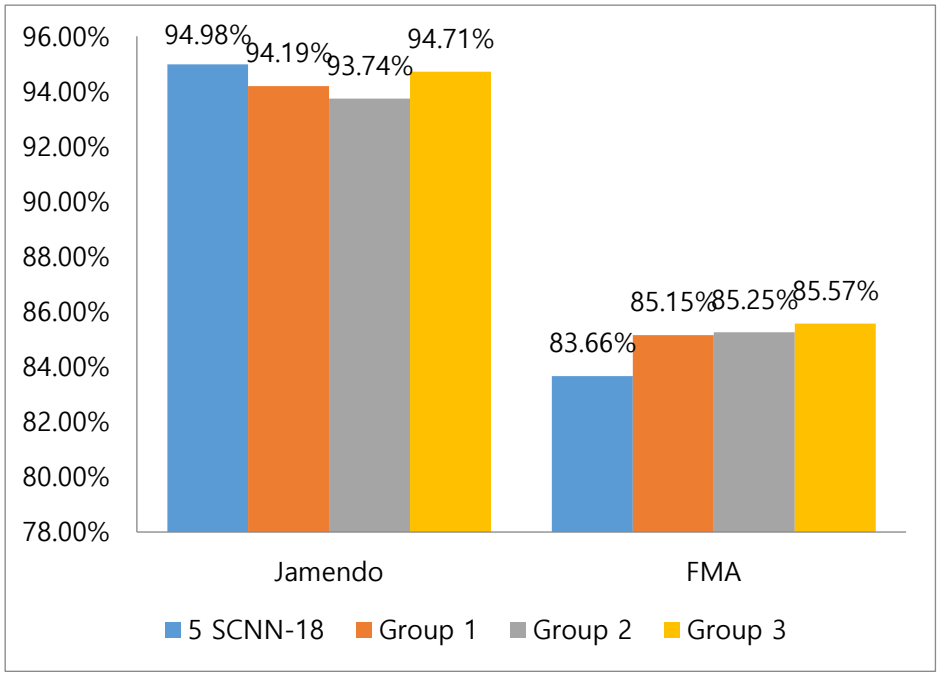

Fig. 15. Accuracy with heterogeneous voting

4.8 Comparison with Existing Results

As we mentioned in the introduction, our vocal detection problem is slightly different from the conventional one. Therefore, it is not so easy to have a fair comparison with other researcher's results. We previously followed the structure of [10] with some adjustments and published our results in [6]. Though not precise, we still use the results, denoted as MCNN-4, in our comparison here. Another target, SCNN-4, is the network model which we also used in [6]. The accuracy of the proposed model versus previous models is given in **Table 2**. In the table, SCNN trained with the Jamendo training set is used to test Jamendo and FMA test sets, and the same procedure for the FMA training set. The results show that the proposed approach improves the detection accuracy by about 2.3% in Jamendo dataset when compared with SCNN-4. This amount of improvement is significant because the accuracy of SCNN-4 is almost 92%. Thus, even 1% of improvement is difficult.

Table 2. Accuracy comparison between the current version and previous version

Training	Test	SCNN-18	SCNN-4 [6]	MCNN-4 [6]
Jamendo	Jamendo	**94.07%**	91.80%	88.20%
	FMA	**82.66%**	81.69%	76.48%
FMA	Jamendo	**92.54 %**	89.62%	85.72%
	FMA	**89.82 %**	88.20%	83.70%

When ensemble learning is used, we compare the accuracy of voting with SCNN-18 and that of the previous results [6]. Previously, we used five heterogeneous classifiers to vote, denoted as H-5. To have a fair comparison, we also choose to use 5 SCNN-18 to vote. **Table 3** shows the results. In the table, models trained with Jamendo training set is used to test only Jamendo test set, and the same for the FMA dataset. The results show that the performance improvement is between 0.8% and 2.3 %. Again, the results show that it is not easy to have large improvement for a dataset with high accuracy.

Table 3. Comparison of accuracy with voting for various methods

Dataset	5 SCNN-18	H-5 [6]
Jamendo train/test	**94.98%**	94.2 %
FMA train/test	**90.96%**	88.6 %

4.9 Limitations of the Proposed Approach

- The input audio must have sufficient duration, say 1.6 s. For some applications, we prefer to detect the presence of vocal signal with very short duration, such as 100 ms. In that case, the proposed approach is not able to deal with.
- When ensemble learning is used, we suggest using multiple identical network models with different initial weights. This approach can achieve good performance if the training and test datasets have similar distributions. If the distributions of the training and test datasets are somewhat different, the proposed approach may not provide sufficient generalization. This situation is also shown in **Fig. 15**. Thus, the proposed ensemble approach works best if the training samples are uniformly drawn from the entire data set to be examined.

5. Conclusion

In this paper, we present our experimental results toward higher accuracy for vocal detection problem. The experimental results show that using a spectrogram as features to an 18-layer CNN still has higher accuracy than other models, including a state-of-the-art, end-to-end model. When compared with our previous work, we have an improvement of almost 2.3%. In terms of ensemble learning, surprisingly a simple majority vote is better than fusion. Furthermore, in contrast to common belief, our simulation results show that it is inconclusive whether using heterogeneous classifiers could outperform the homogeneous ones in voting. Thus, we suggest using seven homogeneous classifiers in voting if computational burden is affordable. Doing so, we can further boost the accuracy by 1.1%. As the Jamendo dataset already has accuracy of more than 90%, an improvement of 1% or 2 % is difficult and valuable.

References

[1] S. D. You, Y. C. Wu, and S.H. Peng, "Comparative study of singing voice detection methods," *Multimedia Tools and Applications*, vol. 75, no. 23, pp. 15509-15524, Dec. 2016. Article (CrossRef Link)

[2] C. L. Hsu, D. L. Wang, J. S. R. Jang, and K. Hu, "A tandem algorithm for singing pitch extraction and voice separation from music accompaniment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 5, pp. 1482-1491, 2012. Article (CrossRef Link)

[3] B. Logan and S. Chu, "Music summarization using key phrases," in *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 11749-11752, 2000. Article (CrossRef Link)

[4] J. Salamon, E. Gomez, D. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications and challenge," *IEEE Signal Processing magazine*, vol. 31, no. 2, pp. 118-134, 2014. Article (CrossRef Link)

[5] Y. E. Kim and B. Whitman, "Singer identification in popular music recordings using voice coding features," in *Proc. of the International Society for Music Information Retrieval(ISMIR) Conference*, 2002. Article (CrossRef Link)

[6] S. D. You, C. H. Liu, and W. K. Chen, "Comparative study of singing voice detection based on deep neural networks and ensemble learning," *Human-centric Computing and Information Sciences*, vol. 8, no. 34, 2018. Article (CrossRef Link)

[7] H. G. Kim and T. Sikora, "Comparison of MPEG-7 audio spectrum projection features and MFCC applied to speaker recognition, sound classification, and audio segmentation," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP*), 2004. Article (CrossRef Link)

[8] A. L. Berenzweig and D. P. W. Ellis, "Locating singing voice segments within music signals," in *Proc. of the IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pp. 119-122, 2001. Article (CrossRef Link)

[9] S. Leglaive, R. Hennequin, and R. Badeau, "Singing voice detection with deep recurrent neural networks," in *Proc. of 2015 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*, pp. 121-125, 2015. Article (CrossRef Link)

[10] J. Schlüter and T. Grill, "Exploring data augmentation for improved singing voice detection with neural networks," in *Proc. of 16th International ISMIR Conference*, pp. 121-126, 2015. Article (CrossRef Link)

[11] M. Lim, D. Lee, H. Park, Y. Kang, J. Oh, J. S. Park, and J. H. Kim, "Convolutional neural network based audio event classification," *KSII Transactions on Internet & Information Systems*, vol. 12, no. 6, pp. 2748-2760, 2018. Article (CrossRef Link)

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016. Article (CrossRef Link)

[13] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *arXiv prepr. arXiv:1710.09829*, 2017. Article (CrossRef Link)

[14] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *Proc. of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6964-6968, 2014. Article (CrossRef Link)

[15] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, pp. 150-153, 2018. Article (CrossRef Link)

[16] Spectrogram. [Online]. Available: https://en.wikipedia.org/wiki/spectrogram

[17] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *Journal of Acoustical Society of America*, vol. 92, no. 5, pp. 2698 -2701, 1992. Article (CrossRef Link)

[18] Cepstrum. [Online]. Available: https://en.wikipedia.org/wiki/cepstrum

[19] Librosa.iirt. [Online] . Available: https://librosa.org/doc/0.8.0/generated/librosa.iirt.html

[20] M. Rocamora and P. Herrera, "Comparing audio descriptors for singing voice detection in music audio files," in *Proc. of the 11th Brazilian Symposium on Computer Music*, pp. 187-196, 2007. Article (CrossRef Link)
[21] C. Dittmar, B. Lehner, and T. Prätzlich, "Cross-version singing voice detection in classical opera recordings," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR)*, pp. 618-624, 2015. Article (CrossRef Link)
[22] H. Lukashevich, M. Gruhne, and C. Dittmar, "Effective singing voice detection in popular music using ARMA filtering," in *Proc. of the 10th International Conference on Digital Audio Effects*, pp. 165-168, 2007. Article (CrossRef Link)
[23] T. L. Nwe, A. Shenoy, and Y. Wang, "Singing voice detection in popular music," in *Proc. of the 12th Annual ACM International Conference on Multimedia*, pp. 324-327, 2004. Article (CrossRef Link)
[24] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv:1804.02767*, 2018. Article (CrossRef Link)
[25] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," in *Proc. of the 13th International Society for Music Information Retrieval Conference*, pp. 403-408, 2012. Article (CrossRef Link)
[26] Librosa. [Online]. Available: https://github.com/librosa/librosa
[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. Article (CrossRef Link)
[28] ResNet. [Online]. Available: https://keras.io/api/applications/ResNet/#ResNet50-function
[29] Keras. [Online]. Available: https://keras.io/
[30] M. Ramona, G. Richard and B. David, "Vocal detection in music with support vector machines," in *Proc. of 2008 IEEE International Conference Acoustics, Speech and Signal Processing*, pp. 1885-1888, 2008. Article (CrossRef Link)
[31] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proc. of the International Society for Music Information Retrieval Conference (ISMIR 2017)*, pp. 316-323, 2017. Article (CrossRef Link)
[32] NTUT-LabASPL. [Online] Available: https://github.com/NTUT-LabASPL
[33] Tensorflow. [Online] Available: https://www.tensorflow.org
[34] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," *arXiv Preprint, arXiv1212.5701*, 2012. Article (CrossRef Link)
[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014. Article (CrossRef Link)

Shingchern D. You received the Ph.D. degree in Electrical Engineering from the University of California, Davis, CA, USA in 1993. He is now a professor in the Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan. Dr. You's research interests include machine learning and applications of signal processing to audio and communication systems.

Chien-Hung Liu received his M.S. degree in Electrical Engineering from University of Southern California in 1994, and his Ph.D. degree in Computer Science and Engineering from University of Texas at Arlington in 2002. He is currently an associate professor of Computer Science and Information Engineering Department at National Taipei University of Technology, Taiwan. His research interests include software testing, vocal detection, and software engineering.

Jia-Wei Lin received his M.S. degree in Computer Science and Information Engineering from National Taipei University of Technology, Taiwan in 2020. He is currently a software engineer. His research interests include vocal detection, deep learning applications, and software testing.